Generic Plaintext Equality and Inequality Proofs

Olivier Blazy, Xavier Bultel, Pascal Lafourcade, Octavio Perez-Kempner

Generic Plaintext Equality and Inequality Proofs (Extended Version)

Olivier Blazy[1], Xavier Bultel[2], Pascal Lafourcade[3] and Octavio Perez Kempner[4,5]

[1] Université de Limoges, XLIM, Limoges, France
olivier.blazy@unilim.fr
[2] INSA Centre Val de Loire, LIFO lab, France
xavier.bultel@insa-cvl.fr
[3] University Clermont Auvergne, LIMOS, France
pascal.lafourcade@uca.fr
[4] DIENS, École normale supérieure, CNRS, PSL University, Paris, France
[5] be-ys Research, France
octavio.perez.kempner@ens.fr

Abstract. Given two ciphertexts generated with a public-key encryption scheme, the problem of plaintext equality consists in determining whether the ciphertexts hold the same value. Similarly, the problem of plaintext inequality consists in deciding whether they hold a different value. Previous work has focused on building new schemes or extending existing ones to include support for plaintext equality/inequality. We propose generic and simple zero-knowledge proofs for both problems, which can be instantiated with various schemes. First, we consider the context where a prover with access to the secret key wants to convince a verifier, who has access to the ciphertexts, on the equality/inequality without revealing information about the plaintexts. We also consider the case where the prover knows the encryption's randomness instead of the secret key. For plaintext equality, we also propose sigma protocols that lead to non-interactive zero-knowledge proofs. To prove our protocols' security, we formalize notions related to malleability in the context of public-key encryption and provide definitions of their own interest.

Keywords: Public-key encryption, Generic plaintext equality, Generic plaintext inequality, Zero-knowledge proofs.

1 Introduction

The problem of proving whether two given ciphertexts encrypt the same or different messages is known as plaintext equality (or inequality) proofs. Considering public-key encryption (PKE), there are scenarios in which deciding equality can easily be done. For instance, if both ciphertexts were generated using the same key and the encryption scheme is deterministic or if access to a trusted third

party, who knows the private key, is provided. However, in practical scenarios, where a prover needs to convince a verifier of the equality or inequality of plaintexts, require stronger guarantees (*i.e.*, the verifier must learn no additional information than the yes or no answer to the problem).

Well-known examples include the use of such proofs in voting protocols [35,34,14], reputation systems [22,15] and cloud-based applications [33]. Additionally, protocols with broadcasting phases where one of the parties needs to broadcast encrypted messages under different keys to several parties can also benefit from these proofs. A less common example involves a client which needs to regularly store encrypted information in a backup server (or in a distributed database such as blockchain), while being able to convince any third party of minimal claims about it. Furthermore, in settings where online interaction between the parties is not desirable or public verifiability is preferred, non-interactive variants can also be very useful.

Sometimes equality or inequality proofs are used as subroutines and need to be integrated with other software. Therefore, having flexible alternatives (*e.g.*, without relying on specific constructions that require particular configurations or specific hardware) is essential to overcome possible conflicts Thus, generic protocols that can be implemented with different PKE schemes, making them more flexible than their customized variants and more suitable to be integrated into different settings, are proposed.

We focus on two-party protocols, where two ciphertexts and auxiliary inputs are given. The prover attempts to convince a verifier on either the plaintext equality or inequality of the ciphertexts. The prover and the verifier share a common input consisting of a set of public keys and ciphertexts generated with those keys. The prover also knows the corresponding set of secret keys or the randomness used to encrypt the plaintexts. As previously mentioned, our aim is to design generic plaintext equality or inequality protocols in this setting.

Contributions. Using randomization properties of PKE schemes, we build secure generic zero-knowledge protocols from standard techniques. Our first contribution introduces different notions related to the concept of malleability in public-key encryption and their formalization. To that end, we make a clear distinction between how a ciphertext can be randomized (*e.g.* the ciphertext alone, the plaintext message or concerning the corresponding key). As a result, we characterize PKE schemes in terms of generic randomizable encryption properties, which we use to build our protocols. Our second contribution is the construction of two interactive zero-knowledge protocols, Π_{PEQ} and Π_{PINEQ}, for plaintext equality and inequality. These protocols are secure against malicious verifiers. For each of them we first present a weaker variant (Π_{HPEQ} and Π_{HPINEQ} respectively) which is only secure against honest verifiers. The protocol Π_{PEQ} requires the PKE scheme to allow the randomization of both, the ciphertext and the corresponding plaintext message. In contrast, the protocol Π_{PINEQ} only requires the former one. Our third contribution is plaintext equality protocols based on proofs of knowledge of the secret key (protocols Π_{MATCHPEQ} and Π_{SIGPEQ}), or of the randomness used for the encryption (protocol Π_{RSPEQ}). Either case admits non-interactive versions ap-

plying the Fiat-Shamir transform, but both require schemes with less common properties. The schemes need to be key-randomizable or random coin decryptable. We base our protocols on simple properties, making them independent of a particular scheme and therefore generic. To support this claim, we list various schemes indicating the relation with our definitions and protocols.

Finally, we also see an added value of our contributions in terms of serving as a pedagogical tool to present zero-knowledge protocols (ZKP). Usual examples to introduce ZKP are graph 3-coloring or graph isomorphism. Although such protocols can be explained without requiring any advanced cryptographic knowledge, they are not used in real-world applications. On the contrary, the protocols that we present are very intuitive, can easily be explained without requiring advanced cryptographic knowledge outside the concept of public key encryption, and are also useful for real-world applications of ZKP. For this reason, we think our protocols can serve as a convincing pedagogical example to explain ZKP to a larger audience who has little mathematical background. With this in mind, as a side contribution, we present a physical protocol using simple objects (boxes and padlocks) to explain how our first proof of plaintext inequality works.

Related Work. Jakobsson et al. [25] introduced the concept of distributed plaintext equality test (PETs), which allows $n > 1$ parties to determine whether two ElGamal ciphertexts encrypt the same or different message without learning it, but given knowledge of the secret key and assuming at least one of the parties is honest. Very recently, McMurtry et al. [27] showed that several follow up works based on the PET from [25] are flawed (because they use it as if no trusting assumptions where needed), and showed how to fix it. Choi et al. [11] proposed zero-knowledge equality/inequality proofs for boolean ElGamal ciphertexts with knowledge of the secret key. In their work, the randomness used to produce the two ciphertexts is required. Parkes et al. [29] proposed zero-knowledge equality/inequality proofs for Paillier ciphertexts given access to the randomness used to produce the ciphertexts or access to the plaintexts. In [3], Blazy et al. introduced a generic approach to prove a non-membership concerning some language in non-interactive zero-knowledge. They showed how to prove plaintext inequality of two ElGamal ciphertexts, given that the prover knows the plaintext and the randomness used to produce one of the ciphertexts. More recently, Blazy et al. [4] introduced a generic technique for non-interactive zero-knowledge plaintext equality/inequality proofs in which the prover is given two ciphertexts and trapdoor information. In such a scenario, none of the parties has access to the secret key nor the randomness used to produce the ciphertexts. While being *generic*, those constructions [3,4] require a specific kind of zero-knowledge proofs. More precisely, they need to build a zero-knowledge proof showing that a zero-knowledge proof was computed honestly. While this design works elegantly with pairing-based cryptography (as Groth-Sahai proofs [21] allows to prove in zero-knowledge a pairing-product equation, while also being verifiable with a pairing product equation), this often fails (or requires ad-hoc constructions that are far from being efficient) in other settings. For example, when considering Schnorr [36] proofs, the random oracle prevents any kind of chaining. There-

fore, another design is required to allow such functionality. Extensions for PKE schemes such that given a plaintext, a ciphertext and a public key, it is universally possible to check whether the ciphertext encrypts the plaintext under the key also exists. Such an extension has been proposed by Canard et al. [8] under the name of Plaintext Checkable Encryption. There are also different works proposing schemes to support plaintext equality tests from user-specified authorization. For instance, in [37], two users who have their keys can issue tokens to a proxy to authorize it to perform the plaintext equality test for their ciphertexts. Yang et al. [38] constructed a probabilistic scheme that allows anyone provided with two ciphertexts to check if they encrypt the same message, considering that the ciphertexts may not have been generated with the same key. They do this achieving a weak form of ind-cca.

Previous work rests on specific constructions, which do not allow the scheme to be separated from the protocol's requirements. Our approach is different because we first seek to define protocols independently of the scheme and then to present, which are the necessary conditions for a scheme to instantiate them. As a result and unlike prior work, we present many protocols which can be integrated with existing pieces of software just as if they were templates allowing one to switch from one scheme to another more easily. To compare the efficiency of our protocols with custom variants, we discuss the case of ElGamal.

Outline. § 2 provides the required background. § 3 defines new notions for generic randomizable encryption. In § 4, we present generic interactive protocols for both, plaintext equality and inequality. In § 5, under different assumptions, we present generic protocols for plaintext equality and discuss how to define non-interactive versions in the random oracle model. Before concluding, we discuss the efficiency of our protocols in § 6.

2 Cryptographic Background

Definition 1 (Public-key encryption scheme [31]). *A public-key encryption (PKE) scheme* $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *is a triple of (possibly randomized) efficient algorithms that verifies the following:*

1. $\mathsf{KGen}(1^k)$ *is a p.p.t algorithm that on input the security parameter* k*, produces a key pair* $(\mathsf{pk}, \mathsf{sk})$.
2. $\mathsf{Enc}_{\mathsf{pk}}(\mathsf{m}; \mathsf{r})$ *is a p.p.t algorithm that given a message* m*, a random coin* r *and* pk *produces a ciphertext* c.
3. $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{c})$ *is a deterministic algorithm that given a ciphertext* c *and* sk *produces a message* m.
4. Correctness*: The triple should be such that the following holds for every valid message defined in the message space and every security parameter:*
   $Pr\left[(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}(1^k) : \mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(\mathsf{m})) = \mathsf{m}\right] = 1.$

By convention, we denote the set of the plaintexts (resp. public keys, random coins, ciphertexts) by $\mathcal{M}$ *(resp.* $\mathcal{K}$*,* $\mathcal{R}$*,* $\mathcal{C}$*).*

Definition 2 (Random Coin Decryptable PKE (RCD-PKE) [6]). *A probabilistic PKE scheme is Random Coin Decryptable if there exists a polynomial-time algorithm* CDec *such that for any public key* $\mathsf{pk} \in \mathcal{K}$*, any* $\mathsf{m} \in \mathcal{M}$*, and any random coin* σ*, the following equation holds:* $\mathsf{CDec}_\sigma(\mathsf{Enc}_{\mathsf{pk}}(\mathsf{m};\sigma),\mathsf{pk}) = \mathsf{m}$.

For interactive machines $\mathcal{P}$ (the prover) and $\mathcal{V}$ (the verifier), we denote as in [30] that $\langle \mathcal{P}(y), \mathcal{V}(z)\rangle(x)$ is the random variable representing $\mathcal{V}$'s output when interacting with $\mathcal{P}$ on common input x, when the random input to each machine is uniformly and independently chosen with z and y being auxiliary inputs. We also denote a witness relation for a language $L \in \mathcal{NP}$ as R_L and say that y is a witness for the membership $x \in L$ if $(x, y) \in R_L$.

Definition 3 (Interactive Proof System [30,18]). *Let* ϵ_c*,* ϵ_s*:* $\mathbb{N} \to [0, 1)$ *such that both are computable in poly*(ℓ)*-time and* $\epsilon_c(\ell) + \epsilon_s(l) < 1 - poly(\ell)^{-1}$*.* $(\mathcal{P}, \mathcal{V})$ *is called an interactive proof system for the language* L *with completeness and soundness errors* ϵ_c *and* ϵ_s*, if* $\mathcal{V}$ *is p.p.t and the following conditions hold:*

- *Completeness: For every* $x \in L$ *there exists a (witness) string* y *such that for every auxiliary input* $z \in \{0,1\}^* : Pr[\langle \mathcal{P}(y), \mathcal{V}(z)\rangle(x) = 1] = 1 - \epsilon_c(|x|)$*.*
- *Soundness: For every* $x \notin L$*, every interactive machine* $\mathcal{B}$*, and every* $y, z \in \{0,1\}^* : Pr[\langle \mathcal{B}(y), \mathcal{V}(z)\rangle(x) = 1] \leq \epsilon_s(|x|)$*.*

If $\epsilon_c \equiv 0$*, we say the system has* perfect completeness*. If the soundness condition is required to hold only with respect to a computationally bounded prover* $\mathcal{B}$*,* $(\mathcal{P}, \mathcal{V})$ *is called an interactive* argument *system.*

Definition 4 (Sigma protocol [23]). *An interactive proof system* $(\mathcal{P}, \mathcal{V})$ *is said to be a sigma protocol for the relation* R_L *when it uses the following pattern:* $\mathcal{P}$ *sends a commitment* C*,* $\mathcal{V}$ *sends a challenge* b*,* $\mathcal{P}$ *sends a response* r *after which* $\mathcal{V}$ *accepts or rejects the proof, and the following requirement holds:*

- *Special soundness: There exists a polynomial-time algorithm* $\mathcal{E}$ *that given any* x *and any pair of accepting transcripts* $(t, t') = ((C, b, r), (C, b', r'))$ *for* x *such that* $b \neq b' : Pr[y \leftarrow \mathcal{E}(x, t, t') : (x, y) \in R_L]$ *is overwhelming.*

In [26], Lindell extends the definition of special soundness to proofs of knowledge that are not sigma protocols. We now recall it using the formalism introduced in [5], where t is a transcript of the protocol execution and s represents the state of $\mathcal{P}^*$ including its random tape.

Definition 5 (Statistical Witness-Extended Emulation [5]). *An interactive proof system* $(\mathcal{P}, \mathcal{V})$ *has* statistical *witness-extended emulation if for all deterministic polynomial-time* $\mathcal{P}^*$*, there exists an expected polynomial-time emulator* $\mathcal{E}$ *such that for all interactive adversaries* $\mathcal{A}$*:*

$$\mathsf{Pr}\left[(y, s) \leftarrow \mathcal{A}(1^k); t \leftarrow \langle \mathcal{P}^*(y, s), \mathcal{V}(y)\rangle; b \leftarrow \mathcal{A}(t) : b = 1\right] \approx$$
$$\mathsf{Pr}\left[\begin{array}{l}(y, s) \leftarrow \mathcal{A}(1^k); (t, x) \leftarrow \mathcal{E}^{\langle \mathcal{P}^*(y,s), \mathcal{V}(y)\rangle}(y); b \leftarrow \mathcal{A}(t) : \\ b = 1 \text{ and if } t \text{ is accepting then } (x, y) \in L\end{array}\right]$$

where the oracle called by $\mathcal{E}^{\langle \mathcal{P}^*(y,s), \mathcal{V}(y)\rangle}(y)$ *permits rewinding to a specific point and resuming with fresh randomness for the verifier from this point onwards.*

Definition 6 (Zero-Knowledge [30]). *An interactive proof system $(\mathcal{P}, \mathcal{V})$ is zero-knowledge if for every p.p.t interactive machine $\mathcal{V}^*$ there exists a probabilistic expected polynomial-time algorithm $\mathcal{S}$ (called simulator) such that the following two ensembles are computationally indistinguishable (when the distinguishing gap is a function in $|x|$) : $\{\langle \mathcal{P}(y), \mathcal{V}^*(z)\rangle(x)\}_{z\in\{0,1\}^*, x\in L}$ for an arbitrary $y \in R_L(x)$ and $\{\mathcal{S}(x,z)\}_{z\in\{0,1\}^*, x\in L}$. That is, for every probabilistic algorithm $\mathcal{D}$ running in time polynomial in the length of its first input, every polynomial p, all $(x,y) \in R_L$ and all auxiliary inputs $z, z' \in \{0,1\}^*$ it holds that: $|Pr[\mathcal{D}(x, z', \langle \mathcal{P}(y), \mathcal{V}^*(z)\rangle(x)) = 1] - Pr[D(x, z', \mathcal{S}(x, z)) = 1]| < p(|x|)^{-1}$.*

The term "perfect zero-knowledge" refers to proof systems where the two ensembles are identically distributed. Furthermore, a weaker variant called *Honest Verifier Zero-Knowledge* (HVZK) is usually considered as well. Only a single verifier $\mathcal{V} = \mathcal{V}^*$ that always follows the protocol is assumed in this variant.

Definition 7 (Commitment Scheme). *A non-interactive commitment scheme $\Gamma = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ on a message space $\mathcal{M}$ is a tuple that verifies:*

1. $\mathsf{ck} \leftarrow \mathsf{Setup}(1^k)$ *generates a commitment key* ck.
2. $\forall\ m \in \mathcal{M} : (c, d) \leftarrow \mathsf{Commit}_{\mathsf{ck}}(m)$ *is the commitment/opening pair for* m.
3. *A commitment can be opened to* $m' \in \mathcal{M} \cup \perp$ *with* $m' \leftarrow \mathsf{Open}(c, d)$, *where* $\perp$ *is returned if* c *is not a valid commitment to any message.*
4. $\mathsf{Correctness} : \forall\ m \in \mathcal{M} : \mathsf{Open}(\mathsf{Commit}_{\mathsf{ck}}(m)) = m$.

Commitment schemes are required to have two security properties: binding and hiding. Binding states that it should be infeasible for any party to come up with an opening that would reveal a different value than the one initially committed. Hiding states that it should be infeasible for any party to reveal a commitment without the corresponding opening. If a scheme is perfectly binding, it can only be computationally hiding or the other way round.

Definition 8 (Hiding and Binding). *A commitment scheme Γ has the hiding security property if the advantage of any p.p.t algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ defined by $\boldsymbol{Adv}^{Hiding}_{\Gamma,\mathcal{A}}(k) := 2 \cdot Pr\Big[\boldsymbol{Exp}^{Hiding}_{\Gamma,\mathcal{A}}(k) \Rightarrow \mathsf{true}\Big] - 1$ is negligible, where $\boldsymbol{Exp}^{Hiding}_{\Gamma,\mathcal{A}}(k)$ is the experiment shown in Figure 1 (left side).*
A commitment scheme Γ has the binding security property if the advantage of any p.p.t algorithm $\mathcal{A}$ defined by $\boldsymbol{Adv}^{Binding}_{\Gamma,\mathcal{A}}(k) := Pr\Big[\boldsymbol{Exp}^{Binding}_{\Gamma,\mathcal{A}}(k) \Rightarrow \mathsf{true}\Big] - 1$ is negligible, where $\boldsymbol{Exp}^{Binding}_{\Gamma,\mathcal{A}}(k)$ is the experiment shown in Figure 1 (right side).

3 Generic Randomizable Encryption

In this section we propose several definitions to characterize the kinds of randomizations that PKE schemes can support.

To begin with, we present a definition of *re-randomizability* [32], which has also been introduced under different names or variants ([9,24,20,17]). Unlike previous work, we include the notion of *derandomizability*, and omit the distinction

Experiment $\mathbf{Exp}^{\text{Hiding}}_{\Gamma,\mathcal{A}}(k)$
$\mathsf{ck} \xleftarrow{\$} \mathsf{Setup}(1^k)$
$(\mathsf{m}_0, \mathsf{m}_1, \mathsf{state}) \leftarrow \mathcal{A}_1(\mathsf{ck})$
$b \xleftarrow{\$} \{0,1\}; (\mathsf{c}, \mathsf{d}) \leftarrow \mathsf{Commit}_{\mathsf{ck}}(\mathsf{m}_b)$
$b' \leftarrow \mathcal{A}_2(\mathsf{c}, \mathsf{state})$
return $b = b'$

Experiment $\mathbf{Exp}^{\text{Binding}}_{\Gamma,\mathcal{A}}(k)$
$\mathsf{ck} \xleftarrow{\$} \mathsf{Setup}(1^k)$
$(\mathsf{c}, \mathsf{d}, \mathsf{d}') \leftarrow \mathcal{A}_1(\mathsf{ck})$
$\mathsf{m} \leftarrow \mathsf{Open}(c, d); \mathsf{m}' \leftarrow \mathsf{Open}(c, d')$
if ($\mathsf{m} = \perp$ or $\mathsf{m}' = \perp$) **then return** 0
else return $m \neq m'$

Fig. 1: Experiments defining Hiding and Binding respectively.

with universal re-randomizability from [32], which we consider implicit unless otherwise said. Informally speaking, a scheme that is randomizable and derandomizable supports not only the generation of fresh ciphertexts but also the "rollback" process. Furthermore, we will say that a scheme achieves perfect randomizability when no adversary can distinguish between a fresh encryption of the original message and a randomization of the ciphertext.

Definition 9 (Randomizable PKE scheme (Rand-PKE) [32]). *Given a PKE scheme* $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$*, we say that* Π *is* randomizable *if there exists a polynomial-time algorithm* Rand *such that:*

1. Rand *takes* $c \in \mathcal{C}$*,* $r \in \mathcal{R}$ *and returns* $c' \in \mathcal{C}$*.*
2. $\forall\ (\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}(1^k)$*,* $r \in \mathcal{R}$ *and* $c \in \mathcal{C}$*:* $\mathsf{Pr}\left[\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Rand}(c, r)) = \mathsf{Dec}_{\mathsf{sk}}(c)\right] = 1$*.*

Moreover, we say that Π *is* derandomizable *if for any* $c \in \mathcal{C}$ *and* $r \in \mathcal{R}$*, there exists an efficiently computable* r^* *such that:* $\mathsf{Pr}\left[c = \mathsf{Rand}(\mathsf{Rand}(c, r), r^*)\right] = 1$*.*

Definition 10 (Computational and perfect randomizability [32]). *We say that a* Rand-PKE *scheme is* computationally *randomizable if for any* k*,* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^k)$*,* $m \in \mathcal{M}$*,* $r \in \mathcal{R}$*,* $c = \mathsf{Enc}_{\mathsf{pk}}(m; r)$ *and any polynomial-time distinguisher* $\mathcal{D}$*, there exists a negligible function* $\epsilon(\cdot)$ *such that:*

$$\left| \mathsf{Pr}\begin{bmatrix} r' \xleftarrow{\$} \mathcal{R}; \\ c' \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m; r'); \\ b \leftarrow \mathcal{D}(\mathsf{pk}, c, c'); \end{bmatrix} : b = 1 \right] - \mathsf{Pr}\begin{bmatrix} r' \xleftarrow{\$} \mathcal{R}; \\ c' \leftarrow \mathsf{Rand}(c, r'); \\ b \leftarrow \mathcal{D}(\mathsf{pk}, c, c'); \end{bmatrix} : b = 1 \right] \Bigg| \leq \epsilon(k).$$

We say that the scheme is perfectly *randomizable when* $\epsilon(k) = 0$*.*

We now introduce the following definitions that specify how the random coins used to produce fresh encryptions and randomizations can relate together. We will say that a PKE scheme is strongly randomizable when it also supports efficient algorithms to compute such relationship.

Definition 11 (Strong Randomizable PKE scheme). *Given a PKE scheme* $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$*, we say that* Π *is* strongly randomizable *if it is a* Rand-PKE *and there exist a polynomial-time algorithm* RandR *such that:*

1. RandR *takes* r *and* $r' \in \mathcal{R}$ *and returns* $r'' \in \mathcal{R}$*.*
2. $\forall\ (\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}(1^k)$*,* $m \in \mathcal{M}$ *and* $r'' \leftarrow \mathsf{RandR}(r, r')$*, the following equation holds:* $\mathsf{Rand}(\mathsf{Enc}_{\mathsf{pk}}(m; r), r') = \mathsf{Enc}_{\mathsf{pk}}(m; r'')$*.*

Moreover, we say that Π *is* random-extractable *if there exists a a polynomial-time algorithm* RandExt *such that for any* $(r, r', r'') \in \mathcal{R}^3$*:*

$\mathsf{Pr}\left[r = \mathsf{RandExt}(r', r'') : r'' \leftarrow \mathsf{RandR}(r, r')\right] = 1$*.*

Definition 12 (Computational and perfect strong randomizability). *We say that a* Rand-PKE *scheme is* computationally *strongly randomizable if for any* k, $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KGen}(1^k)$, $m \in \mathcal{M}$, $r \in \mathcal{R}$, $c = \mathsf{Enc}_{\mathsf{pk}}(m;r)$ *and any polynomial-time distinguisher* $\mathcal{D}$, *there exists a negligible function* $\epsilon(\cdot)$ *such that:*

$$\left| \mathsf{Pr}\begin{bmatrix} r' \xleftarrow{\$} \mathcal{R}; \\ c' \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m;r'); \\ b \leftarrow \mathcal{D}(\mathsf{pk},r,c,r',c'); \end{bmatrix} : b = 1 \right] - \mathsf{Pr}\begin{bmatrix} r'' \xleftarrow{\$} \mathcal{R}; \\ r' \leftarrow \mathsf{RandR}(r,r''); \\ c' \leftarrow \mathsf{Rand}(c,r''); \\ b \leftarrow \mathcal{D}(\mathsf{pk},r,c,r',c'); \end{bmatrix} : b = 1 \right] \right| \leq \epsilon(k).$$

We say that the scheme is perfectly *strongly randomizable when* $\epsilon(k) = 0$.

We now define the notions of *message-randomizability* and *key-randomizability*. For message-randomizability, we consider three different algorithms. The first one computes the plaintext's randomization, the second compute it on the ciphertext, and the third one computes the randomness given two messages.

Definition 13 (Message-randomizable PKE scheme (MsgRand-PKE)). *Given a PKE scheme* $\Pi = (\mathsf{KGen},\mathsf{Enc},\mathsf{Dec})$, *we say that* Π *is* message-randomizable *if there exists a set* $\mathcal{R}_{\mathsf{M}}$ *and two polynomial-time algorithms* MsgRandM *and* MsgRandC *such that:*

1. MsgRandM *takes* $m \in \mathcal{M}$, $r \in \mathcal{R}_{\mathsf{M}}$ *and returns* $m' \in \mathcal{M}$. *Moreover, the function* $f_r : \mathcal{M} \Rightarrow \mathcal{M}$ *defined by* $f_r(m) = \mathsf{MsgRandM}(m,r)$, *is bijective.*
2. MsgRandC *takes* $c \in \mathcal{C}$, $r \in \mathcal{R}_{\mathsf{M}}$ *and returns* $c' \in \mathcal{C}$.
3. $\forall\ (\mathsf{pk},\mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}(1^k)$, $m \in \mathcal{M}$, $r' \in \mathcal{R}$, $r \in \mathcal{R}_{\mathsf{M}}$ *and* $c = \mathsf{Enc}(m;r')$: $\mathsf{Pr}\left[\mathsf{Dec}_{\mathsf{sk}}(\mathsf{MsgRandC}(c,r)) = \mathsf{MsgRandM}(m,r)\right] = 1$.

Moreover, we say that Π *is* message-derandomizable *if for any* $m \in \mathcal{M}$ *and* $r \in \mathcal{R}_{\mathsf{M}}$, *there exists a unique efficiently computable* r^* *such that:*

$\mathsf{Pr}\left[m = \mathsf{MsgRandM}(\mathsf{MsgRandM}(m,r),r^*)\right] = 1$.

Finally, we say that Π *is* message-random-extractable *if there exists a p.p.t algorithm* MsgRandExt *such that for any* $m \in \mathcal{M}$ *and* $r \in \mathcal{R}_{\mathsf{M}}$*:*

$\mathsf{Pr}\left[r = \mathsf{MsgRandExt}(m,\mathsf{MsgRandM}(m,r))\right] = 1$.

Note that we require MsgRandM to be bijective. This property is implicity required for the message-derandomizability. Indeed, if a randomized message can be obtained using different messages but the same random, then it could also be derandomized in several ways, which contradicts our definition.

Definition 14 (Computational and perfect message-randomizability). *We say that a* MsgRand-PKE *scheme is* computationally *message-randomizable if for any* k, $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KGen}(1^k)$, $m \in \mathcal{M}$, $r \in \mathcal{R}$, $c = \mathsf{Enc}_{\mathsf{pk}}(m;r)$ *and any polynomial-time distinguisher* $\mathcal{D}$, *there exists a negligible function* $\epsilon(\cdot)$ *such that:*

$$\left| \mathsf{Pr}\begin{bmatrix} m' \xleftarrow{\$} \mathcal{M}; \\ c' \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m';r); \\ b \leftarrow \mathcal{D}(\mathsf{pk},c,c'); \end{bmatrix} : b = 1 \right] - \mathsf{Pr}\begin{bmatrix} \mathsf{r_m} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}}; \\ c' \leftarrow \mathsf{MsgRandC}(c,\mathsf{r_m}); \\ b \leftarrow \mathcal{D}(\mathsf{pk},c,c'); \end{bmatrix} : b = 1 \right] \right| \leq \epsilon(k).$$

We say that the scheme is perfectly *message-randomizable when* $\epsilon(k) = 0$.

For key-randomizability, we consider three algorithms as well. The first one randomizes the public key, the second one the secret key, and the third one randomizes a ciphertext given a randomized public key.

						Perfect ZK		ZKPoK		
Scheme	Security	RCD	Rand	MsgRand	KeyRand	Π_{PEQ}	Π_{PINEQ}	Π_{MATCHPEQ}	Π_{SIGPEQ}	Π_{RSPEQ}
ElGamal [16]	IND-CPA	✓	✓	✓	✓	✓	✓	✓	✓	✓
Paillier [28]	IND-CPA	✓	✓	✓		✓	✓	✓		✓
GM [19]	IND-CPA		✓	✓		✓	✓	✓		
DEG [13]	IND-CCA1	✓	✓	✓	✓	✓	✓	✓	✓	✓
CS-lite [12]	IND-CCA1	✓	✓	✓		✓	✓			✓
DSCS [32]	RCCA	✓	✓				✓			

Table 1: PKE schemes and their properties.

Definition 15 (Key-randomizable PKE scheme (KeyRand-PKE)). *Given a PKE scheme* $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$*, we say that* Π *is* key-randomizable *if there exists a set* $\mathcal{R}_{\mathsf{K}}$ *and three polynomial-time algorithms such that:*

1. $\mathsf{KeyRandPk}$ *takes a public key* pk*,* $r \in \mathcal{R}_{\mathsf{K}}$ *and returns* pk'*.*
2. $\mathsf{KeyRandSk}$ *takes a secret key* sk*,* $r \in \mathcal{R}_{\mathsf{K}}$ *and returns* sk'*.*
3. $\mathsf{KeyRandC}$ *takes* $c \in \mathcal{C}$*,* $r \in \mathcal{R}_{\mathsf{K}}$ *and returns* $c' \in \mathcal{C}$*.*
4. $\forall\ (\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}(1^k)$*,* $m \in \mathcal{M}$*,* $r \in \mathcal{R}$*,* $r_{\mathsf{k}} \in \mathcal{R}_{\mathsf{K}}$ *and* $c = \mathsf{Enc}(m; r)$*:*

$$\Pr\left[\begin{array}{l}(\mathsf{Dec}_{\mathsf{sk}}(c) = \mathsf{Dec}_{\mathsf{KeyRandSk}(\mathsf{sk},r_{\mathsf{k}})}(\mathsf{KeyRandC}(c, r_{\mathsf{k}})))\\ \wedge(\mathsf{Dec}_{\mathsf{KeyRandSk}(\mathsf{sk},r_{\mathsf{k}})}(\mathsf{Enc}_{\mathsf{KeyRandPk}(\mathsf{pk},r_{\mathsf{k}})}(m; r)) = m)\end{array}\right] = 1.$$

Moreover, we say that Π *is* key-derandomizable *if for any secret key* sk *and* $r_{\mathsf{k}} \in \mathcal{R}_{\mathsf{K}}$*, there exists an efficiently computable* r^*_{k} *such that:*

$$\Pr\left[\mathsf{sk} = \mathsf{KeyRandSk}(\mathsf{KeyRandSk}(\mathsf{sk}, r_{\mathsf{k}}), r^*_{\mathsf{k}})\right] = 1.$$

Definition 16 (Computational and perfect key-randomizability). *We say that a* $\mathsf{KeyRand\text{-}PKE}$ *scheme is* computationally *key-randomizable if for any* k*,* $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}(1^k)$*,* $m \in \mathcal{M}$*,* $r \in \mathcal{R}$*,* $c = \mathsf{Enc}_{\mathsf{pk}}(m; r)$ *and any polynomial-time distinguisher* $\mathcal{D}$*, there exists a negligible function* $\epsilon(\cdot)$ *such that:*

$$\left|\Pr\left[\begin{array}{l}(\mathsf{pk}', \mathsf{sk}') \xleftarrow{\$} \mathsf{KGen}(1^k);\\ c' \leftarrow \mathsf{Enc}_{\mathsf{pk}'}(m; r);\\ b \leftarrow \mathcal{D}(\mathsf{sk}, \mathsf{pk}, c', \mathsf{sk}', \mathsf{pk}');\end{array} : b = 1\right] - \Pr\left[\begin{array}{l}r_{\mathsf{k}} \xleftarrow{\$} \mathcal{R}_{\mathsf{K}};\\ \mathsf{pk}' = \mathsf{KeyRandPk}(\mathsf{pk}, r_{\mathsf{k}});\\ \mathsf{sk}' = \mathsf{KeyRandPk}(\mathsf{sk}, r_{\mathsf{k}});\\ c' = \mathsf{KeyRandC}(c, r_{\mathsf{k}});\\ b \leftarrow \mathcal{D}(\mathsf{sk}, \mathsf{pk}, c', \mathsf{sk}', \mathsf{pk}');\end{array} : b = 1\right]\right| \leq \epsilon(k).$$

We say that the scheme is perfectly *key-randomizable when* $\epsilon(k) = 0$*.*

In Table 1, we list some well-known PKE schemes and their relationship with our definitions and protocols. We stress that although fully homomorphic schemes such as those based on lattices could also be used to instantiate our protocols, partial homomorphic properties presented in the scheme can be used to implement the different algorithms as well. Nonetheless, as shown with the DSCS scheme [32], we note that being partially homomorphic is not necessary to achieve re-randomizability. We refer the reader to Appendix A for examples of how our protocols can be instantiated with ElGamal and Paillier.

Prover $\mathsf{P}(\mathsf{sk},\mathsf{pk},c_0,c_1)$		**Verifier** $\mathsf{V}(\mathsf{pk},c_0,c_1)$
		if $(\mathsf{pk},c_0,c_1)\notin\mathcal{K}\times\mathcal{C}^2$ **then** Abort
		$r\xleftarrow{\$}\mathcal{R}$; $b\xleftarrow{\$}\{0,1\}$
if $\mathsf{Dec}_{\mathsf{sk}}(c'_b)=\mathsf{Dec}_{\mathsf{sk}}(c_0)$	$\xleftarrow{c'_b}$	$c'_b\leftarrow\mathsf{Rand}(c_b,r)$
then $z=0$ **else** $z=1$	$\xrightarrow{z}$	**if** $(z=b)$ **then** Accept **else** Reject

Fig. 2: One round of the protocol Π_{HPINEQ} (repeated k times).

4 Interactive Protocols

This section presents protocols for proving plaintext equality and inequality where the common input consists of a public key and two ciphertexts generated with it. As private input, the prover will have the corresponding private key. For plaintext inequality protocols we will require the scheme to be randomizable whereas for plaintext equality we will also require it to be message-randomizable.

In both cases, we first introduce an HVZK variant, which we then modify to achieve zero-knowledge in the presence of malicious verifiers. Complete security proofs for all protocols in this work are given in Appendix B.

4.1 Plaintext Inequality

Let us first introduce our protocol Π_{HPINEQ} (Figure 2). It starts with the verifier randomly choosing $r\xleftarrow{\$}\mathcal{R}$ and $b\in\{0,1\}$. Then it computes $c'_b\leftarrow\mathsf{Rand}(c_b,r)$ and sends c'_b to the prover. At this stage, the prover receives a ciphertext that cannot link without decryption to c_0 or c_1. Since the verifier is honest, the prover either decrypts c'_b to m_0 or m_1 and can determine b and send it back to the verifier. The verifier accepts if and only if it receives b as expected.

Theorem 1. *Let Π be a PKE scheme, which is (computationally) randomizable. If Π is used in Π_{HPINEQ}, then Π_{HPINEQ} is complete, computationally sound and perfect HVZK.*

The idea of this protocol can easily be explained to a large audiance replacing the ciphertexts with closed boxes using a padlock. Consider two identical closed boxes that contain a white card and a black card respectively. The prover has a key that allows him to open both boxes, and wants to prove the verifier that the boxes contains different things without revealing anything else. The verifier secretly chooses one of the two boxes and challenges the prover to guess which box he picked. The prover secretly opens the box and deduces which one it was from the color of the card. He then tells the verifier which was the box and if the verifier agrees, they repeat the protocol k times. If the two identical boxes contain the same card, then the prover has no information about the box he receives and fails one of the rounds with probability $1/2^k$.

Protocol Π_{PINEQ} (Figure 3), is an amendment of Π_{HPINEQ} that uses a commitment scheme. Without it, a malicious verifier could send a ciphertext that is not a

Prover $\mathsf{P}(\mathsf{sk},\mathsf{pk},c_0,c_1)$		**Verifier** $\mathsf{V}(\mathsf{pk},c_0,c_1)$
		if $(\mathsf{pk},c_0,c_1)\notin\mathcal{K}\times\mathcal{C}^2$ **then** Abort
$z=\neg(\mathsf{Dec}_{\mathsf{sk}}(c_0)=\mathsf{Dec}_{\mathsf{sk}}(c'_b))$	$\xleftarrow{c'_b}$	$r\xleftarrow{\$}\mathcal{R};b\xleftarrow{\$}\{0,1\};c'_b\leftarrow\mathsf{Rand}(c_b,r)$
$(\mathsf{comm},\mathsf{op})\leftarrow\mathsf{Commit}(z)$	$\xrightarrow{\mathsf{comm}}$	
	$\xleftarrow{r,b}$	
if $c'_b\neq\mathsf{Rand}(c_b,r)$ **then** $\mathsf{op}:=\perp$	$\xrightarrow{\mathsf{op}}$	$z'\leftarrow\mathsf{Open}(\mathsf{comm},\mathsf{op})$
		if $(z'=b)$ **then** Accept **else** Reject

Fig. 3: One round of the protocol Π_{PINEQ} (repeated k times).

Prover $\mathsf{P}(\mathsf{sk},\mathsf{pk},c_0,c_1)$		**Verifier** $\mathsf{V}(\mathsf{pk},c_0,c_1)$
		if $(\mathsf{pk},c_0,c_1)\notin\mathcal{K}\times\mathcal{C}^2$ **then** Abort
		$r\xleftarrow{\$}\mathcal{R};r_{\mathsf{m}}\xleftarrow{\$}\mathcal{R}_{\mathsf{M}};\ b\xleftarrow{\$}\{0,1\}$
		$c'_b\leftarrow\mathsf{Rand}(c_b,r)$
$m'\leftarrow\mathsf{Dec}_{\mathsf{sk}}(c''_b);m\leftarrow\mathsf{Dec}_{\mathsf{sk}}(c_0)$	$\xleftarrow{c''_b}$	$c''_b\leftarrow\mathsf{MsgRandC}(c'_b,r_{\mathsf{m}})$
$z\leftarrow\mathsf{MsgRandExt}(m',m)$	$\xrightarrow{z}$	**if** $(z=r_{\mathsf{m}})$ **then** Accept **else** Reject

Fig. 4: One round of the protocol Π_{HPEQ} (repeated k times).

randomization of c_0 or c_1 and check whether it encrypts the same value. The commitment scheme protects the prover from such verifiers. To this end, the verifier first randomizes the ciphertext and then sends it to the prover, which computes z and commits to the resulting value. Then, the verifier reveals the randomness used at the first stage and the prover opens the commitment if and only if these values are consistent with the ciphertext obtained from the verifier.

Theorem 2. *Let Π be a PKE scheme, which is perfectly strong randomizable and derandomizable. Let Γ be a commitment scheme, which is computationally binding and perfectly hiding. If Π and Γ are used in the protocol Π_{PINEQ}, then Π_{PINEQ} is complete, computationally sound and perfect zero-knowledge.*

4.2 Plaintext Equality

As before, we begin explaining our protocol Π_{HPEQ} (Figure 4). First, the verifier randomly chooses $\mathsf{r}\in\mathcal{R}$, $\mathsf{r_m}\in\mathcal{R}_{\mathsf{M}}$ and $b\in\{0,1\}$. Then it computes $c'_b\leftarrow\mathsf{Rand}(c_b,\mathsf{r})$ and $c''_b\leftarrow\mathsf{MsgRandC}(c'_b,r_{\mathsf{m}})$ to send c''_b to the prover. At this stage, the prover receives a ciphertext that cannot be linked to c_0 nor to c_1. The prover decrypts c''_b obtaining a message m', which corresponds to a message-randomization of either the message decrypted by c_0 or by c_1. The prover computes $z=\mathsf{MsgRandExt}(m',m)$ and sends it to the verifier. The verifier accepts if and only if $z=r_{\mathsf{m}}$. Since both ciphertexts, c_0 and c_1, belong to $\mathsf{Enc}_{\mathsf{pk}}(m)$, the prover can always compute z correctly. If this is not the case, then a cheating prover can only correctly guess the bit b with probability at most $1/2$.

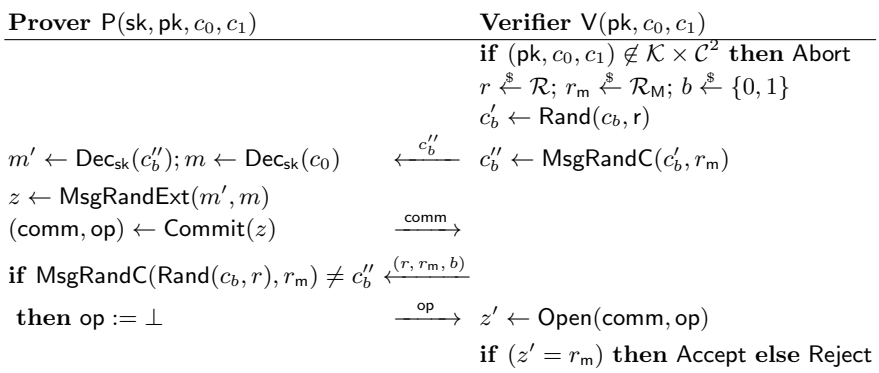

Fig. 5: One round of the protocol Π_{PEQ} (repeated k times).

Theorem 3. *Let Π be a PKE scheme, which is (computationally) randomizable, (computationally) message-randomizable and message-random-extractable. If Π is the scheme used in Π_{HPEQ}, then Π_{HPEQ} is complete, computationally sound and perfect HVZK.*

Figure 5 shows our last variant, which makes use of a commitment scheme. Without one, a malicious verifier could send a ciphertext c^* for which he knows the corresponding message m^*. Once z is received from the prover, the malicious verifier will gain information about the relation of (m, m^*, z) and could eventually compute m. By relying on the commitment scheme's hiding property, the prover first commits to the value z. Then, it checks whether the verifier has correctly randomized the messages or not to open the commitment.

Theorem 4. *Let Π be a PKE scheme, which is perfectly strong randomizable and derandomizable, perfectly message-randomizable and message-derandomizable and message-random-extractable. Let Γ be the commitment scheme, which is computationally binding and perfectly hiding. If Π and Γ are used in the protocol Π_{PEQ}, then Π_{PEQ} is complete, computationally sound and perfect zero-knowledge.*

Note that in Theorems 2 and 4 zero-knowledge can be computational if the randomization conditions are computational instead of perfect or if the commitment scheme being used has only computational hiding.

5 ZKPoK for Plaintext Equality

We switch our attention to protocols that are Zero-Knowledge Proofs of Knowledge (ZKPoK) for plaintext equality. In § 5.1 we focus on ZKPoK of the secret key whereas in § 5.2 we focus on ZKPoK of the randomness used to generate the ciphertexts. The application is not the same because if the prover knows the secret key, the use case to consider is when the prover acts as a receiver of those ciphertexts. On the other hand, if the prover knows the randomness used

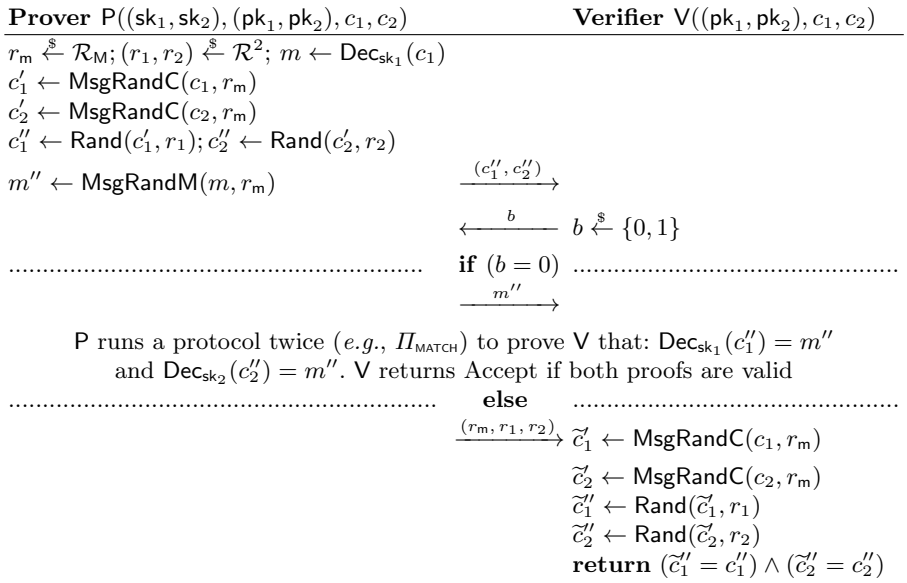

Fig. 6: One round of the protocol Π_{MATCHPEQ} (repeated k times).

to generate the ciphertexts, the use case to consider is when the prover acts as a sender. Finally, we outline how non-interactive variants can be defined in § 5.3.

5.1 Protocols based on knowledge of the secret key

Protocols in this section additionally require the scheme to be key-randomizable. We present a protocol called Π_{MATCHPEQ} (Figure 6), which relies on a ZKP to prove that the decryption of a given ciphertext matches a given message. Such proofs are known for numerous encryption schemes (*e.g.*, [13,16,19,28]). Then, we introduce an auxiliary protocol called Π_{MATCH} (Figure 7), that meets the requirement of Π_{MATCHPEQ} (its a proof system for the above mentioned). We also present here a third protocol called Π_{SIGPEQ} (Figure 8) that merges the two previous ones. It requires a randomizable, message-randomizable and key-randomizable scheme, but it does not require any other protocol as a subroutine, which makes it more efficient that Π_{MATCHPEQ} instantiated with Π_{MATCH}. An interesting additional property of Π_{MATCHPEQ} and Π_{SIGPEQ} is that both can also be used to prove plaintext equality of two ciphertexts encrypted under different keys.

In protocol Π_{MATCHPEQ} the prover sends two message-randomizations to the verifier who then challenges it on these ciphertexts. If both ciphertexts encrypt message-randomizations of the same message, then the prover can either prove that it correctly did the message-randomizations or that both ciphertexts encrypt the same message.

Prover $\mathsf{P}(\mathsf{sk},\mathsf{pk},c,m)$		**Verifier** $\mathsf{V}(\mathsf{pk},c,m)$
$r_\mathsf{k} \xleftarrow{\$} \mathcal{R}_\mathsf{K}$; $\mathsf{pk}' \leftarrow \mathsf{KeyRandPk}(\mathsf{pk}, r_\mathsf{k})$		
$\mathsf{sk}' \leftarrow \mathsf{KeyRandSk}(\mathsf{sk}, r_\mathsf{k})$; $r \xleftarrow{\$} \mathcal{R}$		
$c'' \leftarrow \mathsf{Rand}(c', r)$; $c' \leftarrow \mathsf{KeyRandC}(c, r_\mathsf{k})$	$\xrightarrow{(\mathsf{pk}', c'')}$	
	$\xleftarrow{b}$	$b \xleftarrow{\$} \{0,1\}$
if $(b = 0)$ **then** $z = \mathsf{sk}'$ **else** $z = (r, r_\mathsf{k})$	$\xrightarrow{z}$	**if** $b = 1$ **then**
		$\widetilde{\mathsf{pk}}' \leftarrow \mathsf{KeyRandPk}(\mathsf{pk}, r_\mathsf{k})$
		$\widetilde{c}' \leftarrow \mathsf{KeyRandC}(c, r_\mathsf{k})$
		$\widetilde{c}'' \leftarrow \mathsf{Rand}(\widetilde{c}', r)$
		return $(\widetilde{c}'' = c'') \wedge (\widetilde{\mathsf{pk}}' = \mathsf{pk}')$
		else return $(m = \mathsf{Dec}_{sk'}(c''))$

Fig. 7: One round of the protocol Π_{MATCH} (repeated k times).

Theorem 5. *Let Π be the PKE scheme used in Π_{MATCHPEQ}. If Π is perfectly randomizable and derandomizable, perfectly message-randomizable and message-derandomizable, and if the proof in step three is instantiated by a sigma protocol that is correct, special sound, and perfectly zero-knowledge, then Π_{MATCHPEQ} is complete, has statistical witness-extended emulation, and perfect zero-knowledge.*

For Π_{MATCH}, we consider a setting in which the verifier has access to the pk, the ciphertext c, the message m and challenges the prover to prove that c is an encryption of m. This protocol's intuition is that if the scheme is randomizable and key-randomizable, the prover can generate a new ciphertext for the same massage but under different keys. The verifier is then allowed to check that 1) the prover can generate a new ciphertext c'' which decrypts to the same message and 2) by decrypting c'' to m conclude that the original ciphertext c is also an encryption of m.

Theorem 6. *Let Π be the PKE scheme used in Π_{MATCH}. If Π is perfectly randomizable, perfectly key-randomizable and key-derandomizable, then Π_{MATCH} is complete, special sound, and perfect zero-knowledge.*

To conclude this section, we present the protocol Π_{SIGPEQ}, a sigma protocol for plaintext equality of two ciphertexts built upon the previous ones. In this protocol, the prover performs a message-randomization on the ciphertexts and a key-randomization to obtain new ciphertexts. These ciphertexts decrypt to the same message m' but under a different key. Once the prover sends the public keys and the new ciphertexts to the verifier, the verifier challenges the prover. The intuition behind the challenge is that if the two ciphertexts obtained by the verifier are message-randomizations of the same message, then the prover should be able to provide either the corresponding secret key to confirm it or the randomness used to verify the procedure. This protocol is more efficient

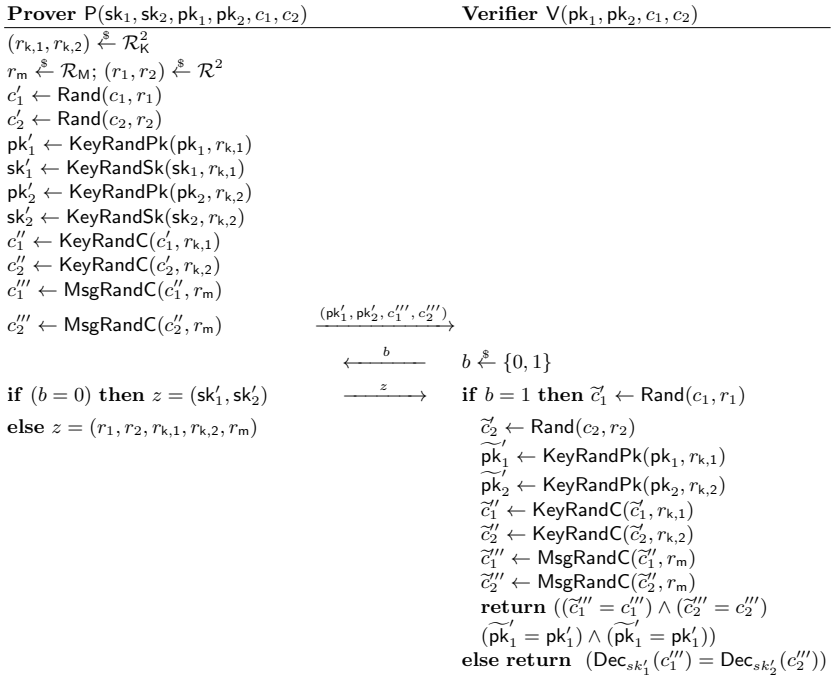

Fig. 8: One round of the protocol Π_{SIGPEQ} (repeated k times).

because it requires exactly k rounds, while Π_{MATCHPEQ} requires k rounds times the number of rounds of Π_{MATCH}.

Theorem 7. *Let Π be the PKE scheme used in Π_{SIGPEQ}. If Π is perfectly randomizable, perfectly message-randomizable and perfectly key-randomizable, then Π_{SIGPEQ} is complete, special sound, and perfect zero-knowledge.*

5.2 Protocols based on knowledge of the encryption randomness

Based on the previous ideas, we present in this section the protocol Π_{RSPEQ}, which requires the PKE scheme to be random coin decryptable, strong randomizable and message-randomizable. The intuition behind this protocol (Figure 9) is the same as in Π_{SIGPEQ}; the verifier challenges the prover to either provide the randomizers or to allow it to check the procedure.

Theorem 8. *Let Π be the PKE scheme used in Π_{RSPEQ}. If Π is perfectly strong randomizable, random-extractable, perfectly message-randomizable and random coin decryptable, then Π_{RSPEQ} is complete, special sound, and perfect zero-knowledge.*

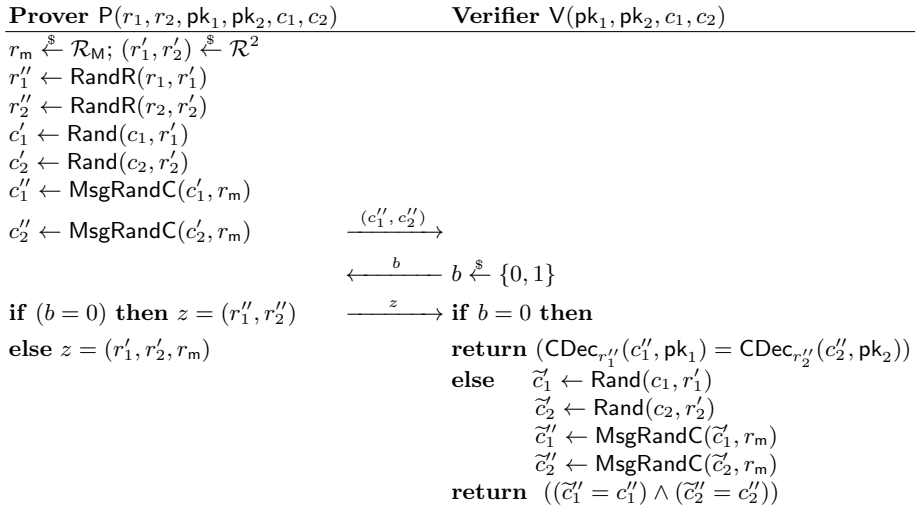

Fig. 9: One round of the protocol Π_{RSPEQ} (repeated k times).

5.3 Non-interactive variants

Considering that sigma protocols are invariant under parallel composition, for protocols Π_{SIGPEQ} and Π_{RSPEQ}, one can apply the strong Fiat-Shamir transformation [2] and obtain a Non-Interactive Zero-Knowledge Proof, which is secure in the random oracle model. In other words, the prover should generate k commitments $(r_1, ..., r_k)$, calculate $c \in \{0,1\}^k \leftarrow \mathcal{H}(r_1||r_2...r_k||\text{public\_parameters})$, and finally compute the responses z_i for all r_i using the i-th bit of c. This way, the soundness error $(1/2)$ is amplified to $1/2^k$.

6 Efficiency

In order to compare the efficiency of our protocols with custom ones, we provide here an efficiency analysis and implementation details using ElGamal.

Comparison with custom variants. Our generic ptotocols Π_{PEQ}, Π_{RSPEQ}, and Π_{PINEQ} are perfect zero-knowledge and do not rely on the random oracle model. We compare the efficiency of our protocols with the best (as far as we know) custom protocols for ElGamal that achieve the same security properties. Note that more efficient protocols exist under weaker hypothesis: HVZK proofs can be done using Shnorr-like protocols [36], non-interactive protocols can be done in the random oracle model replacing the challenge by the hash of the commitment, and non-interactive but computationaly zero-knowlege proofs can be done using the Groth-Sahai construction from pairings [21].

Proving the equality of two ElGamal plaintexts (m_1, m_2) given two ciphertexts $c_1 = \mathsf{Enc}_\mathsf{pk}(m_1; r_1) = (g^{r_1}, \mathsf{pk}^{r_1} m_1)$ and $c_2 = \mathsf{Enc}_\mathsf{pk}(m_2; r_2) = (g^{r_2}, \mathsf{pk}^{r_2} m_2)$ is equivalent to prove that $(g^\alpha, g^{r_1 - r_2}, g^{\alpha(r_1 - r_2)})$ is a Diffie-Hellman tuple, which

	Equality proofs			Inequality proofs	
	Using [10]	Π_{PEQ}	Π_{RSPEQ}	Using [7]	Π_{PINEQ}
Prover	2	6	4	6	6
Verifier	2	4	4	4	4
Rounds	3	4	3	3	4

Table 2: Number of exp. and rounds for plaintext equality/inequality proofs.

Protocol	Π_{HPEQ}	Π_{PEQ}	Π_{HPINEQ}	Π_{PINEQ}	Π_{RSPEQ}	Π_{SIGPEQ}
Avg. time	27.47	70.31	26.13	68.75	62.12	112.98
Deviation	0.21	1.28	0.15	0.6	2.06	3.70

Table 3: Running times in ms for different protocols using ElGamal.

can be efficiently done with the Chaum-Pedersen protocol [10] (using either the secret key or the randomness as the witness). Similarly, proving the inequality of the two plaintexts is equivalent to prove that $(g^{\alpha}, g^{r_1-r_2}, g^{\alpha(r_1-r_2)}m_1/m_2)$ is not a Diffie-Hellman tuple, which can be efficiently done with the Camenisch-Shoup protocol [7]. These protocols must be repeated k times for a security parameter k, like ours. Table 2 gives the number of exponentiations (the dominant operation in all the considered protocols) and rounds for a single run of each protocol. This comparison suggests that our generic protocols' cost is reasonable for perfect zero-knowledge protocols in the standard model.

Implementation. We implemented the protocols Π_{HPEQ}, Π_{PEQ}, Π_{HPINEQ}, Π_{PINEQ}, Π_{RSPEQ} and Π_{SIGPEQ} in Rust using the dalek library [1]. Although the implementation was done for academic purposes and simulating the interaction between a prover and a verifier (it is not production-ready), it serves to demonstrate the practicality of our protocols. More in detail, we show on Table 3 the average running times using a regular laptop (Macbook Pro from 2015) with no extra optimizations and considering a security parameter of 128. Therefore, the times shown consist of 128 *repetitions* for each protocol run so to achieve the desired soundness error. This information was gathered using the external crate *bencher*.

7 Conclusion

We characterized malleability in terms of randomizability, message-randomizability and key-randomizability for public-key encryption. Based on those notions, we defined and presented interactive and non-interactive zero-knowledge protocols for plaintext equality and inequality. As a result, we obtained generic protocols that can be instantiated with different encryption schemes. We provided examples of PKE schemes, which have different properties and that are secure under different security models to support the claim. As future work, we first want to design non-interactive protocols for plaintext inequality. We also would like to propose protocols that do not require k rounds from a generic encryption

scheme. Another idea is to construct generic "plaintext inequality test" to prove that the ciphertext's plaintext is smaller or greater than another plaintext.

Acknowledgements We would like to thank Travis Mayberry for his useful suggestions and comments to improve this work. Olivier Blazy was supported by the French ANR Project IDFIX (ANR-16-CE39-004). The European Commission partially supported Octavio Perez Kempner's work as part of the CUREX project (H2020-SC1-FA-DTS-2018-1 under grant agreement No 826404).

References

1. Protocols' implementation in Rust, `https://github.com/oblazy/proofofeq`
2. Bernhard, D., Pereira, O., Warinschi, B.: How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios. In: Wang, X., Sako, K. (eds.) Advances in Cryptology – ASIACRYPT 2012. pp. 626–643. Springer Berlin Heidelberg (2012)
3. Blazy, O., Chevalier, C., Vergnaud, D.: Non-Interactive Zero-Knowledge Proofs of Non-Membership. In: Nyberg, K. (ed.) Topics in Cryptology — CT-RSA 2015. pp. 145–164. Springer International Publishing, Cham (2015)
4. Blazy, O., Derler, D., Slamanig, D., Spreitzer, R.: Non-Interactive Plaintext In-Equality Proofs and Group Signatures with Verifiable Controllable Linkability. In: Proceedings of the RSA Conference on Topics in Cryptology - CT-RSA 2016 - Volume 9610. pp. 127–143. Springer-Verlag (2016)
5. Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting. In: Fischlin, M., Coron, J.S. (eds.) Advances in Cryptology – EUROCRYPT 2016. pp. 327–357. Springer Berlin Heidelberg (2016)
6. Bultel, X., Lafourcade, P.: A Posteriori Openable Public Key Encryption. In: Hoepman, J.H., Katzenbeisser, S. (eds.) ICT Systems Security and Privacy Protection. pp. 17–31. Springer International Publishing, Cham (2016)
7. Camenisch, J., Shoup, V.: Practical Verifiable Encryption and Decryption of Discrete Logarithms. In: CRYPTO 2003. Springer (2003)
8. Canard, S., Fuchsbauer, G., Gouget, A., Laguillaumie, F.: Plaintext-Checkable Encryption. In: Dunkelman, O. (ed.) Topics in Cryptology – CT-RSA 2012. pp. 332–348. Springer Berlin Heidelberg (2012)
9. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing Chosen-Ciphertext Security. In: Boneh, D. (ed.) Advances in Cryptology - CRYPTO 2003. pp. 565–582. Springer Berlin Heidelberg (2003)
10. Chaum, D., Pedersen, T.P.: Wallet Databases with Observers. In: Brickell, E.F. (ed.) Advances in Cryptology — CRYPTO' 92. pp. 89–105. Springer Berlin Heidelberg (1993)
11. Choi, S.G., Elbaz, A., Juels, A., Malkin, T., Yung, M.: Two-Party Computing with Encrypted Data. In: Proceedings of the Advances in Crypotology 13th International Conference on Theory and Application of Cryptology and Information Security. pp. 298–314. ASIACRYPT '07, Springer-Verlag (2007)
12. Cramer, R., Shoup, V.: A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In: Krawczyk, H. (ed.) Advances in Cryptology — CRYPTO '98. pp. 13–25. Springer Berlin Heidelberg (1998)

13. Damgård, I.: Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks. In: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology. p. 445456. CRYPTO 91, Springer-Verlag (1991)
14. Damgård, I., Jurik, M., Nielsen, J.B.: A Generalization of Paillier's Public-Key System with Applications to Electronic Voting. Int. J. Inf. Secur. **9**(6), 371385 (2010)
15. Dimitriou, T., Michalas, A.: Multi-Party Trust Computation in Decentralized Environments in the Presence of Malicious Adversaries. Ad Hoc Netw. **15**, 5366 (2014)
16. ElGamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In: Blakley, G.R., Chaum, D. (eds.) Advances in Cryptology. pp. 10–18. Springer Berlin Heidelberg (1985)
17. Faonio, A., Fiore, D., Herranz, J., Ràfols, C.: Structure-Preserving and Rerandomizable RCCA-Secure Public Key Encryption and Its Applications. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology – ASIACRYPT 2019. pp. 159–190. Springer International Publishing, Cham (2019)
18. Goldreich, O., Teichner, L.: Super-Perfect Zero-Knowledge Proofs. Springer International Publishing, Cham (2020)
19. Goldwasser, S., Micali, S.: Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. In: Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing. pp. 365–377. STOC '82, Association for Computing Machinery, New York, NY, USA (1982)
20. Golle, P., Jakobsson, M., Juels, A., Syverson, P.: Universal Re-encryption for Mixnets. In: Okamoto, T. (ed.) Topics in Cryptology – CT-RSA 2004. pp. 163–178. Springer Berlin Heidelberg (2004)
21. Groth, J., Sahai, A.: Efficient Non-interactive Proof Systems for Bilinear Groups. In: Smart, N. (ed.) Advances in Cryptology – EUROCRYPT 2008. pp. 415–432. Springer Berlin Heidelberg (2008)
22. Hasan, O., Brunie, L., Bertino, E., Shang, N.: A Decentralized Privacy Preserving Reputation Protocol for the Malicious Adversarial Model. IEEE Transactions on Information Forensics and Security **8**(6), 949–962 (2013)
23. Hazay, C., Lindell, Y.: Efficient Secure Two-Party Protocols: Techniques and Constructions. Springer-Verlag, 1st edn. (2010)
24. Hirt, M., Sako, K.: Efficient Receipt-Free Voting Based on Homomorphic Encryption. In: Proceedings of the 19th International Conference on Theory and Application of Cryptographic Techniques. pp. 539–556. EUROCRYPT '00, Springer-Verlag (2000)
25. Jakobsson, M., Juels, A.: Mix and Match: Secure Function Evaluation via Ciphertexts. In: Okamoto, T. (ed.) Advances in Cryptology — ASIACRYPT 2000. pp. 162–177. Springer Berlin Heidelberg (2000)
26. Lindell, Y.: Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation. In: Kilian, J. (ed.) Advances in Cryptology — CRYPTO 2001. pp. 171–189. Springer Berlin Heidelberg (2001)
27. McMurtry, E., Pereira, O., Teague, V.: When Is a Test Not a Proof? In: Chen, L., Li, N., Liang, K., Schneider, S. (eds.) Computer Security – ESORICS 2020. pp. 23–41. Springer International Publishing, Cham (2020)
28. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) Advances in Cryptology — EUROCRYPT '99. pp. 223–238. Springer Berlin Heidelberg (1999)
29. Parkes, D., Rabin, M., Shieber, S., Thorpe, C.: Practical secrecy-preserving, verifiably correct and trustworthy auctions. In: ICEC '06 (2006)

30. Pass, R.: Alternative Variants of Zero-Knowledge Proofs. Tech. rep., KTH Royal Institute of Technology (2004)
31. Pass, R., Shelat, A.: A course in Cryptography (2010), http://www.cs.cornell.edu/courses/cs4830/2010fa/lecnotes.pdf
32. Prabhakaran, M., Rosulek, M.: Rerandomizable RCCA Encryption. In: Menezes, A. (ed.) Advances in Cryptology - CRYPTO 2007. pp. 517–534. Springer Berlin Heidelberg (2007)
33. Reinert, M.: Cryptographic Techniques for Privacy and Access Control in Cloud-Based Applications. Ph.D. thesis, Saarland University, Saarbrücken, Germany (2018)
34. Ryan, P.Y.A.: Prêt à Voter with Paillier Encryption. Math. Comput. Model. **48**(910), 16461662 (2008)
35. Ryan, P.Y.A., Schneider, S.A.: Prêt à Voter with Re-encryption Mixes. In: Gollmann, D., Meier, J., Sabelfeld, A. (eds.) Computer Security – ESORICS 2006. pp. 313–326. Springer Berlin Heidelberg (2006)
36. Schnorr, C.P.: Efficient Identification and Signatures for Smart Cards. In: Brassard, G. (ed.) Advances in Cryptology — CRYPTO' 89 Proceedings (1990)
37. Tang, Q.: Public key encryption supporting plaintext equality test and user-specified authorization. Sec. and Commun. Netw. **5**(12), 1351–1362 (2012)
38. Yang, G., Tan, C.H., Huang, Q., Wong, D.S.: Probabilistic Public Key Encryption with Equality Test. In: Pieprzyk, J. (ed.) Topics in Cryptology - CT-RSA 2010. pp. 119–131. Springer Berlin Heidelberg (2010)

A Instantiation

Based on the literature review, we found that ElGamal and Paillier were the most used schemes to implement plaintext equality/inequality proofs. For this reason, we present here examples of how to instantiate a subset of our protocols using these schemes (both summarized in Table 4).

Let us first note that PKE schemes whose set of random coins and messages are cyclic groups $(\mathbb{G}_1, *)$ and $(\mathbb{G}_2, *)$ with identity elements e_1 and e_2 and which are homomorphic for $*$ (*i.e.* $\mathsf{Enc}(m,r) * \mathsf{Enc}(m',r') = \mathsf{Enc}(m * m', r * r')$), are randomizable and message-randomizable. To randomize a ciphertext $\mathsf{Enc}(m,r)$ with r' one can compute $\mathsf{Enc}(m,r) * \mathsf{Enc}(e_1,r') = \mathsf{Enc}(m, r * r')$, and to randomize the plaintext with m' one can compute $\mathsf{Enc}(m,r) * \mathsf{Enc}(m', e_2) = \mathsf{Enc}(m * m', r)$. We show that ElGamal and Pailler verify this property. Considering two ElGamal ciphertexts $(g^{r_1}, \mathsf{pk}^{r_1} \cdot m_1)$ and $(g^{r_2}, \mathsf{pk}^{r_2} \cdot m_2)$, we define the operation $*$ as $(g^{r_1}, \mathsf{pk}^{r} \cdot m_1) * (g^{r_2}, \mathsf{pk}^{r} \cdot m_2) = (g^{r_1} \cdot g^{r_2}, \mathsf{pk}^{r_1} \cdot m_1 \cdot \mathsf{pk}^{r_2} \cdot m_2) = (g^{(r_1+r_2)}, \mathsf{pk}^{(r_1+r_2)} \cdot (m_1 \cdot m_2))$. Considering two Pailler ciphertexts $((1+n)^{m_1} \cdot r_1^n \mod n^2)$ and $((1+n)^{m_2} \cdot r_2^n \mod n^2)$, we define the operation $*$ as $((1+n)^{m_1} \cdot r_1^n \mod n^2) * ((1+n)^{m_2} \cdot r_2^n \mod n^2) = ((1+n)^{m_1} \cdot r_1^n \cdot (1+n)^{m_2} \cdot r_2^n \mod n^2) = ((1+n)^{m_1} \cdot (1+n)^{m_2} \cdot r_1^n \cdot r_2^n \mod n^2) = ((1+n)^{(m_1+m_2)} \cdot (r_1 \cdot r_2)^n \mod n^2)$. It follows that ElGamal and Pailler can instantiate the protocols Π_{PEQ}, Π_{PINEQ} and Π_{RSPEQ}.

ElGamal			
	Key Generation	Encryption	Decryption
Input:	1^λ	$m \in \mathbb{Z}_p, \mathsf{pk}$	$(c_1, c_2) \in \mathbb{Z}_p \times \mathbb{Z}_p, \mathsf{pk}, \mathsf{sk}$
Steps:	Choose a λ-bit prime p, a generator g of $\mathbb{Z}_p^*$ and set $h \xleftarrow{\$} \mathbb{Z}_p^*$	Choose $r \xleftarrow{\$} \mathbb{Z}_p^*$ and compute $(c_1, c_2) = (m \cdot \mathsf{pk}^r, g^r)$	Compute $m = \frac{c_1}{(c_2)^{\mathsf{sk}}}$
Output:	$\mathsf{pk} = g^h, \mathsf{sk} = h$	$(c_1, c_2) \in \mathbb{Z}_p \times \mathbb{Z}_p$	$m \in \mathbb{Z}_p$
Paillier			
	Key Generation	Encryption	Decryption
Input:	1^λ	$m \in \mathbb{Z}_n, \mathsf{pk}$	$c \in \mathbb{Z}_{n^2}, \mathsf{pk}, \mathsf{sk}$
Steps:	Choose λ-bit primes p and q and set $n = pq$	Choose $r \xleftarrow{\$} \mathbb{Z}_n^*$ and compute $c = (1+n)^m \cdot r^n \bmod n^2$	Compute $m = \frac{(c^{\phi(n)} \bmod n^2)-1}{n} \cdot \phi(n)^{-1} \bmod n$
Output:	$\mathsf{pk} = n, \mathsf{sk} = \phi(n)$	$c \in \mathbb{Z}_{n^2}$	$m \in \mathbb{Z}_n$

Table 4: Summary of ElGamal and Paillier encryption schemes.

B Proofs

Theorem 1. *Let Π be a PKE scheme, which is (computationally) randomizable. If Π is used in Π_{HPINEQ}, then Π_{HPINEQ} is complete, computationally sound and perfect HVZK.*

Proof. (**Completeness**). When interacting with an honest verifier, the prover can compute $\mathsf{Dec}_{\mathsf{sk}}(c'_b) = m'$, check whether it is equal to the message encrypted by c_0 or c_1 and send the value z so that the verifier always accepts. (**Soundness**). Let us define the following algorithm:

$\mathsf{GenInstance}(1^k, \mathcal{R}, \mathcal{M})$**:** It picks $(r_0, r_1) \xleftarrow{\$} \mathcal{R}^2$ and $m \xleftarrow{\$} \mathcal{M}$, generates $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^k)$, $c_0 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m; r_0)$, $c_1 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m; r_1)$, and returns $(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$.

We recall that $\forall\ y \notin \mathcal{K} \times \mathcal{C}^2$, the verifier aborts the protocol. It follows that $\forall y$ such that $y \notin L$ and $y \notin \mathcal{K} \times \mathcal{C}^2 : Pr[\langle \mathcal{B}(x), \mathcal{V}(z) \rangle(y) = 1] = 0$ for any witness x and any bit-string z. Furthermore, for all instances y such that $y \notin L$ and $y \in \mathcal{K} \times \mathcal{C}^2$, we have that $y \in \{(\mathsf{pk}, c_0, c_1) | (\mathsf{sk}, \mathsf{pk}, c_0, c_1) \leftarrow \mathsf{GenInstance}(1^k, \mathcal{M})\}$. This means that the soundness of the protocol Π_{HPINEQ} can be proven by showing that for any witness x, any bit-string z and any instance $y = (\mathsf{pk}, c_0, c_1)$ generated from the output of $\mathsf{GenInstance}$, $Pr[\langle \mathcal{B}(x), \mathcal{V}(z) \rangle(y) = 1]$ is negligible.

Let us now define an experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Sound}}$ that takes a witness x and a tuple $(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$ generated by $\mathsf{GenInstance}$ as input, where an adversary $\mathcal{A}$ plays one round of the protocol Π_{HPINEQ} as a (dishonest) prover against a *challenger* that plays the role of an honest verifier. We define $\mathcal{A}$ as a pair of p.p.t algorithms $(\mathcal{A}_1, \mathcal{A}_2)$ where $\mathcal{A}_1(x, y)$ instantiates the dishonest prover and returns a state st, and $\mathcal{A}_2(\mathsf{st}, c')$ corresponds to the interaction between the verifier and the prover (*i.e.*, it takes a challenge c' as input and returns a response z).

Game G_0	**Game** G_1	**Game** G_2
$\mathsf{st} \leftarrow \mathcal{A}_1(x,(\mathsf{pk},c_0,c_1))$	$\mathsf{st} \leftarrow \mathcal{A}_1(x,(\mathsf{pk},c_0,c_1))$	$\mathsf{st} \leftarrow \mathcal{A}_1(x,(\mathsf{pk},c_0,c_1))$
$r \xleftarrow{\$} \mathcal{R}; b \xleftarrow{\$} \{0,1\}$	$r \xleftarrow{\$} \mathcal{R}; b \xleftarrow{\$} \{0,1\}$	$r \xleftarrow{\$} \mathcal{R}; b \xleftarrow{\$} \{0,1\}$
$c'_b \leftarrow \mathsf{Rand}(c_b, r)$	$m \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c_0)$	$m \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c_0)$
$z \leftarrow \mathcal{A}_2(\mathsf{st}, c'_b)$	$c'_0 \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(m, r)$	$c'_0 \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(m, r)$
if $(z = b)$ **return** 1	$c'_1 \leftarrow \mathsf{Rand}(c_1, r)$	$m \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c_1)$
else return 0	$z \leftarrow \mathcal{A}_2(\mathsf{st}, c'_b)$	$c'_1 \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(m, r)$
	if $(z = b)$ **return** 1	$z \leftarrow \mathcal{A}_2(\mathsf{st}, c'_b)$
	else return 0	**if** $(z = b)$ **return** 1
		else return 0

Fig. 10: Sequence of games for Π_{HPINEQ}.

The experiment $\mathsf{Exp}^{\mathsf{Sound}}_{\mathcal{A}}(1^k,(x,\mathsf{sk},\mathsf{pk},c_0,c_1))$ runs $\mathsf{st} \leftarrow \mathcal{A}_1(x,(\mathsf{pk},c_0,c_1))$, $r \xleftarrow{\$} \mathcal{R}; b \xleftarrow{\$} \{0,1\}$, $c'_b \leftarrow \mathsf{Rand}(c_b,r)$ and $z \leftarrow \mathcal{A}_2(\mathsf{st},c'_b)$. If $(z \neq b)$ it returns 0, otherwise it returns 1.

Next, we prove that for any adversary $\mathcal{A}$, the probability of winning this experiment is negligibly close to $1/2$ for any tuple $(x,\mathsf{sk},\mathsf{pk},c_0,c_1)$. Observe that for all y such that $y \notin L$ and $y \in \mathcal{K}\times\mathcal{C}^2$, it holds that $y \in \{(\mathsf{pk},c_0,c_1) \,|(\mathsf{sk},\mathsf{pk},c_0,c_1) \leftarrow \mathsf{GenInstance}(1^k,\mathcal{M})\}$. Since the protocol is repeated k times, it follows that for all instances y such that $y \notin L$ and $y \in \mathcal{K}\times\mathcal{C}^2$, $Pr[\langle\mathcal{B}(x),\mathcal{V}(z)\rangle(y) = 1]$ is negligibly close to $1/2^k$, which means that the soundness probability is also negligible.

We define a sequence of games (Figure 10) which are played between an adversary $\mathcal{A}$ and a challenger, where the first game G_0 is $\mathsf{Exp}^{\mathsf{Sound}}_{\mathcal{A}}(1^k,(x,\mathsf{sk},\mathsf{pk},c_0,c_1))$ for a fixed $(x,\mathsf{sk},\mathsf{pk},c_0,c_1)$. Considering the adversary's view, game G_0 represents a real execution of one round of the protocol Π_{HPINEQ}. We say that "$\mathcal{A}$ wins G_0" when the output is 1. G_1 is defined as G_0 except that we replace the instruction $c'_0 \leftarrow \mathsf{Rand}(c_0,r)$ by $m \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c_0)$ and $c'_0 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m;r)$.

We claim and prove by reduction that: $|\mathsf{Pr}[\mathcal{A} \text{ wins } G_0] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_1]| \leq \epsilon_{\mathsf{rand}}(k)$ where $\epsilon_{\mathsf{rand}}(k)$ is the re-randomizability advantage of the encryption scheme (Definition 10). Let c' be a ciphertext generated by one of these two methods: $r' \xleftarrow{\$} \mathcal{R}; c' \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m;r')$ or $r' \xleftarrow{\$} \mathcal{R}; c' \leftarrow \mathsf{Rand}(c_0,r')$ (where $m = \mathsf{Dec}_{\mathsf{sk}}(c_0)$. We build the distinguisher $\mathcal{D}(\mathsf{pk},c_0,c')$ as follows: $\mathcal{D}$ simulates the protocol Π_{HPINEQ} for the fixed statement $y = (\mathsf{pk},c_0,c_1)$, except that if $b = 0$, then it sets $c'_0 = c'$. If the proof is accepted then $\mathcal{D}$ returns 1, else it returns 0.

- If $c' \leftarrow \mathsf{Rand}(c_0,r')$ then $\mathcal{D}$ perfectly simulates G_0, so:

  $\mathsf{Pr}[\mathcal{A} \text{ wins } G_0] = \mathsf{Pr}\left[r' \xleftarrow{\$} \mathcal{R}; c' \leftarrow \mathsf{Rand}(c,r'); b \leftarrow \mathcal{D}(\mathsf{pk},c_0,c'); : b = 1\right]$
- If $c' \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m;r')$ then $\mathcal{D}$ perfectly simulates G_1, so:

  $\mathsf{Pr}[\mathcal{A} \text{ wins } G_1] = \mathsf{Pr}\left[r' \xleftarrow{\$} \mathcal{R}; c' \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m;r'); b \leftarrow \mathcal{D}(\mathsf{pk},c_0,c'); : b = 1\right]$

  which concludes the proof of the claim.

Similarly, G_2 is defined as G_1 except that we replace the instruction $c'_1 \leftarrow \mathsf{Rand}(c_1,r)$ by $m \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c_1)$ and $c'_1 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m;r)$.

We claim that: $|\mathsf{Pr}[\mathcal{A} \text{ wins } G_1] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_2]| \leq \epsilon_{\mathsf{rand}}(k)$, which we prove as in the previous game.

Finally, since $m = \mathsf{Dec}_{\mathsf{sk}}(c_0) = \mathsf{Dec}_{\mathsf{sk}}(c_1)$ and c'_b is always computed from $\mathsf{Enc}_{\mathsf{pk}}(m)$ in G_2, we deduce that c'_b does not depend on b, which implies that $\mathcal{A}$ receives no information that depends on b. Therefore, the best strategy of $\mathcal{A}$ in G_2 is to guess b at random, so: $\mathsf{Pr}[\mathcal{A} \text{ wins } G_2] = \frac{1}{2}$.

Based on the indistinguishability of transitions from the given game sequence, we conclude that if we repeat the protocols k times, the probability that $\mathcal{A}$ breaks the soundness is negligible and majored by: $\mathsf{Pr}[\mathcal{A} \text{ wins } G_0]^k \leq \left(2 \cdot \epsilon_{\mathsf{rand}}(k) + \frac{1}{2}\right)^k$. (**Zero-Knowledge**). We define the simulator $\mathcal{S}(y)$ where $y = (\mathsf{pk}, c_0, c_1)$. The simulator $\mathcal{S}$ picks $b \xleftarrow{\$} \{0,1\}$ and computes $r \xleftarrow{\$} \mathcal{R}$ and $c'_b \leftarrow \mathsf{Rand}(\mathsf{pk}, c_b, r)$. Finally, it returns (c'_b, b). The simulator acts as in the real protocol, so it perfectly simulates the proof.

Theorem 2. *Let Π be a PKE scheme, which is perfectly strong randomizable and derandomizable. Let Γ be a commitment scheme, which is computationally binding and perfectly hiding. If Π and Γ are used in the protocol Π_{PINEQ}, then Π_{PINEQ} is complete, computationally sound and perfect zero-knowledge.*

Proof. (**Completeness**). Since Π is randomizable, the prover can decrypt c'_b and c'_{1-b} to compare the messages with the decryptions of c_0 and c_1 to determine z. Hence P commits a value, and V always accepts its proof. (**Soundness**). Let us define the following algorithm:

$\mathsf{GenInstance}(1^k, \mathcal{R}, \mathcal{M})$: It picks $(r, r_0, r_1) \xleftarrow{\$} \mathcal{R}^3$ and $m \xleftarrow{\$} \mathcal{M}$, generates $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^k)$, $c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m; r)$, $c_0 \leftarrow \mathsf{Rand}(c, r_0)$, $c_1 \leftarrow \mathsf{Rand}(c, r_1)$, and returns $(\mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1)$.

We recall that $\forall\ y \notin \mathcal{K} \times \mathcal{C}^2$, the verifier aborts the protocol. It follows that $\forall y$ such that $y \notin L$ and $y \notin \mathcal{K} \times \mathcal{C}^2$: $Pr[\langle \mathcal{B}(x), \mathcal{V}(z) \rangle(y) = 1] = 0$ for any witness x and any bit-string z. Since the scheme is perfectly randomizable and derandomizable, the ciphertexts produced by the encryption algorithm follow the same distribution than the ones produced by the randomization algorithm, which implies that $\forall y$ such that $y \notin L$ and $y \in \mathcal{K} \times \mathcal{C}^2$, we have $y \in \{(\mathsf{pk}, c_0, c_1) | (\mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1) \leftarrow \mathsf{GenInstance}(1^k, \mathcal{R}, \mathcal{M})\}$. This means that the soundness of the protocol Π_{PINEQ} can be proven by showing that for any witness x, $\forall z$ and $\forall y = (\mathsf{pk}, c_0, c_1)$ generated from the output of $\mathsf{GenInstance}$, $Pr[\langle \mathcal{B}(x), \mathcal{V}(z) \rangle(y) = 1]$ is negligible.

Let us define an experiment $\mathsf{Exp}^{\mathsf{Sound}}_{\mathcal{A}}$ that takes a witness x and a tuple $(\mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1)$ generated by $\mathsf{GenInstance}$ as input. In the experiment, the adversary $\mathcal{A}$ plays one round of the protocol Π_{PINEQ} as a (dishonest) prover against a *challenger* that plays the role of an honest verifier. We define $\mathcal{A}$ as a triplet of p.p.t algorithms $(\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ where $\mathcal{A}_1(x, y)$: instantiates the dishonest prover and returns st_1, $\mathcal{A}_2(\mathsf{st}_1, c'_b)$ corresponds to the first interaction of the protocol (*i.e.*, it takes a challenge c'_b as input and returns a state st_2 and the response comm), and $\mathcal{A}_3(\mathsf{st}_2, r, b)$ corresponds to the second interaction (*i.e.*, it takes a challenge r and b as input and returns a response op).

Game G_0	**Game** G_1
$\mathsf{st}_1 \leftarrow \mathcal{A}_1(x, (\mathsf{pk}, c_0, c_1))$	$\mathsf{st}_1 \leftarrow \mathcal{A}_1(x, (\mathsf{pk}, c_0, c_1))$
$b \xleftarrow{\$} \{0,1\}; r \xleftarrow{\$} \mathcal{R}$	$b \xleftarrow{\$} \{0,1\}; r' \xleftarrow{\$} \mathcal{R}; r \leftarrow \mathsf{RandR}(r_b, r')$
$c'_b \leftarrow \mathsf{Rand}(c_b, r)$	$c'_b \leftarrow \mathsf{Rand}(c, r')$
$(\mathsf{comm}, \mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{st}_1, c'_b)$	$(\mathsf{comm}, \mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{st}_1, c'_b)$
$\mathsf{op} \leftarrow \mathcal{A}_3(\mathsf{st}_2, r, b)$	$\mathsf{op} \leftarrow \mathcal{A}_3(\mathsf{st}_2, r, b)$
if $(\mathsf{Open}(\mathsf{comm}, \mathsf{op}) = b)$ **return** 1	**if** $(\mathsf{Open}(\mathsf{comm}, \mathsf{op}) = b)$ **return** 1
else return 0	**else return** 0

Fig. 11: Sequence of games for Π_{PINEQ}.

The experiment $\mathsf{Exp}^{\mathsf{Sound}}_{\mathcal{A}}(1^k, (x, \mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1))$ runs $\mathsf{st}_1 \leftarrow \mathcal{A}_1(x, (\mathsf{pk}, c_0, c_1))$, $b \xleftarrow{\$} \{0,1\}$, $r \xleftarrow{\$} \mathcal{R}$, $c'_b \leftarrow \mathsf{Rand}(c_b, r)$, $(\mathsf{comm}, \mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{st}_1, c'_b)$ and $\mathsf{op} \leftarrow \mathcal{A}_3(\mathsf{st}_2, r, b)$. If $(\mathsf{Open}(\mathsf{comm}, \mathsf{op}) \neq b)$ then it returns 0, otherwise 1.

We prove that for any adversary $\mathcal{A}$, the probability of winning this experiment is negligibly close to $1/2$ for any tuple $(x, \mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1)$. $\forall y$ such that $y \notin L$ and $y \in \mathcal{K} \times \mathcal{C}^2$, we have that $y \in \{(\mathsf{pk}, c_0, c_1) | (\mathsf{sk}, \mathsf{pk}, r_0, r_1, c_0, c_1) \leftarrow \mathsf{GenInstance}(1^k, \mathcal{R}, \mathcal{M})\}$. Moreover, since the protocol is repeated k times, one can deduce that for all instances y such that $y \notin L$ and $y \in \mathcal{K} \times \mathcal{C}^2$, it holds that $Pr[\langle \mathcal{B}(x), \mathcal{V}(z) \rangle(y) = 1]$ is negligibly close to $1/2^k$. This means that the soundness probability is negligible. In Figure 11, we define a sequence of games between an adversary $\mathcal{A}$ and a challenger. For a fixed $(x, \mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1)$, the game G_0 is $\mathsf{Exp}^{\mathsf{Sound}}_{\mathcal{A}}(1^k, (x, \mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1))$.

Considering the adversary's view, game G_0 is a real execution of one round of Π_{PINEQ}. We say that "$\mathcal{A}$ wins G_0" when the output is 1. We define a game G_1 that proceeds as G_0 except that we replace the instructions $r \xleftarrow{\$} \mathcal{R}$ and $c'_b \leftarrow \mathsf{Rand}(c_b, r)$ by $r' \xleftarrow{\$} \mathcal{R}$, $c'_b \leftarrow \mathsf{Rand}(c, r')$ and $r \leftarrow \mathsf{RandR}(r_b, r')$. We note that since the encryption scheme is strongly randomizable and derandomizable, and $c_b \leftarrow \mathsf{Rand}(c, r_b)$, it holds that in G_1: $c = \mathsf{Rand}(c_b, r_b)$ and $c'_b = \mathsf{Rand}(c_b, \mathsf{RandR}(r_b, r')) = \mathsf{Rand}(c_b, r)$. On the other hand, since the encryption scheme is perfectly strongly randomizable, an element r produced by the sequence of instructions $r' \xleftarrow{\$} \mathcal{R}; r \leftarrow \mathsf{RandR}(r_b, r')$ follows the same distribution as $r \xleftarrow{\$} \mathcal{R}$, which implies that $\mathsf{Pr}[\mathcal{A} \text{ wins } G_0] = \mathsf{Pr}[\mathcal{A} \text{ wins } G_1]$.

Next, we claim and prove by reduction that $\left|\mathsf{Pr}[\mathcal{A} \text{ wins } G_1] - \frac{1}{2}\right| = \frac{\epsilon_{binding}(k)}{2}$. We use the following strategy. In game G_1, the challenger can generate a random coin r for both $b = 1$ and $b = 0$ on the same challenge $c' = c'_0 = c'_1$ because it builds c'_b by computing $r' \xleftarrow{\$} \mathcal{R}$, $c' \leftarrow \mathsf{Rand}(c, r')$, and r by computing $r \leftarrow \mathsf{RandR}(r_b, r')$. To break the soundness of the protocol, $\mathcal{A}$ must be able to succeed in both cases ($b = 1$ and $b = 0$) with non negligible probability at each round. If it is not the case, the advantage of $\mathcal{A}$ is bounded by a value that is negligibly close to $1/2^k$. We show that if such an adversary exists, we can build an algorithm that breaks the binding property of the commitment scheme. If the adversary is able to succeed for both cases ($b = 1$ and $b = 0$), then it is able to open its

commitment comm for two different message $z = 0$ and $z = 1$, which is equivalent to breaking the binding property.

We build an adversary $\mathcal{B}$ that has an advantage $\epsilon_{binding}(k)$ on the binding experiment as follows: $\mathcal{B}$ receives ck and simulates the game G_1 for $b = 0$ to $\mathcal{A}$ using ck as commitment key. More formally, $\mathcal{B}$ runs $\mathsf{st}_1 \leftarrow \mathcal{A}_1(x, (\mathsf{pk}, c_0, c_1))$ $r' \xleftarrow{\$} \mathcal{R}$, $\hat{r}_0 \leftarrow \mathsf{RandR}(r_0, r')$, $\hat{r}_1 \leftarrow \mathsf{RandR}(r_1, r')$, $c' \leftarrow \mathsf{Rand}(c, r')$, $(\mathsf{comm}, \mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{st}_1, c')$, $\mathsf{op}_0 \leftarrow \mathcal{A}_3(\mathsf{st}_2, \hat{r}_0, b)$, $\mathsf{op}_1 \leftarrow \mathcal{A}_3(\mathsf{st}_2, \hat{r}_1, b)$. $\mathcal{B}$ sets $\mathsf{win}_0 = 1$ if and only if $(\mathsf{Open}(\mathsf{comm}, \mathsf{op}_0) = 0)$, and $\mathsf{win}_1 = 1$ if and only if $(\mathsf{Open}(\mathsf{comm}, \mathsf{op}_1) = 1)$. If $\mathsf{win}_0 = 1$ and $\mathsf{win}_1 = 1$, then $\mathcal{B}$ returns $(\mathsf{comm}, \mathsf{op}_0, \mathsf{op}_1)$. Note that in this case, $\mathsf{Open}(\mathsf{comm}, \mathsf{op}_0) = 0 \neq 1 = \mathsf{Open}(\mathsf{comm}, \mathsf{op}_1)$, so $\mathcal{B}$ wins the experiment.

First, we observe that: $\mathsf{Pr}[\mathcal{A} \text{ wins } G_1] = \mathsf{Pr}[b = 0] \cdot \mathsf{Pr}[\mathsf{win}_0 = 1] + \mathsf{Pr}[b = 1] \cdot \mathsf{Pr}[\mathsf{win}_1 = 1] = \frac{1}{2}\mathsf{Pr}[\mathsf{win}_0 = 1] + \frac{1}{2}\mathsf{Pr}[\mathsf{win}_1 = 1]$.

Let us now set three events E_0, E_1 and E_2 as follows:

- E_0 ="$\mathsf{Open}(\mathsf{comm}, \mathsf{op}_0) = \perp$ or $\mathsf{Open}(\mathsf{comm}, \mathsf{op}_1) = \perp$".
- E_1 ="$\mathsf{Open}(\mathsf{comm}, \mathsf{op}_0) \neq \mathsf{Open}(\mathsf{comm}, \mathsf{op}_1)$ and $\mathsf{Open}(\mathsf{comm}, \mathsf{op}_0) \neq \perp$ and $\mathsf{Open}(\mathsf{comm}, \mathsf{op}_1) \neq \perp$". Note that $\mathsf{Pr}[E_1] = \epsilon_{binding}(k)$.
- E_2 ="$\mathsf{Open}(\mathsf{comm}, \mathsf{op}_0) = \mathsf{Open}(\mathsf{comm}, \mathsf{op}_1)$ and $\mathsf{Open}(\mathsf{comm}, \mathsf{op}_0) \neq \perp$ and $\mathsf{Open}(\mathsf{comm}, \mathsf{op}_1) \neq \perp$".

We oberve that $\mathsf{Pr}[E_0] + \mathsf{Pr}[E_1] + \mathsf{Pr}[E_2] = 1 \Leftrightarrow \mathsf{Pr}[E_2] = 1 - \mathsf{Pr}[E_0] - \mathsf{Pr}[E_1] \Leftrightarrow \mathsf{Pr}[E_2] \leq 1 - \epsilon_{binding}(k)$. Moreover, we have:

$$\begin{aligned}\mathsf{Pr}[\mathsf{win}_1 = 1] &= \textstyle\sum_{i=0}^{2} \mathsf{Pr}[E_i] \cdot \mathsf{Pr}[\mathsf{win}_1 = 1 | E_i] \\ &\leq \mathsf{Pr}[E_1] \cdot \mathsf{Pr}[\mathsf{win}_1 = 1 | E_1] + \mathsf{Pr}[E_2] \cdot \mathsf{Pr}[\mathsf{win}_1 = 1 | E_2] \\ &\leq \epsilon_{binding}(k) \cdot \mathsf{Pr}[\mathsf{win}_1 = 1 | E_1] + (1 - \epsilon_{binding}(k)) \cdot \mathsf{Pr}[\mathsf{win}_1 = 1 | E_2].\end{aligned}$$

In the case E_2, if $\mathcal{A}$ wins the case $b = 0$, it loses the case $b = 1$. We have $\mathsf{Pr}[\mathsf{win}_1 = 1 | E_2] = 1 - \mathsf{Pr}[\mathsf{win}_0 = 1]$. On the other hand, in the case E_1, $\mathsf{Pr}[\mathsf{win}_1 = 1 | E_1] = \mathsf{Pr}[\mathsf{win}_0 = 1]$, which implies that:

$$\begin{aligned}\mathsf{Pr}[\mathsf{win}_1 = 1] &\leq \epsilon_{binding}(k) \cdot \mathsf{Pr}[\mathsf{win}_0 = 1] + (1 - \epsilon_{binding}(k)) \cdot (1 - \mathsf{Pr}[\mathsf{win}_0 = 1]) \\ &\leq 1 - \mathsf{Pr}[\mathsf{win}_0 = 1] + \epsilon_{binding}(k) \cdot (2 \cdot \mathsf{Pr}[\mathsf{win}_0 = 1] - 1).\end{aligned}$$

This in turns implies that:

$\mathsf{Pr}[\mathsf{win}_1 = 1] + \mathsf{Pr}[\mathsf{win}_0 = 1] \leq 1 + \epsilon_{binding}(k) \cdot (2 \cdot \mathsf{Pr}[\mathsf{win}_0 = 1] - 1)$

$\Leftrightarrow \frac{1}{2} \cdot \mathsf{Pr}[\mathsf{win}_1 = 1] + \frac{1}{2} \cdot \mathsf{Pr}[\mathsf{win}_0 = 1] \leq \frac{1}{2} + \epsilon_{binding}(k) \cdot \left(\mathsf{Pr}[\mathsf{win}_0 = 1] - \frac{1}{2}\right)$

$\Leftrightarrow \left|\mathsf{Pr}[\mathcal{A} \text{ wins } G_1] - \frac{1}{2}\right| \leq \frac{\epsilon_{binding}(k)}{2}$.

Finally, if the protocol is repeated k times, then we deduce that the soundness probability is: $\mathsf{Pr}[\mathcal{A} \text{ wins } G_0]^k = \mathsf{Pr}[\mathcal{A} \text{ wins } G_1]^k \leq \left(\frac{1+\epsilon_{binding}(k)}{2}\right)^k$. Since the commitment scheme has the binding property by hypothesis, $\epsilon_{binding}(k)$ is negligible and so is $\mathsf{Pr}[\mathcal{A}\text{wins } G_0]^k$, which concludes the proof of the soundness.

(**Zero-Knowledge**). Let $\mathcal{V}^*$ be a dishonest verifier. We define a Simulator $\mathcal{S}_{\mathcal{V}^*}(y)$ where $y = (\mathsf{pk}, c_0, c_1)$ that perfectly simulates $\mathcal{V}^*$. The simulator $\mathcal{S}$ generates c'_b and r as the dishonest verifier $\mathcal{V}^*$.

- If $c'_b = \mathsf{Rand}(c_z, r_z)$ where $z \in \{0, 1\}$, then the simulator runs $(\mathsf{comm}, \mathsf{op}) \leftarrow \mathsf{Commit}(z)$ and returns the transcript $(c'_b, \mathsf{comm}, (r, z), \mathsf{op})$.
- If $c'_b \neq \mathsf{Rand}(c_z, r_z)$ where $z \in \{0, 1\}$, then the simulator picks comm in the uniform distribution on the commitment set and returns the transcript $(c'_b, \mathsf{comm}, (r, z), \perp)$.

The simulator follows the same distribution as the real protocol. If the verifier sends a wrong r, then it simulates the prover's messages using a random commitment comm and the $\perp$ symbol. Indeed, since the commitment's open key remains unknown, the prover commitment is like a random commitment picked in the uniform distribution according to the perfect hiding property.

Theorem 3. *Let Π be a PKE scheme, which is (computationally) randomizable, (computationally) message-randomizable and message-random-extractable. If Π is the scheme used in Π_{HPEQ}, then Π_{HPEQ} is complete, computationally sound and perfect HVZK.*

Proof. (**Completeness**). When interacting with an honest verifier, the prover can compute $\mathsf{Dec}_{\mathsf{sk}}(c'_b) = m'$. Then, given that $c_0, c_1 \in \mathsf{Enc}_{\mathsf{pk}}(m)$ and that the scheme is message-random-extractable, the prover can always compute $z = r_M = \mathsf{MsgRandExt}(m', m)$ so that the verifier always accepts.
(**Soundness**). Let us define the following algorithm:
$\mathsf{GenInstance}(1^k, \mathcal{R}, \mathcal{M})$**:** It picks $(r_0, r_1) \xleftarrow{\$} \mathcal{R}^2$, $m_0 \xleftarrow{\$} \mathcal{M}$ and $m_1 \xleftarrow{\$} \mathcal{M}\backslash\{m_0\}$, generates $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^k)$, $c_0 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_0; r_0)$ and $c_1 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_1; r_1)$, and returns $(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$.

We recall that for all $y \notin \mathcal{K} \times \mathcal{C}^2$, the verifier aborts the protocol. Furthermore, for all instances y such that $y \notin L$ and $y \notin \mathcal{K} \times \mathcal{C}^2$, it holds that $Pr[\langle \mathcal{B}(x), \mathcal{V}(z)\rangle(y) = 1] = 0$ for any witness x and any bit-string z.

On the other hand, for all instances y such that $y \notin L$ and $y \in \mathcal{K} \times \mathcal{C}^2$, we stress that $y \in \{(\mathsf{pk}, c_0, c_1) | (\mathsf{sk}, \mathsf{pk}, c_0, c_1) \leftarrow \mathsf{GenInstance}(1^k, \mathcal{M})\}$, which means that the soundness of the protocol Π_{HPEQ} can be proven by showing that for any witness x, any bit-string z and any instance $y = (\mathsf{pk}, c_0, c_1)$ generated from the output of $\mathsf{GenInstance}$, $Pr[\langle \mathcal{B}(x), \mathcal{V}(z)\rangle(y) = 1]$ is negligible.

We define an experiment that takes on input a witness x and a tuple $(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$ generated by $\mathsf{GenInstance}$, where an adversary $\mathcal{A}$ plays one round of the protocol Π_{HPEQ} as a (dishonest) prover with a *challenger* that plays the role of an honest verifier. We define $\mathcal{A}$ as a pair of p.p.t algorithms $(\mathcal{A}_1, \mathcal{A}_2)$:
$\mathcal{A}_1(x, y)$**:** This algorithm instantiates the dishonest prover. It returns a state st.
$\mathcal{A}_2(\mathsf{st}, c')$**:** This algorithm corresponds to the interaction between the verifier and the prover. It takes a challenge c' as input and returns a response z.
The experiment is defined as follows:
$\mathsf{Exp}^{\mathsf{Sound}}_{\mathcal{A}}(1^k, (x, \mathsf{sk}, \mathsf{pk}, c_0, c_1))$**:** compute $\mathsf{st} \leftarrow \mathcal{A}_1(x, (\mathsf{pk}, c_0, c_1))$, $r \xleftarrow{\$} \mathcal{R}$, $r_{\mathsf{m}} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}}$, $b \xleftarrow{\$} \{0, 1\}$, $c'_b \leftarrow \mathsf{Rand}(c_b, r)$, $c''_b \leftarrow \mathsf{MsgRandC}(c'_b, r_{\mathsf{m}})$, $z \leftarrow \mathcal{A}(\mathsf{st}, c''_b)$, if $(z = r_{\mathsf{m}})$ then return 1, else return 0.
In what follows, we will prove that for any adversary $\mathcal{A}$, the probability of winning this experiment is negligibly close to $1/2$ for any tuple $(x, \mathsf{sk}, \mathsf{pk}, c_0, c_1)$. Since for all instances y such that $y \notin L$ and $y \in \mathcal{K} \times \mathcal{C}^2$, it holds that $y \in \{(\mathsf{pk}, c_0, c_1) | (\mathsf{sk}, \mathsf{pk}, c_0, c_1) \leftarrow \mathsf{GenInstance}(1^k, \mathcal{M})\}$, and since the protocol is repeated k times, we will deduce that for all instances y such that $y \notin L$ and $y \in \mathcal{K} \times \mathcal{C}^2$, it holds that $Pr[\langle \mathcal{B}(x), \mathcal{V}(z)\rangle(y) = 1]$ is negligibly close to $1/2^k$, which means that the soundness probability is negligible.

We define a sequence of games (Figure 12) which are played between an adversary $\mathcal{A}$ and a challenger, where the first game G_0 is $\mathsf{Exp}^{\mathsf{Sound}}_{\mathcal{A}}(1^k, (x, \mathsf{sk}, \mathsf{pk}, c_0, c_1))$ for a fixed $(x, \mathsf{sk}, \mathsf{pk}, c_0, c_1)$. Considering the adversary's view, game G_0 represents a real execution of one round of the protocol Π_{HPEQ}. We say that "$\mathcal{A}$ wins G_0" when the output is 1.

Game G_0	**Game** G_1	**Game** G_2
$\mathsf{st} \leftarrow \mathcal{A}_1(x, \mathsf{pk}, c_0, c_1)$	$\mathsf{st} \leftarrow \mathcal{A}_1(x, \mathsf{pk}, c_0, c_1)$	$\mathsf{st} \leftarrow \mathcal{A}_1(x, \mathsf{pk}, c_0, c_1)$
$r \xleftarrow{\$} \mathcal{R}; r_{\mathsf{m}} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}}; b \xleftarrow{\$} \{0,1\}$	$r \xleftarrow{\$} \mathcal{R}; r_{\mathsf{m}} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}}; b \xleftarrow{\$} \{0,1\}$	$r \xleftarrow{\$} \mathcal{R}; r_{\mathsf{m}} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}}; b \xleftarrow{\$} \{0,1\}$
$c'_b \leftarrow \mathsf{Rand}(c_b, r)$	$m_0 = \mathsf{Dec}_{\mathsf{sk}}(c_0)$	$m_0 = \mathsf{Dec}_{\mathsf{sk}}(c_0)$
$c''_b \leftarrow \mathsf{MsgRandC}(c'_b, r_{\mathsf{m}})$	$c'_0 \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(m_0, r)$	$c'_0 \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(m_0, r)$
$z \leftarrow \mathcal{A}_2(\mathsf{st}, c''_b)$	$c'_1 \leftarrow \mathsf{Rand}(c_1, r)$	$m_1 = \mathsf{Dec}_{\mathsf{sk}}(c_1)$
if $(z = r_{\mathsf{m}})$ **return** 1 **else** 0	$c''_b \leftarrow \mathsf{MsgRandC}(c'_b, r_{\mathsf{m}})$	$c'_1 \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(m_1, r)$
	$z \leftarrow \mathcal{A}_2(\mathsf{st}, c''_b)$	$c''_b \leftarrow \mathsf{MsgRandC}(c'_b, r_{\mathsf{m}})$
	if $(z = r_{\mathsf{m}})$ **return** 1 **else** 0	$z \leftarrow \mathcal{A}_2(\mathsf{st}, c''_b)$
		if $(z = r_{\mathsf{m}})$ **return** 1 **else** 0

Game G_3	**Game** G_4
$\mathsf{st} \leftarrow \mathcal{A}_1(x, \mathsf{pk}, c_0, c_1)$	$\mathsf{st} \leftarrow \mathcal{A}_1(x, \mathsf{pk}, c_0, c_1)$
$r \xleftarrow{\$} \mathcal{R}; r_{\mathsf{m}} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}}; b \xleftarrow{\$} \{0,1\}$	$r \xleftarrow{\$} \mathcal{R}; r_{\mathsf{m}} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}}; b \xleftarrow{\$} \{0,1\}$
$m_0 = \mathsf{Dec}_{\mathsf{sk}}(c_0)$	$m_0 = \mathsf{Dec}_{\mathsf{sk}}(c_0)$
$c'_0 \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(m_0, r)$	$c'_0 \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(m_0, r)$
$m_1 = \mathsf{Dec}_{\mathsf{sk}}(c_1)$	$m_1 = \mathsf{Dec}_{\mathsf{sk}}(c_1)$
$c'_1 \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(m_1, r)$	$c'_1 \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(m_1, r)$
$m'_0 \xleftarrow{\$} \mathcal{M}; c''_0 \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(m'_0, r)$	$m'_0 \xleftarrow{\$} \mathcal{M}; c''_0 \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(m'_0, r)$
$c''_1 \leftarrow \mathsf{MsgRandC}(c'_1, r_{\mathsf{m}})$	$m'_1 \xleftarrow{\$} \mathcal{M}; c''_1 \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(m'_1, r)$
$z \leftarrow \mathcal{A}_2(\mathsf{st}, c''_b)$	$z \leftarrow \mathcal{A}_2(\mathsf{st}, c''_b)$
if $(b = 0)$ **then**	**return** $z = \mathsf{MsgRandExt}(m_b, m'_b)$
return $z = \mathsf{MsgRandExt}(m_0, m'_0)$	
else return $z = r_{\mathsf{m}}$	

Fig. 12: Sequence of games for Π_{HPEQ}.

G_1 is defined as G_0 except that we replace the instruction $c'_0 \leftarrow \mathsf{Rand}(c_0, r)$ by $m_0 \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c_0)$ and $c'_0 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_0; r)$.

We claim and prove by reduction that $|\mathsf{Pr}[\mathcal{A} \text{ wins } G_0] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_1]| \leq \epsilon_{\mathsf{rand}}(k)$ where $\epsilon_{\mathsf{rand}}(k)$ is the re-randomizability advantage of the encryption scheme (Definition 10). Let c' be a ciphertext generated by one of these two methods $r' \xleftarrow{\$} \mathcal{R}; c' \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_0; r')$ or $r' \xleftarrow{\$} \mathcal{R}; c' \leftarrow \mathsf{Rand}(c_0, r')$ (where $m_0 \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c_0)$). We build the distinguisher $\mathcal{D}(\mathsf{pk}, c_0, c')$ as follows: $\mathcal{D}$ simulates the protocol Π_{HPEQ}, except that if $b = 0$, then it sets $c'_0 = c'$. If the proof is accepted then $\mathcal{D}$ returns 1, else it returns 0.

– If $c' \leftarrow \mathsf{Rand}(c_0, r')$ then $\mathcal{D}$ perfectly simulates G_0, so:

$$\mathsf{Pr}[\mathcal{A} \text{ wins } G_0] = \mathsf{Pr}\left[r' \xleftarrow{\$} \mathcal{R}; c' \leftarrow \mathsf{Rand}(c_0, r'); b \leftarrow \mathcal{D}(\mathsf{pk}, c_0, c'); : b = 1\right]$$

– If $c' \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_0; r')$ then $\mathcal{D}$ perfectly simulates G_1, so:

$$\mathsf{Pr}[\mathcal{A} \text{ wins } G_1] = \mathsf{Pr}\left[r' \xleftarrow{\$} \mathcal{R}; c' \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_0; r'); b \leftarrow \mathcal{D}(\mathsf{pk}, c_0, c'); : b = 1\right]$$

which concludes the proof of the claim.

Similarly, G_2 is defined as G_1 except that we replace the instruction $c'_1 \leftarrow \mathsf{Rand}(c_1, r)$ by $m_1 \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c_1)$ and $c'_1 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_1; r)$. We claim and prove as before that $|\mathsf{Pr}[\mathcal{A} \text{ wins } G_1] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_2]| \leq \epsilon_{\mathsf{rand}}(k)$.

Game G_3 is defined as G_2 except that we replace the instruction $c''_0 \xleftarrow{\$} \mathsf{MsgRand}(c'_0, r_{\mathsf{m}})$ by $m'_0 \xleftarrow{\$} \mathcal{M}$ and $c''_0 \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(m'_0, r)$. We claim and prove by reduction that $|\mathsf{Pr}[\mathcal{A} \text{ wins } G_2] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_3]| \leq \epsilon_{\mathsf{msgRand}}(k)$ where $\epsilon_{\mathsf{msgRand}}(k)$ is the message-randomizability advantage of the encryption scheme (Definition 14). Let c'' be a ciphertext generated by one of these two methods $m'_0 \xleftarrow{\$} \mathcal{M}; c'' \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m'_0; r'_0)$ or $r_{\mathsf{m}} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}}; c'' \leftarrow \mathsf{MsgRandC}(c'_0, r_{\mathsf{m}})$. We build a distinguisher $\mathcal{D}(\mathsf{pk}, c'_0, c'')$ as follows: $\mathcal{D}$ simulates the protocol Π_{HPEQ}, except that if $b = 0$, then it sets $c''_0 = c''$. If the proof is accepted then $\mathcal{D}$ returns 1 , else it returns 0.

– If $c'' \leftarrow \mathsf{MsgRandC}(c'_0, r_{\mathsf{m}})$ then $\mathcal{D}$ perfectly simulates G_2, so:

$$\mathsf{Pr}[\mathcal{A} \text{ wins } G_2] = \mathsf{Pr}\begin{bmatrix} r_{\mathsf{m}} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}}; \\ c'' \leftarrow \mathsf{MsgRandC}(c'_0, r_{\mathsf{m}}); : b = 1 \\ b \leftarrow \mathcal{D}(\mathsf{pk}, c'_0, c''); \end{bmatrix}$$

– If $c'' \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m'_0; r'_0)$ then $\mathcal{D}$ perfectly simulates G_3, so:

$$\mathsf{Pr}[\mathcal{A} \text{ wins } G_3] = \mathsf{Pr}\left[m'_0 \xleftarrow{\$} \mathcal{M}; c'' \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m'_0; r'_0); b \leftarrow \mathcal{D}(\mathsf{pk}, c'_0, c''); : b = 1\right]$$

which concludes the proof of the claim.

Let us now define G_4 as G_3 except that we replace the instruction $c''_1 \xleftarrow{\$} \mathsf{MsgRand}(c'_1, r_{\mathsf{m}})$ by $m'_1 \xleftarrow{\$} \mathcal{M}$ and $c''_1 \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(m'_0, r)$. By the same argument as before, it follows that $|\mathsf{Pr}[\mathcal{A} \text{ wins } G_3] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_4]| \leq \epsilon_{\mathsf{msgRand}}(k)$.

Finally, in G_4, m'_0 and m'_1 are randomly picked and independent of b. Therefore, the adversary receives no information that depends on b from c''_b. Since m_0 and m_1 are different, it follows that the best strategy $\mathcal{A}$ has is to guess b, compute z as $z = \mathsf{MsgRandExt}(m_b, \mathsf{Dec}_{\mathsf{sk}}(c''_b))$. We conclude that $\mathsf{Pr}[\mathcal{A} \text{ wins } G_4] = \frac{1}{2}$.

Based on the indistinguishability of transitions from the given game sequence, we conclude that if we repeat the protocols k times, the probability that $\mathcal{A}$ breaks the soundness is negligible and majored by:

$$\mathsf{Pr}[\mathcal{A} \text{ wins } G_0]^k \leq \left(2 \cdot (\epsilon_{\mathsf{rand}}(k) + \epsilon_{\mathsf{msgRand}}(k)) + \frac{1}{2}\right)^k.$$

(**Zero-Knowledge**). We define the simulator $\mathcal{S}(y)$ where $y = (\mathsf{pk}, c_0, c_1)$. The simulator $\mathcal{S}$ picks $b \xleftarrow{\$} \{0, 1\}$, computes $\mathsf{r} \xleftarrow{\$} \mathcal{R}, r_{\mathsf{m}} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}}$, $c'_b \leftarrow \mathsf{Rand}(\mathsf{pk}, c_b, r)$ and $c''_b \leftarrow \mathsf{MsgRandC}(c'_b, r_{\mathsf{m}})$. Finally, it returns (c''_b, b). The simulator acts as in the real protocol, so it perfectly simulates the proof.

Theorem 4. *Let Π be a PKE scheme, which is perfectly strong randomizable and derandomizable, perfectly message-randomizable and message-derandomizable and message-random-extractable. Let Γ be the commitment scheme, which is computationally binding and perfectly hiding. If Π and Γ are used in the protocol Π_{PEQ}, then Π_{PEQ} is complete, computationally sound and perfect zero-knowledge.*

Proof. (**Completeness**). If $\mathsf{Dec}_{\mathsf{sk}}(c_0) = \mathsf{Dec}_{\mathsf{sk}}(c_1)$, a message-randomization of both ciphertexts will decrypt to the same message m'. Hence, regardless of which ciphertext the prover receives from the verifier, if it is a message-randomization of either c_0 or c_1 and the scheme is message-random-extractable, the prover can correctly compute $z = r_{\mathsf{m}} = \mathsf{MsgRandExt}(m', m)$. It follows that the prover always opens a commitment for r_{m} so that the verifier accepts.
(**Soundness**). We define the following algorithm:

$\mathsf{GenInstance}(1^k, \mathcal{R}, \mathcal{M})$**:** It picks $(r, r_0, r_1) \xleftarrow{\$} \mathcal{R}^3$, $r_{\mathsf{m},0} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}}$, $r_{\mathsf{m},1} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}} \backslash \{r_{\mathsf{m},0}\}$ and $m \xleftarrow{\$} \mathcal{M}$, generates $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^k)$, $c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m; r)$, $c'_0 \leftarrow \mathsf{MsgRandC}(c, r_{\mathsf{m},0})$, $c'_1 \leftarrow \mathsf{MsgRandC}(c, r_{\mathsf{m},1})$, $c_0 \leftarrow \mathsf{Rand}(c'_0, r_0)$, $c_1 \leftarrow \mathsf{Rand}(c'_1, r_1)$, and returns $(\mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1)$.

Note that since c'_0 and c'_1 are message-randomizations of c with two different coins $r_{\mathsf{m},0} \neq r_{\mathsf{m},1}$, it holds that $\mathsf{Dec}_{\mathsf{sk}}(c'_0) \neq \mathsf{Dec}_{\mathsf{sk}}(c'_1)$ for any $(\mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1)$ returned by $\mathsf{GenInstance}$. We recall that for all $y \notin \mathcal{K} \times \mathcal{C}^2$, the verifier aborts the protocol. We deduce that for all instances y such that $y \notin L$ and $y \notin \mathcal{K} \times \mathcal{C}^2$, it holds that $Pr[\langle \mathcal{B}(x), \mathcal{V}(z) \rangle(y) = 1] = 0$ for any witness x and any bit-string z.

On the other hand, since the encryption scheme is perfectly randomizable, message-randomizable, derandomizable and message-derandomizable, the ciphertexts produced by the encryption algorithms on random messages follow the same distribution as the ones produced by the randomization and messages randomization algorithms, which implies that for all instances y such that $y \notin L$ and $y \in \mathcal{K} \times \mathcal{C}^2$, it holds that $y \in \{(\mathsf{pk}, c_0, c_1) | (\mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1) \leftarrow \mathsf{GenInstance}(1^k, \mathcal{R}, \mathcal{M})\}$, which means that the soundness of the protocol Π_{PEQ} can be proven by showing that for any witness x, any bit-string z and any instance $y = (\mathsf{pk}, c_0, c_1)$ generated from the output of $\mathsf{GenInstance}$, $Pr[\langle \mathcal{B}(x), \mathcal{V}(z) \rangle (y) = 1]$ is negligible.

We define an experiment that takes on input a witness x and a tuple $(\mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1)$ generated by $\mathsf{GenInstance}$, where an adversary $\mathcal{A}$ plays one round of the protocol Π_{PEQ} as a (dishonest) prover with a *challenger* that plays the role of an honest verifier. We define $\mathcal{A}$ as a triplet of p.p.t algorithms $(\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$:

$\mathcal{A}_1(x, y)$**:** This algorithm instantiates the dishonest prover. It returns a state st_1.

$\mathcal{A}_2(\mathsf{st}_1, c''_b)$**:** This algorithm corresponds to the first interaction between the verifier and the prover. It takes a challenge c''_b as input and returns a state st_2 and the response comm.

$\mathcal{A}_3(\mathsf{st}_2, (r, r_{\mathsf{m}}, b))$**:** This algorithm corresponds to the second interaction between the verifier and the prover. It takes a challenge (r, r_{m}, b) as input and returns a response op.

The experiment is defined as follows:

$\mathsf{Exp}^{\mathsf{Sound}}_{\mathcal{A}}(1^k,(x,\mathsf{sk},c,r_0,r_1,\mathsf{pk},c_0,c_1))$**:** compute $\mathsf{st}_1 \leftarrow \mathcal{A}_1(x,(\mathsf{pk},c_0,c_1))$, $b \xleftarrow{\$} \{0,1\}$, $r \xleftarrow{\$} \mathcal{R}$, $r_\mathsf{m} \xleftarrow{\$} \mathcal{R}_\mathsf{M}$, $c'_b \leftarrow \mathsf{Rand}(c_b,r)$, $c''_b \leftarrow \mathsf{MsgRandC}(c'_b,r_\mathsf{m})$, $(\mathsf{comm},\mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{st}_1,c''_b)$ and $\mathsf{op} \leftarrow \mathcal{A}_3(\mathsf{st}_2,(r,r_\mathsf{m},b))$. If $(\mathsf{Open}(\mathsf{comm},\mathsf{op}) \neq r_\mathsf{m})$ then return 0, else return 1.

In what follows, we will prove that for any adversary $\mathcal{A}$, the probability of winning this experiment is negligibly close to $1/2$ for any tuple $(x,\mathsf{sk},c,r_0,r_1,\mathsf{pk},c_0,c_1)$. Since for all instances y such that $y \notin L$ and $y \in \mathcal{K}\times\mathcal{C}^2$, it holds that $y \in \{(\mathsf{pk},c_0,c_1)|(\mathsf{sk},c,r_0,r_1,\mathsf{pk},c_0,c_1) \leftarrow \mathsf{GenInstance}(1^k,\mathcal{R},\mathcal{M})\}$, and since the protocol is repeated k times, we will deduce that for all instances y such that $y \notin L$ and $y \in \mathcal{K}\times\mathcal{C}^2$, it holds that $Pr[\langle\mathcal{B}(x),\mathcal{V}(z)\rangle(y)=1]$ is negligibly close to $1/2^k$, which means that the soundness probability is negligible.

We define a sequence of games (Figure 13) which are played between an adversary $\mathcal{A}$ and a challenger, where the first game G_0 is $\mathsf{Exp}^{\mathsf{Sound}}_{\mathcal{A}}(1^k,(x,\mathsf{sk},c,r_0,r_1,\mathsf{pk},c_0,c_1))$ for a fixed $(x,\mathsf{sk},c,r_0,r_1,\mathsf{pk},c_0,c_1)$. Considering the adversary's view, game G_0 represents a real execution of one round of the protocol Π_{PEQ}. We say that "$\mathcal{A}$ wins G_0" when the output is 1.

Game G_0	**Game** G_1
$\mathsf{st}_1 \leftarrow \mathcal{A}_1(x,(\mathsf{pk},c_0,c_1))$	$\mathsf{st}_1 \leftarrow \mathcal{A}_1(x,(\mathsf{pk},c_0,c_1))$
$b \xleftarrow{\$} \{0,1\}$	$b \xleftarrow{\$} \{0,1\}$
$r \xleftarrow{\$} \mathcal{R}$	$r' \xleftarrow{\$} \mathcal{R}$
$c'_b \leftarrow \mathsf{Rand}(c_b,r)$	$c'_b \leftarrow \mathsf{Rand}(c,r');r \leftarrow \mathsf{RandR}(r_b,r')$
$r_\mathsf{m} \xleftarrow{\$} \mathcal{R}_\mathsf{M}$	$r'_\mathsf{m} \xleftarrow{\$} \mathcal{R}_\mathsf{M}$
$c''_b \leftarrow \mathsf{MsgRandC}(c'_b,r_\mathsf{m})$	$c''_b \leftarrow \mathsf{MsgRandC}(c'_b,r'_\mathsf{m})$
	$r_\mathsf{m} \leftarrow \mathsf{MsgRandExt}(\mathsf{Dec}_\mathsf{sk}(c_b),\mathsf{Dec}_\mathsf{sk}(c''_b))$
$(\mathsf{comm},\mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{st}_1,c''_b)$	$(\mathsf{comm},\mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{st}_1,c''_b)$
$\mathsf{op} \leftarrow \mathcal{A}_3(\mathsf{st}_2,(r,r_\mathsf{m},b))$	$\mathsf{op} \leftarrow \mathcal{A}_3(\mathsf{st}_2,(r,r_\mathsf{m},b))$
if $(\mathsf{Open}(\mathsf{comm},\mathsf{op})=r_\mathsf{m})$ **return** 1	**if** $(\mathsf{Open}(\mathsf{comm},\mathsf{op})=r_\mathsf{m})$ **return** 1
else return 0	**else return** 0

Fig. 13: Sequence of games for Π_{PEQ}.

We define a game G_1 that proceeds as G_0 except that:

- We replace the instructions $r \xleftarrow{\$} \mathcal{R}$ and $c'_b \leftarrow \mathsf{Rand}(c_b,r)$ by $r' \xleftarrow{\$} \mathcal{R}$, $c'_b \leftarrow \mathsf{Rand}(c,r')$ and $r \leftarrow \mathsf{RandR}(r_b,r')$. We note that since the encryption scheme is strongly randomizable and derandomizable, and $c_b = \mathsf{Rand}(c,r_b)$, it holds that in G_1: $c = \mathsf{Rand}(c_b,r_b)$ and $c'_b = \mathsf{Rand}(c_b,\mathsf{RandR}(r_b,r')) = \mathsf{Rand}(c_b,r)$. On the other hand, since the encryption scheme is perfectly strongly randomizable, an element r produced by the sequence of instructions $r' \xleftarrow{\$} \mathcal{R};r \leftarrow \mathsf{RandR}(r_b,r')$ follows the same distribution as $r \xleftarrow{\$} \mathcal{R}$.
- We replace the instructions $r_\mathsf{m} \xleftarrow{\$} \mathcal{R}_\mathsf{M}$ and $c''_b \leftarrow \mathsf{MsgRandC}(c'_b,r_\mathsf{m})$ by $r'_\mathsf{m} \xleftarrow{\$} \mathcal{R}_\mathsf{M}$, $c''_b \leftarrow \mathsf{MsgRandC}(c'_b,r'_\mathsf{m})$ and $r_\mathsf{m} \leftarrow \mathsf{MsgRandExt}(\mathsf{Dec}_\mathsf{sk}(c_b),\mathsf{Dec}_\mathsf{sk}(c''_b))$. We note that since the encryption scheme is message-randomizable and message-random-extractable, the message encrypted in c''_b is indistinguish-

able form a message chosen at random in $\mathcal{M}$ and it holds that $c_b'' \leftarrow \mathsf{MsgRandC}($ $\mathsf{Rand}(c_b, r)$, $r_{\mathsf{m}})$ in both G_0 and G_1.

We deduce that $\mathsf{Pr}[\mathcal{A} \text{ wins } G_0] = \mathsf{Pr}[\mathcal{A} \text{ wins } G_1]$.

Next, we claim and prove by reduction that $\left|\mathsf{Pr}[\mathcal{A} \text{ wins } G_1] - \frac{1}{2}\right| = \frac{\epsilon_{binding}(k)}{2}$. We use the following strategy. In game G_1, the challenger can generate a random coin r for both $b = 1$ and $b = 0$ for the same challenge $c'' = c_0'' = c_1''$ because it builds c'' by computing $r' \xleftarrow{\$} \mathcal{R}$, $c' \leftarrow \mathsf{Rand}(c, r')$, $r'_{\mathsf{m}} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}}$, $c'' \leftarrow \mathsf{MsgRandC}(c', r'_{\mathsf{m}})$, $r \leftarrow \mathsf{RandR}(r_b, r')$ and $r_{\mathsf{m}} \leftarrow \mathsf{MsgRandExt}(\mathsf{Dec}_{\mathsf{sk}}(c_b), \mathsf{Dec}_{\mathsf{sk}}(c_b''))$. To break the protocol's soundness, the adversary must succeed for both cases $b = 1$ and $b = 0$ with non negligible probability at each round. If it is not the case, the adversary's advantage is bounded by a value that is negligibly close to $1/2^k$. We will show that if such an adversary exists, we can build an algorithm that breaks the commitment scheme's binding property. We show that if the adversary is able to succeed the proof for both cases $b = 1$ and $b = 0$, then he is able to open its commitment comm for two different message $z = 0$ and $z = 1$, which is equivalent to breaking the binding property.

We build an adversary $\mathcal{B}$ that has an advantage $\epsilon_{binding}(k)$ on the binding experiment as follows. $\mathcal{B}$ receives ck and simulates the game G_1 to $\mathcal{A}$ using ck as commitment key. More formally, $\mathcal{B}$ runs $\mathsf{st}_1 \leftarrow \mathcal{A}_1(x, (\mathsf{pk}, c_0, c_1))$ $r' \xleftarrow{\$} \mathcal{R}$, $\hat{r}_0 \leftarrow \mathsf{RandR}(r_0, r')$, $\hat{r}_1 \leftarrow \mathsf{RandR}(r_1, r')$, $c' \leftarrow \mathsf{Rand}(c, r')$, $r'_{\mathsf{m}} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}}$, $c'' \leftarrow \mathsf{MsgRandC}(c', r'_{\mathsf{m}})$, $\hat{r}_{\mathsf{m},0} \leftarrow \mathsf{MsgRandExt}(\mathsf{Dec}_{\mathsf{sk}}(c_0), \mathsf{Dec}_{\mathsf{sk}}(c''))$, $\hat{r}_{\mathsf{m},1} \leftarrow \mathsf{MsgRandExt}(\mathsf{Dec}_{\mathsf{sk}}(c_1), \mathsf{Dec}_{\mathsf{sk}}(c''))$, $(\mathsf{comm}, \mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{st}_1, c'')$, $\mathsf{op}_0 \leftarrow \mathcal{A}_3(\mathsf{st}_2, (\hat{r}_0, \hat{r}_{\mathsf{m},0}, b))$, $\mathsf{op}_1 \leftarrow \mathcal{A}_3(\mathsf{st}_2, (\hat{r}_1, \hat{r}_{\mathsf{m},1}), b)$. $\mathcal{B}$ sets $\mathsf{win}_0 = 1$ iff $(\mathsf{Open}(\mathsf{comm}, \mathsf{op}_0) = 0)$, and $\mathsf{win}_1 = 1$ iff $(\mathsf{Open}(\mathsf{comm}, \mathsf{op}_1) = 1)$.

If $\mathsf{win}_0 = 1$ and $\mathsf{win}_1 = 1$, then $\mathcal{B}$ returns $(\mathsf{comm}, \mathsf{op}_0, \mathsf{op}_1)$. Note that in this case, $\mathsf{Open}(\mathsf{comm}, \mathsf{op}_0) = 0 \neq 1 = \mathsf{Open}(\mathsf{comm}, \mathsf{op}_1)$, so $\mathcal{B}$ wins his experiment.

Using the same argument as in the soundness proof of Π_{PINEQ}, we can prove that:

$$\mathsf{Pr}[\mathcal{A} \text{ wins } G_0]^k = \mathsf{Pr}[\mathcal{A} \text{ wins } G_1]^k \leq \left(\frac{1 + \epsilon_{binding}(k)}{2}\right)^k.$$

Since the commitment scheme has the binding property by hypothesis, $\epsilon_{binding}(k)$ is negligible, so $\mathsf{Pr}[\mathcal{A} \text{ wins } G_0]^k$, which concludes the proof of the soundness.

(**Zero-Knowledge**). Let $\mathcal{V}^*$ be a dishonest verifier. In order to prove that the protocol is zero-knowledge, we define a Simulator $\mathcal{S}_{\mathcal{V}^*}(y)$ where $y = (\mathsf{pk}, c_0, c_1)$ that perfectly simulates the interaction with $\mathcal{V}^*$. The simulator $\mathcal{S}_{\mathcal{V}^*}$ generates c_b'', $\mathsf{r_b}$ and $\mathsf{r}_{\mathsf{m_b}}$ as the dishonest verifier $\mathcal{V}^*$.

- If $c_b'' = \mathsf{MsgRandCRand}((c_b, \mathsf{r_b}), \mathsf{r}_{\mathsf{m_b}})$ where $b \in \{0, 1\}$, then the simulator runs $(\mathsf{comm}, \mathsf{op}) \leftarrow \mathsf{Commit}(\mathsf{r}_{\mathsf{m_b}})$ and returns the transcript $(c_b', \mathsf{comm}, (\mathsf{r_b}, \mathsf{r}_{\mathsf{m_b}}, \mathsf{b}), \mathsf{op})$.
- If $c_b'' \neq \mathsf{MsgRandCRand}((c_b, \mathsf{r_b}), \mathsf{r}_{\mathsf{m_b}})$ where $b \in \{0, 1\}$, then the simulator picks comm in the uniform distribution on the commitment set and returns the transcript $(c_b', \mathsf{comm}, (\mathsf{r_b}, \mathsf{r}_{\mathsf{m_b}}, \mathsf{b}), \perp)$.

The simulator follows the same distribution as the real protocol. When the verifier sends a wrong pair $(\mathsf{r}, \mathsf{r}_{\mathsf{m}})$, it simulates the prover messages using a random commitment comm and the $\perp$ symbol. Indeed, since the commitment's open key

remains unknown, the prover commitment is like a random commitment picked in the uniform distribution according to the perfect hiding property.

Theorem 6. *Let Π be the PKE scheme used in Π_{MATCH}. If Π is perfectly randomizable, perfectly key-randomizable and key-derandomizable, then Π_{MATCH} is complete, special sound, and perfect zero-knowledge.*

Proof. (**Completeness**). Since $c \in \mathsf{Enc}_{\mathsf{pk}}(m)$, when the prover randomizes c into c'' after randomizing the keys, it holds that $c'' \in \mathsf{Enc}_{\mathsf{pk}}(m)$. It follows that when the verifier's challenge is $b = 1$, the verifier always accepts the proof. Similarly, when the verifier's challenge is $b = 0$, upon receiving the randomness used by the prover, the verifier can check that indeed $\widetilde{c}'' = c''$ and that $\widetilde{\mathsf{pk}}' = \mathsf{pk}'$, so it always accepts the proof as well.
(**Special Soundness**). Let the two following transcripts $t_0 = ((\mathsf{pk}', c''), 0, (r, r_K))$ and $t_1 = ((\mathsf{pk}', c''), 1, \mathsf{sk}'))$ for the statement $y = (\mathsf{pk}, c, m)$. We assume that t_0 and t_1 are transcripts of accepted proofs, *i.e.*,: $\mathsf{pk}' = \mathsf{KeyRandPk}(\mathsf{pk}, r_K)$, $c' = \mathsf{KeyRandC}(c, r_K)$, $c'' = \mathsf{Rand}(c', r)$ and $m = \mathsf{Dec}_{sk'}(c'')$. We define the knowledge extractor as follows: $\mathcal{E}(y, t, t')$ returns $\mathsf{sk} = \mathsf{KeyRandSk}(\mathsf{sk}', r_K^*)$. We first note that this algorithm is polynomial-time since computing r_K^* from r and running $\mathsf{KeyRandSk}$ are polynomial-time operations. In the following, we prove that $m = \mathsf{Dec}_{sk}(c)$. First, we have: $c'' = \mathsf{Rand}(c', r) \Rightarrow \mathsf{Dec}_{sk'}(c') = \mathsf{Dec}_{sk'}(c'')$. On the other hand, we have $\mathsf{sk} = \mathsf{KeyRandSk}(\mathsf{sk}', r_K^*)$, so $\mathsf{sk}' = \mathsf{KeyRandSk}(\mathsf{sk}, r_K)$, we deduce: $\mathsf{pk}' = \mathsf{KeyRandPk}(\mathsf{pk}, r_K) \wedge c' = \mathsf{KeyRandC}(c, r_K) \Rightarrow \mathsf{Dec}_{sk'}(c') = \mathsf{Dec}_{\mathsf{sk}}(c)$. Finally, we have: $m = \mathsf{Dec}_{sk'}(c'') = \mathsf{Dec}_{sk'}(c') = \mathsf{Dec}_{\mathsf{sk}}(c)$, which concludes the proof of the special soundness.
(**Zero-knowledge**). We define the simulator $\mathcal{S}(y)$ where $y = (\mathsf{pk}, c, m)$. The simulator $\mathcal{S}$ picks $b \xleftarrow{\$} \{0, 1\}$, then:

- If $b = 0$, the simulator picks $r \xleftarrow{\$} \mathcal{R}$, generated $(\mathsf{pk}', \mathsf{sk}') \xleftarrow{\$} \mathsf{KGen}(1^k)$ and runs $c'' \leftarrow \mathsf{Enc}_{\mathsf{pk}'}(m; r)$, then it returns the transcript $t = ((\mathsf{pk}', c''), 0, \mathsf{sk}'))$.
- If $b = 1$, the simulator picks $r_K \xleftarrow{\$} \mathcal{R}_K$ and $r \xleftarrow{\$} \mathcal{R}$, generates $\mathsf{pk}' \leftarrow \mathsf{KeyRandPk}(\mathsf{pk}, r_K); \mathsf{sk}' \leftarrow \mathsf{KeyRandPk}(\mathsf{sk}, r_K); c' \leftarrow \mathsf{KeyRandC}(c, r_K)$; and $c'' \leftarrow \mathsf{Rand}(c', r)$, then it returns $t = ((\mathsf{pk}', c''), 1, (r, r_K))$.

Note that in the case $b = 1$ the simulator follows the same steps as in the real protocol, so the protocol is perfectly simulated. To prove that the simulator follows the same distribution as the real protocol $\{\langle \mathcal{P}(x), \mathcal{V}^*(y) \rangle(y)\}$ when $b = 0$, we define a hybrid distribution $\mathcal{H}$ that runs the real protocol, except that it replaces the instruction $c'' = \mathsf{Rand}(c', r)$ by $c'' = \mathsf{Enc}_{\mathsf{pk}'}(m)$. First, we have that the real protocol is indistinguishable from the distribution $\mathcal{H}$ according to the definition of re-randomization. On the other hand, the distribution $\mathcal{H}$ is indistinguishable from the distribution of the simulator $\mathcal{S}(y)$. To show that, we argue that the only one difference between the two distribution is that $(\mathsf{sk}', \mathsf{pk}')$ are generated from the key generation algorithm in the simulator, and by key-randomization algorithms on $(\mathsf{sk}, \mathsf{pk})$ in $\mathcal{H}$, so the two distribution are indistinguishable according

to the key-randomization definition. Finally, we deduce that the real protocol and the simulator produce indistinguishable distribution.

Theorem 5. *Let Π be the PKE scheme used in Π_{MATCHPEQ}. If Π is perfectly randomizable and derandomizable, perfectly message-randomizable and message-derandomizable, and if the proof in step three is instantiated by a sigma protocol that is correct, special sound, and perfectly zero-knowledge, then Π_{MATCHPEQ} is complete, has statistical witness-extended emulation, and perfect zero-knowledge.*

Proof. (**Completeness**). Note that if c_1 and c_2 are both encryptions of the same message m, then c_1'' and c_2'' are both encryptions of m''. This is because m'' is a message-randomization of m and the same r_{m} is used to compute c_1'', c_2'' and m''. It follows that if the verifier's challenge is $b = 0$, the verifier accepts both proof and outputs accept. Similarly, if the challenge is $b = 1$, the verifier will obtain the same ciphertexts (it uses the same randomness) and so it outputs accept. We conclude that the verifier always accepts the proof.
(**Statistical witness-extended emulation**). As this proof system is not a sigma protocol because the verifier sends several challenges to the prover, it cannot be special sound. Hence, instead of special soundness, we prove the more general notion of statistical witness-extended emulation. To prove the statistical witness-extended emulation, we use the forking lemma given in [5]: let $\mathcal{T}$ be an accepted transcript tree for our protocol, and a statement y, *i.e.*, a tree where each node is labeled by a message transmitted during the protocol, each node labeled by a prover message has 2 children labeled by two different challenges, and any path from the root to any leaf is an accepted transcript. The forking lemma given in [5] ensures that if there exists an extractor $\mathcal{E}$ such that $Pr[x \leftarrow \mathcal{E}(y, \mathcal{T}) : (x, y) \in \mathcal{R}]$ is overwhelming, then our protocol is statistical witness-extended emulation. This can be viewed as a generalization of proofs for special soundness: each time the verifier sends a challenge, we fork the protocol for two possible challenges.

We parse $\mathcal{T}$ as $((c_1'', c_2''), (0, m'', \mathcal{T}'), (1, (r_{\mathsf{m}}, r_1, r_2)))$ where $\mathcal{T}'$ is a subtree that contains transcript trees of the two proofs $\mathsf{ZK}\{\mathsf{sk}_1 : \mathsf{Dec}_{\mathsf{sk}_1}(c_1'') = m''\}$ and $\mathsf{ZK}\{\mathsf{sk}_2 : \mathsf{Dec}_{\mathsf{sk}_2}(c_2'') = m''\}$. Note that we can extract two accepted transcripts for these two proofs from $\mathcal{T}'$ such that the two transcripts have the same commitment and two different challenges.

Since our protocol is instantiated with a special sound sigma protocol, there exists an extractor $\mathcal{E}'$ that takes theses transcripts in input and returns sk_1 and sk_2 such that $\mathsf{Dec}_{\mathsf{sk}_1}(c_1'') = m'' = \mathsf{Dec}_{\mathsf{sk}_2}(c_2'')$. Our simulator $\mathcal{E}$ runs $\mathcal{E}'$ with the appropriates transcripts, receives sk_1 and sk_2, returns $\mathsf{sk}_1, \mathsf{sk}_2$. We set $c_1' = \mathsf{MsgRandC}(c_1, r_{\mathsf{m}})$ and $c_2' = \mathsf{MsgRandC}(c_2, r_{\mathsf{m}})$. Since each path of the tree correspond to an accepted transcript, we have that $c_1'' = \mathsf{Rand}(c_1', r_1)$ and $c_2'' = \mathsf{Rand}(c_2', r_2)$. In the following, we prove that $\mathsf{Dec}_{\mathsf{sk}_1}(c_1) = \mathsf{Dec}_{\mathsf{sk}_2}(c_2)$. first, from message-randomizability, we have, for $i \in \{1, 2\}$: $c_i' = \mathsf{MsgRandC}(c_i, r_{\mathsf{m}}) \Rightarrow \mathsf{Dec}_{\mathsf{sk}_i}(c_i') = \mathsf{MsgRandM}(\mathsf{Dec}_{\mathsf{sk}_i}(c_i), r_{\mathsf{m}})$. Moreover, using re-randomizability of the encryption scheme, we have: $c_i'' = \mathsf{Rand}(c_i', r_i) \Rightarrow \mathsf{Dec}_{\mathsf{sk}_i}(c_i') = \mathsf{Dec}_{\mathsf{sk}_i}(c_i'')$. We

deduce that $\mathsf{Dec}_{\mathsf{sk}_i}(c_i'') = \mathsf{MsgRandM}\ (\mathsf{Dec}_{\mathsf{sk}_i}(c_i), r_\mathsf{m})$. Finally, since $\mathsf{Dec}_{\mathsf{sk}_1}(c_1'') = \mathsf{Dec}_{\mathsf{sk}_2}(c_2'')$, we have that: $\mathsf{MsgRandM}(\mathsf{Dec}_{\mathsf{sk}_1}(c_1), r_\mathsf{m}) = \mathsf{MsgRandM}(\mathsf{Dec}_{\mathsf{sk}_2}(c_2), r_\mathsf{m}) \Rightarrow \mathsf{MsgRandM}(\mathsf{MsgRandM}(\mathsf{Dec}_{\mathsf{sk}_1}(c_1), r_M), r_\mathsf{m}^*) = \mathsf{MsgRandM}(\mathsf{MsgRandM}(\mathsf{Dec}_{\mathsf{sk}_2}\ (c_2), r_M), r_\mathsf{m}^*) \Rightarrow \mathsf{Dec}_{\mathsf{sk}_1}(c_1) = \mathsf{Dec}_{\mathsf{sk}_2}(c_2)$. Which concludes the proof of the statistical witness-extended emulation.
(**Zero-knowledge**). We define the simulator $\mathcal{S}(y)$ where $y = (\mathsf{pk}_1, \mathsf{pk}_2, c_1, c_2, m)$. Since the two proofs used in the case $b = 0$ are zero-knowledge by hypothesis, there exist two simulators $\mathcal{S}_1$ and $\mathcal{S}_2$ that perfectly simulate the transcripts of these two proof systems. The simulator $\mathcal{S}$ picks $b \xleftarrow{\$} \{0, 1\}$, then:

- If $b = 0$, the simulator picks $m'' \xleftarrow{\$} \mathcal{M}$ and $(r_1'', r_2'') \xleftarrow{\$} \mathcal{R}^2$, generates $c_1'' \leftarrow \mathsf{Enc}_\mathsf{pk}(m''; r_1'')$ and $c_2'' \leftarrow \mathsf{Enc}_\mathsf{pk}(m''; r_2'')$, then it generates two transcripts t_1 and t_2 for $\mathsf{ZK}\{\mathsf{sk}_1 : \mathsf{Dec}_{\mathsf{sk}_1}(c_1'') = m''\}$ and $\mathsf{ZK}\{\mathsf{sk}_2 : \mathsf{Dec}_{\mathsf{sk}_2}(c_2'') = m''\}$ using the simulator $\mathcal{S}_1(pk_1, c_1'', m'')$ and $\mathcal{S}_2(pk_1, c_1'', m'')$.
- If $b = 1$, the simulator picks $r_\mathsf{m} \xleftarrow{\$} \mathcal{R}_M$ and $(r_1, r_2) \xleftarrow{\$} \mathcal{R}^2$, then it computes $c_1' \leftarrow \mathsf{MsgRandC}(c_1, r_\mathsf{m})$, $c_2' \leftarrow \mathsf{MsgRandC}(c_2, r_\mathsf{m})$, $c_1'' \leftarrow \mathsf{Rand}(c_1', r_1)$ and $c_2'' \leftarrow \mathsf{Rand}(c_2', r_2)$.

Note that in the case $b = 1$ the simulator follows the same steps as in the real protocol, so the protocol is perfectly simulated. To prove that the simulator follows the same distribution as the real protocol $\{\langle \mathcal{P}(x), \mathcal{V}^*(y)\rangle(y)\}$, we define a hybrid distribution $\mathcal{H}$ that runs the real protocol, except that it replaces the instructions $c_1'' \leftarrow \mathsf{Rand}(c_1', r_1)$ and $c_2'' \leftarrow \mathsf{Rand}(c_2', r_2)$ by $c_1'' \leftarrow \mathsf{Enc}_\mathsf{pk}(m''; r_1)$ and $c_2'' \leftarrow \mathsf{Enc}_\mathsf{pk}(m''; r_2)$ respectively. First, we have that the real protocol is indistinguishable from the distribution $\mathcal{H}$ according to the definition of re-randomization. On the other hand, the distribution $\mathcal{H}$ is indistinguishable from the distribution of the simulator $\mathcal{S}(y)$. To show that, we argue that the only one difference between the two distribution is that m'' is chosen at random in the simulator, and is generated by key-randomization algorithms on m using a random coin r_m in $\mathcal{H}$, so the two distribution are indistinguishable according to the message-randomization definition. Finally, we deduce that the real protocol and the simulator produce indistinguishable distribution, which concludes the proof.

Theorem 7. *Let Π be the PKE scheme used in Π_{SIGPEQ}. If Π is perfectly randomizable, perfectly message-randomizable and perfectly key-randomizable, then Π_{SIGPEQ} is complete, special sound, and perfect zero-knowledge.*

Proof. (**Completeness**). Since the scheme is perfectly randomizable, perfectly message-randomizable and key-randomizable the ciphertexts c_1''' and c_2''' will both decrypt using the key sk_1 (resp. sk_2) to the same message m', which is a randomization of m with r_m. We conclude that all the verifier needs to do is check the procedures following the same steps, and so it always accepts the proof.
(**Special soundness**). Let the two following transcripts $t_0 = ((\mathsf{pk}', c_1''', c_2'''), 0, (r_1, r_2, r_{K,1}, r_{K,2}, r_\mathsf{m}))$ and $t_1 = ((\mathsf{pk}', c_1''', c_2'''), 1, (\mathsf{sk}_1', \mathsf{sk}_2')$ for the statement $y =$

$(\mathsf{pk}_1, \mathsf{pk}_2, c_1, c_2)$. We assume that t_0 and t_1 are transcripts of accepted proofs, *i.e.*, computing:

- $\widetilde{c}'_1 \leftarrow \mathsf{Rand}(c_1, r_1)$ and $\widetilde{c}'_2 \leftarrow \mathsf{Rand}(c_2, r_2)$,
- $\widetilde{\mathsf{pk}}'_1 \leftarrow \mathsf{KeyRandPk}(\mathsf{pk}_1, r_{K,1})$ and $\widetilde{\mathsf{pk}}'_2 \leftarrow \mathsf{KeyRandPk}(\mathsf{pk}_2, r_{K,2})$,
- $\widetilde{c}''_1 \leftarrow \mathsf{KeyRandC}(\widetilde{c}'_1, r_{K,1})$ and $\widetilde{c}''_2 \leftarrow \mathsf{KeyRandC}(\widetilde{c}'_2, r_{K,2})$,
- $\widetilde{c}'''_1 \leftarrow \mathsf{MsgRandC}(\widetilde{c}''_1, \mathsf{r_m})$ and $\widetilde{c}'''_2 \leftarrow \mathsf{MsgRandC}(\widetilde{c}''_2, \mathsf{r_m})$

we have:

- $(\widetilde{c}'''_1 = c'''_1)$ and $(\widetilde{c}'''_2 = c'''_2)$,
- $(\widetilde{\mathsf{pk}}'_1 = \mathsf{pk}'_1)$ and $(\widetilde{\mathsf{pk}}'_1 = \mathsf{pk}'_1)$, and
- $(\mathsf{Dec}_{sk'_1}(c'''_1) = \mathsf{Dec}_{sk'_2}(c'''_2))$

We define the knowledge extractor as follows: $\mathcal{E}(y, t, t')$ returns $(\mathsf{sk}^*_1, \mathsf{sk}^*_2)$ such that $\mathsf{sk}^*_1 = \mathsf{KeyRandSk}(\mathsf{sk}'_1, r^*_{K,1})$ and $\mathsf{sk}^*_2 = \mathsf{KeyRandSk}(\mathsf{sk}'_2, r^*_{K,2})$. We first note that this algorithm is polynomial-time since computing $r^*_{K,1}$ and $r^*_{K,2}$from $r_{K,1}$ and $r_{K,2}$, and running $\mathsf{KeyRandSk}$ are polynomial-time operations. In the following, we prove that $\mathsf{Dec}_{sk^*_1}(c_1) = \mathsf{Dec}_{sk^*_2}(c_2)$. We set $m''' = \mathsf{Dec}_{sk^*_1}(c'''_1)$, then we have $m''' = \mathsf{Dec}_{sk^*_2}(c'''_2)$. We set $m''_1 = \mathsf{Dec}_{sk^*_1}(\widetilde{c}''_1)$ and $m''_2 = \mathsf{Dec}_{sk^*_2}(\widetilde{c}''_2)$ For all $i \in \{1, 2\}$, we have:

$$\widetilde{c}'''_i = \mathsf{MsgRandC}(\widetilde{c}''_i, \mathsf{r_m}) \Rightarrow m''' = \mathsf{Dec}_{sk'_i}(c'''_i) = \mathsf{MsgRandM}(m''_i, r_\mathsf{m})$$

We deduce that $\mathsf{MsgRandM}(m''_1, r_\mathsf{m})$=$\mathsf{MsgRandM}(m''_2, r_\mathsf{m})$,

and since $\mathsf{MsgRandM}(\,\cdot, r_\mathsf{m})$ is bijective, we have that $m''_1 = m''_2$. We set $m'' = m''_1$. Moreover, since $\mathsf{sk}^*_i = \mathsf{KeyRandSk}(\mathsf{sk}'_i, r^*_{K,i})$ we have $\mathsf{sk}'_i = \mathsf{KeyRandSk}(\mathsf{sk}^*_i, r_{K,i})$, so:

$$\widetilde{c}''_i = \mathsf{KeyRandC}(\widetilde{c}'_i, r_{K,i}) \Rightarrow m'' = \mathsf{Dec}_{sk'_i}(\widetilde{c}''_i) = \mathsf{Dec}_{sk^*_i}(\widetilde{c}'_i)$$

We deduce that $\mathsf{Dec}_{sk^*_1}(\widetilde{c}'_i) = \mathsf{Dec}_{sk^*_2}(\widetilde{c}'_2)$. Finally, we have:

$$\widetilde{c}'_i = \mathsf{Rand}(c_i, r_i) \Rightarrow m'' = \mathsf{Dec}_{sk^*_i}(\widetilde{c}'_i) = \mathsf{Dec}_{sk^*_i}(c_i)$$

We deduce that $\mathsf{Dec}_{sk^*_1}(c_1) = \mathsf{Dec}_{sk^*_2}(c_2)$, which conclude the proof of the special soundness.

(**Zero-knowledge**). We define the simulator $\mathcal{S}(y)$ where $y = (\mathsf{pk}_1, \mathsf{pk}_2, c_1, c_2, m)$. The simulator $\mathcal{S}$ picks $b \xleftarrow{\$} \{0, 1\}$, then:

- If $b = 0$, the simulator picks $m'' \xleftarrow{\$} \mathcal{M}$, generates $(r_1, r_2) \xleftarrow{\$} \mathcal{R}$ the keys $(\mathsf{pk}'_1, \mathsf{sk}'_1) \xleftarrow{\$} \mathsf{KGen}(1^k)$ and $(\mathsf{pk}'_2, \mathsf{sk}'_2) \xleftarrow{\$} \mathsf{KGen}(1^k)$, and computes $c'''_1 = \mathsf{Enc}_{\mathsf{pk}'_1}(m', r_1)$ and $c'''_2 = \mathsf{Enc}_{\mathsf{pk}'_2}(m', r_2)$.
  It then returns $((\mathsf{pk}'_1, \mathsf{pk}'_2, c'''_1, c'''_2), 0, (\mathsf{sk}'_1, \mathsf{sk}'_2))$.
- If $b = 1$, the simulator picks $(r_{K,1}, r_{K,1}) \xleftarrow{\$} \mathcal{R}^2_K$, $r_\mathsf{m} \xleftarrow{\$} \mathcal{R}_M$, $(r_1, r_2) \xleftarrow{\$} \mathcal{R}^2$ and generates $c'_1 \leftarrow \mathsf{Rand}(c_1, r_1)$, $c'_2 \leftarrow \mathsf{Rand}(c_2, r_2)$, $\mathsf{pk}'_1 \leftarrow \mathsf{KeyRandPk}(\mathsf{pk}_1, r_{K,1})$, $\mathsf{pk}'_2 \leftarrow \mathsf{KeyRandPk}(\mathsf{pk}_2, r_{K,2})$, $c''_1 \leftarrow \mathsf{KeyRandC}(c'_1, r_{K,1})$, $c''_2 \leftarrow \mathsf{KeyRandC}(c'_2, r_{K,2})$, $c'''_1 \leftarrow \mathsf{MsgRandC}(c''_1, r_\mathsf{m})$, and $c'''_2 \leftarrow \mathsf{MsgRandC}(c''_2, r_\mathsf{m})$.
  It then returns $((\mathsf{pk}'_1, \mathsf{pk}'_2, c'''_1, c'''_2), 1, (r_1, r_2, r_{K,1}, r_{K,2}, r_\mathsf{m}))$.

Note that in the case $b = 1$, the simulator follows the same steps as in the real protocol, so it is perfectly simulated. To prove that the simulator follows the same distribution as the real protocol $\{\langle \mathcal{P}(x), \mathcal{V}^*(y)\rangle(y)\}$, we define the following hybrid distributions:

- $\mathcal{H}_1$ that runs the real protocol, except that it replaces the instructions: $c'_1 \leftarrow \mathsf{Rand}(c_1, r_1)$ and $c'_2 \leftarrow \mathsf{Rand}(c_2, r_2)$ by: $c'_1 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m; r_1)$ and $c'_2 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m; r_2)$. The distribution of $\mathcal{H}_1$ is indistinguishable from the distribution induced by the real protocol. To show that, we argue that the only one difference between the two distribution is that:
  - **in the real protocol**, c'_1 and c'_2 are re-randomized
  - **in** $\mathcal{H}_1$ c'_1 and c'_2 are fresh ciphertexts of the same message m as c_1 and c_2 in $\mathcal{H}_1$

  So the two distribution are indistinguishable according to the (perfect) re-randomization definition.
- $\mathcal{H}_2$ that runs the same protocol as $\mathcal{H}_1$, except that it replaces the instructions: $\mathsf{pk}'_1 \leftarrow \mathsf{KeyRandPk}(\mathsf{pk}_1, r_{K,1})$, $\mathsf{sk}'_1 \leftarrow \mathsf{KeyRandPk}(\mathsf{sk}_1, r_{K,1})$, $\mathsf{pk}'_2 \leftarrow \mathsf{KeyRandPk}(\mathsf{pk}_2, r_{K,2}), \mathsf{sk}'_2 \leftarrow \mathsf{KeyRandPk}(\mathsf{sk}_2, r_{K,2}), c''_1 \leftarrow \mathsf{KeyRandC}(c'_1, r_{K,1})$ and $c''_2 \leftarrow \mathsf{KeyRandC}(c'_2, r_{K,2})$ by $(\mathsf{pk}'_1, \mathsf{sk}'_1) \xleftarrow{\$} \mathsf{KGen}(1^k)$, $(\mathsf{pk}'_2, \mathsf{sk}'_2) \xleftarrow{\$} \mathsf{KGen}(1^k)$, $c''_1 \leftarrow \mathsf{Enc}_{\mathsf{pk}'}(m; r_1)$, and $c''_2 \leftarrow \mathsf{Enc}_{\mathsf{pk}'}(m; r_2)$. The distribution of $\mathcal{H}_2$ is indistinguishable from the distribution of $\mathcal{H}_2$. To show that, we argue that the only one difference between the two distribution is that:
  - **in** $\mathcal{H}_1$ pk'_1, pk'_2, sk'_1, sk'_2, c''_1 and c''_2 are obtained by randomization of the key using the same coin r_K in $\mathcal{H}_1$
  - **in** $\mathcal{H}_2$ (i) $(\mathsf{pk}'_1, \mathsf{sk}'_1)$ and $(\mathsf{pk}'_1, \mathsf{sk}'_1)$ are fresh keys and (ii) c''_1 and c''_2 are obtain by using the same message m and the same coins (r_1, r_2) as c'_1 and c'_2 but using the fresh public keys pk'_1 and pk'_2

  So the two distribution are indistinguishable according to the (perfect) key-randomization definition.
- $\mathcal{H}_3$ that runs the same protocol as $\mathcal{H}_2$, except that it replaces the instructions: $c'''_1 \leftarrow \mathsf{MsgRandC}(c''_1, r_{\mathsf{m}})$ and $c'''_2 \leftarrow \mathsf{MsgRandC}(c''_2, r_{\mathsf{m}})$ by: $m' \xleftarrow{\$} \mathcal{M}$, $c'''_1 \leftarrow \mathsf{Enc}_{\mathsf{pk}'}(m'; r_1)$, and $c'''_2 \leftarrow \mathsf{Enc}_{\mathsf{pk}'}(m'; r_2)$. Note that $\mathcal{H}_3$ and $\mathcal{S}(y)$ induce the same distribution. Moreover, the distribution of $\mathcal{H}_3$ is indistinguishable from the distribution of $\mathcal{H}_2$. To show that, we argue that the only one difference between the two distribution is that:
  - **in** $\mathcal{H}_2$ m', c'''_1 and c'''_2 are obtained by randomizing the message in m, c''_1 and c''_2 using the same coin r_{m}
  - **in** $\mathcal{H}_3$ m' is a fresh message encrypted in c'''_1 and c'''_2 using the same public-key pk' and the same random coins (r_1, r_2) as in c''_1 and c''_2

  So the two distribution are indistinguishable according to the (perfect) message-randomization definition.

Theorem 8. *Let Π be the PKE scheme used in Π_{RSPEQ}. If Π is perfectly strong randomizable, random-extractable, perfectly message-randomizable and random coin decryptable, then Π_{RSPEQ} is complete, special sound, and perfect zero-knowledge.*

Proof. (**Completeness**). Since the scheme is perfectly strong randomizable, perfectly message-randomizable and random coin decryptable, the ciphertexts c_1'' and c_2'' will both decrypt using the key r_1'' (resp. r_2'') to the same message m', which is a randomization of m with r_{m}. We conclude that all the verifier needs to do is to check the procedure following the same steps, and so it always accepts the proof.
(**Special Soundness**). Let $t_0 = ((c_1'', c_2''), 0, (r_1'', r_2''))$ and $t_1 = ((c_1'', c_2''), 1, (r_1', r_2', \mathsf{r_M}))$ be two transcripts of accepted proofs for the statement $y = (\mathsf{pk}_1, \mathsf{pk}_2, c_1, c_2)$. We define the knowledge extractor $\mathcal{E}(y, t, t')$ as follows: it parses t_0 and t_1 and returns $r_1^* \leftarrow \mathsf{RandExt}(r_1', r_1'')$ and $r_2^* \leftarrow \mathsf{RandExt}(r_2', r_2'')$. Since the shceme is random-extractable and RCD-PKE we have that $\mathsf{CDec}_{\mathsf{r}_1^*}(c_1) = \mathsf{CDec}_{\mathsf{r}_2^*}(c_2)$ which concludes the proof of special soundness.
(**Zero-knowledge**). We define the simulator $\mathcal{S}(y)$ where $y = (\mathsf{pk}_1, \mathsf{pk}_2, c_1, c_2, m)$. The simulator $\mathcal{S}$ picks $b \xleftarrow{\$} \{0, 1\}$, then:

- If $b = 0$, the simulator picks $m' \xleftarrow{\$} \mathcal{M}$, $(r_1, r_2) \xleftarrow{\$} \mathcal{R}$, computes $c_1'' = \mathsf{Enc}_{\mathsf{pk}_1}(m', r_1)$ and $c_2'' = \mathsf{Enc}_{\mathsf{pk}_2'}(m', r_2)$. It then returns $((c_1'', c_2''), 0, (r_1, r_2))$.
- If $b = 1$, the simulator picks $r_{\mathsf{m}} \xleftarrow{\$} \mathcal{R}_M$, $(r_1', r_2') \xleftarrow{\$} \mathcal{R}^2$, and generates $c_1' \leftarrow \mathsf{Rand}(c_1, r_1')$, $c_2' \leftarrow \mathsf{Rand}(c_2, r_2')$, $c_1'' \leftarrow \mathsf{MsgRandC}(c_1', r_{\mathsf{m}})$, and $c_2'' \leftarrow \mathsf{MsgRandC}(c_2', r_{\mathsf{m}})$. It then returns $((c_1'', c_2''), 1, (r_1', r_2', r_{\mathsf{m}}))$.

Note that in the case $b = 1$, the simulator follows the same steps as in the real protocol, so the protocol is perfectly simulated. To prove that the simulator follows the same distribution as the real protocol $\{\langle \mathcal{P}(x), \mathcal{V}^*(y)\rangle(y)\}$, we define the following hybrid distribution:

- $\mathcal{H}_1$ that runs the real protocol, except that it replaces the instructions: $r_1'' \leftarrow \mathsf{RandR}(r_1, r_1')$, $r_2'' \leftarrow \mathsf{RandR}(r_2, r_2')$, $c_1' \leftarrow \mathsf{Rand}(c_1, r_1')$ and $c_2' \leftarrow \mathsf{Rand}(c_2, r_2')$ by: $c_1' \leftarrow \mathsf{Enc}_{\mathsf{pk}_1}(m; r_1'')$, $c_2' \leftarrow \mathsf{Enc}_{\mathsf{pk}_2}(m; r_2'')$. We argue that $\mathcal{H}_1$ is indistinguishable from the distribution induced by the real protocol. The only difference between both distributions is that **in the real protocol**, c_1' and c_2' are re-randomized with an r computed by RandR whereas **in** $\mathcal{H}_1$, c_1' and c_2' are fresh ciphertexts of the same message m (like c_1 and c_2). Such difference is indistinguishable according to the perfectly strong re-randomization definition.
- $\mathcal{H}_2$ that runs the same protocol as $\mathcal{H}_1$, except that it replaces the instruction: $c_1'' \leftarrow \mathsf{MsgRandC}(c_1', \mathsf{r_m})$ and $c_2'' \leftarrow \mathsf{MsgRandC}(c_2', \mathsf{r_m})$ by: $m' \xleftarrow{\$} \mathcal{M}$, $c_1'' \leftarrow \mathsf{Enc}_{\mathsf{pk}_1}(m'; r_1'')$, and $c_2'' \leftarrow \mathsf{Enc}_{\mathsf{pk}_2}(m'; r_2'')$. Note that $\mathcal{H}_2$ and $\mathcal{S}(y)$ induce the same distribution. Moreover, the distribution of $\mathcal{H}_2$ is indistinguishable from the distribution of $\mathcal{H}_1$. To show that, we argue that the only one difference between the two distribution is that:
  - **in** $\mathcal{H}_1$ m', c_1'' and c_2'' are obtained by randomizing the message m in c_1'' and c_2'' using the same $\mathsf{r_m}$.
  - **in** $\mathcal{H}_2$ m' is a fresh message encrypted in c_1'' and c_2'' using the random coins (r_1'', r_2'').

  It follows that the two distributions are indistinguishable according to the (perfect) message-randomization definition, which concludes the proof.