Watermarkable Public key Encryption With Efficient Extraction Under Standard Assumptions

Foteini Baldimtsi[1][0000−0003−3296−5336], Aggelos Kiayias[2][0000−0001−8501−3633], and Katerina Samari[3][0000−0001−7125−4524]

[1] George Mason University, Fairfax VA 22030, USA,
foteini@gmu.edu
[2] University of Edinburgh, Edinburgh EH8 9YL, UK and IOG,
aggelos.kiayias@ed.ac.uk
[3] Hypertech Energy Labs, Athens, Greece,
k.samari@hypertech.gr

Abstract. The current state of the art in watermarked public-key encryption schemes under standard cryptographic assumptions suggests that extracting the embedded message requires either linear time in the number of marked keys or the a-priori knowledge of the marked key employed in the decoder.
We present the first scheme that obviates these restrictions in the secret-key marking model, i.e., the setting where extraction is performed using a private extraction key. Our construction offers constant time extraction complexity with constant size keys and ciphertexts and is secure under standard assumptions, namely the Decisional Composite Residuosity Assumption [Eurocrypt'99] and the Decisional Diffie Hellman in prime order subgroups of square higher order residues.

Keywords: Watermarking · Public-key encryption

1 Introduction

Watermarking is a mechanism used to secure copyrighted material and counter the unauthorized distribution of digital content. In a high level, a watermarking scheme embeds a "mark" into a digital object and ensures that (a) the watermarked object is functionally equivalent to the original object (*functionality preserving*), and (b) it is difficult for an adversary to remove the mark without damaging the object (*unremovability*).

Recently, there has been an extensive line of research focusing in the special case of software watermarking with software being modeled as a Boolean circuit C. A software watermarking schemes consists of two main algorithms: the Mark algorithm that takes as input a circuit C and optionally a mark τ (for the case of message embedding watermarking) and outputs $\widetilde{C}$, and an Extract algorithm that takes as input a circuit C and outputs marked or not alongside with the mark τ if relevant.

The first rigorous study and formal definitions of software watermarking dates back to 2001 when Barak et al. [BGI+01, BGI+12] explored the relation between software watermarking and indistinguishability obfuscation (iO) and provided an impossibility result. In particular, they showed that if a marked circuit $\widetilde{C}$ has *exactly* the same functionality as the original, unmarked circuit C, then under the assumption that iO exists, watermarking is impossible. To overcome this impossibility result, a first line of work proposed schemes that are secure in restricted models where the allowed strategies for the unremovability adversary were limited [YF11, Nis13]. Later, [CHN+16, CHN+18] considered a more relaxed, in the statistical sense, variation of the functionality preserving property and propose a watermarking scheme for any family of *puncturable* pseudorandom functions (PRFs). Following the work of Cohen et al. [CHN+16], a long line of work has appeared in the literature focusing on watermarking PRFs under different models and assumptions ([KW17, BLW17, QWZ18, KW19, YAYX20]). Beyond watermarking schemes for PRFs, a number of works have focused on watermarking primitives such as encryption and signatures ([CHN+16, BKS17, GKM+19, YAL+19, Nis20]). This entails the topic of the present work.

Watermarking public key primitives. Cohen et al. [CHN+16, CHN+18], were the first to consider the notion of watermarking for the case of public-key cryptographic primitives. In particular, they define the notions of "Watermarkable Public-key Encryption" and "Watermarkable Signatures" making the important observation that the marking and key generation algorithms can be fused into one. Such primitives —watermarkable public-key encryption in particular— can be very useful to the enterprise setting, where multiple users belonging to the same organization use personal enterprise keys to access various services, such as a VPN. In such a setting, an organization may want to embed marks on the cryptographic algorithms used by employees and ensure that such marks can be extracted given any functioning decoder of one of the users. Cohen et al. [CHN+16, CHN+18], proceeded to describe how watermarkable schemes can be constructed by taking advantage of the work of Sahai and Waters [SW14], where public key encryption schemes and digital signature schemes are constructed based on iO. In the aforementioned constructions, a decryption or a signing algorithm is essentially an evaluation of a puncturable PRF. The idea of relying to the work of Sahai and Waters [SW14] for constructing watermarkable primitives has subsequently been utilized in [YAL+19]. Yang et al. [YAL+19] introduce the notion of *collusion-resistant watermarking*, meaning that an adversary is capable of receiving multiple watermarked copies of the same initial circuit embedded with different messages. The authors provide a watermarking scheme for PRFs and based on that and the constructions in [SW14] propose constructions for primitives like public key encryption, digital signatures, symmetric-key encryption and message-authentication codes.

The crucial question of whether watermarkable public key primitives can be constructed from *standard assumptions* was subsequently addressed by the works [BKS17], [GKM+19] and [Nis20] where different watermarking models are considered. Baldimtsi et al. [BKS17] mainly focus on watermarking existing

public key encryption schemes under minimal hardness assumptions and they achieve this for a more relaxed watermarking framework where a small public shared state is maintained between the marking and extraction algorithms. In their definitions, they follow a general approach in defining watermarking for public key primitives by distinguishing between the notions of watermarking of a given scheme (watermarking the implementation) versus constructing watermarkable instances of a public key primitive in the sense of [CHN+16, CHN+18].

Goyal et al. [GKM+19], revisit the notion of watermarkable public key primitives by considering stronger properties such as *public marking*, meaning that one can mark circuits as a public procedure, and *collusion-resistance*. They provide a construction for watermarkable signatures, as well as watermarkable constructions for more generalized notions for encryption, such as Attribute-based encryption (ABE) and Predicate Encryption (PE), by relying on standard assumptions. Regarding their watermarkable constructions for encryption primitives, even for the case of public key encryption, although they rely on the LWE assumption, they require heavy tools like Mixed Functional Encryption [GKW18] and Hierarhical Functional Encryption [BCG+17].

Nishimaki [Nis20] showed how to watermark existing public key cryptographic primitives under the condition that they have a canonical all-but-one (ABO) reduction, which is a standard technique usually employed for proving selective security. Nishimaki presents a general framework which shows how to transform a public key scheme with the above feature to a watermarked public key primitive by utilizing the simulation algorithms that appear in the proofs as the watermarked versions of the algorithms. Based on this novel idea, they provide watermarking constructions for IBE, IPE, ABE which are secure under the same assumptions as the underlying primitives. We note that a selective secure variant for watermarking definitions is employed in [Nis20].

One of the key issues in terms of the efficiency of watermarkable primitives is the complexity of the extraction algorithm, i.e., the steps required to extract (or detect) the embedded mark from a circuit. Ideally, the process of extraction should be independent (or at least polylogarithmic) in the number of marked programs. This is particularly crucial when in the underlying application extraction is time sensitive. For instance, for the application of tracing unauthorized distribution of digital content there is often the need to immediately identify (and potentially revoke) malicious users. Thus, if one assumes that the mark is some identifiable user information, it is important to be able to extract efficiently.

Unfortunately, all previous approaches that provided constructions under standard assumptions require a linear running time for this step. The linear overhead either comes directly, e.g., in [BKS17] where the extraction algorithm has to re-generate one-by-one all the marked public keys in order to decide whether a circuit is marked or not, or indirectly by requiring the marked public-key as separate input to the extraction algorithm (an approach followed by [GKM+19], [Nis20]). In the latter case, this is due to the fact that the extraction algorithm needs to traverse *all* marked public keys in order to determine whose associated decryption circuit is being detected (we discuss this in more details in Section 1.3).

Relying on non-standard assumptions, such as iO, is the only known approach so far to allow for efficient extraction [CHN+16] without knowledge of the marked public-key.

Motivated by the above, in this work, we study the following question:

Can we build efficient watermarkable public key encryption schemes under standard assumptions where the extraction algorithm is sublinear or even constant in the number of marked programs, without relying on the knowledge of the key in the given decoder?

We answer this question in the affirmative.

1.1 Our Contribution

We present a concrete watermarkable PKE scheme with the following features: (1) The running time of the extraction algorithm is **independent** of the number of generated marked circuits (decryption circuits in the case of PKE) and does not require knowledge of the marked key, (2) the ciphertexts, the public and secret keys have all **constant size** in all salient parameters exhibiting only a linear dependency in the security parameter.

Our construction is an El-Gamal like scheme that shares features with Paillier encryption and whose security relies on the Decisional Composite Residuosity DCR assumption [Pai99] as well the $\mathsf{DDH}_{\mathsf{SQNR}}$ assumption [KTY07]. The latter assumption is simply the Decisional Diffie Hellman assumption over prime order subgroups of the group of n-th residues modulo n^2, where n is an RSA modulus as in Paillier encryption. We note that residuosity and discrete-logarithm related assumptions over modular arithmetic groups have been studied extensively and are considered standard assumptions compared to more recent cryptographic assumptions such as those needed to obtain iO. We provide a technical overview of our construction in Section 1.2 .

Our scheme is proven secure in the secret-marking model, under the definitional framework of [CHN+16, BKS17] where there exists a single marking algorithm responsible for both key-generation of the public-key encryption scheme and marking at the same time (i.e., a mark is embedded into a circuit when generating public and secret encryption keys). We provide a more detailed discussion of our model and how it compares to related work in Section 1.3.

1.2 Technical Overview of our Construction

Our main goal is to construct a watermarkable PKE scheme with an extraction algorithm that is independent of the number of previously marked decryption circuits. In a typical watermarked PKE scheme, the marked object is a decryption circuit C. As noted above, in all previous works in order to detect whether C is marked or not, the extraction algorithm had to either re-generate and test all marked public keys [BKS17] or require the marked public-key as separate input to the extraction algorithm [GKM+19, Nis20]).

To avoid this linear dependency to the number of public keys, we need to take a very different extraction approach. Our starting point is a PKE with the following property: it should be possible to generate a probability distribution in the ciphertext space in a manner independent of the public-key. At the same time, when such ciphertexts are sampled and given as input to a decryption circuit C, it should be possible to extract information about the decryption key hidden inside the decoder. Assume for example that by giving such a ciphertext as input to C, one can reconstruct a public key indicating that the corresponding secret key was used to decrypt the ciphertext. However, deciding whether C is marked or not would still require to check whether the reconstructed public key is one of the keys that had been previously marked by the marking service. Thus, we additionally require the PKE to allow the embedding of some authenticated information as part of the secret and public keys, which can only be recovered by using the private extraction key.

Consider the El-Gamal PKE scheme with public parameters $(\mathbb{G}, q, g)$ where $\mathbb{G}$ is a cyclic group of prime order q and g is a generator of that group. Assume a public-secret key pair (g^x, x) where $x \xleftarrow{\$} \mathbb{Z}_q$. An encrypted plaintext m is of the form $(g^r, g^{rx} \cdot m)$, where $r \xleftarrow{\$} \mathbb{Z}_q$, is indistinguishable from a random pair $(g^r, g^{r'})$, where $r, r' \xleftarrow{\$} \mathbb{Z}_q$. If $(g^r, g^{r'})$ is fed to a decryption algorithm under the secret key $sk = x$ the result would be $\mathsf{Dec}(x, ((g^r, g^{r'})) = g^{r'-rx}$. Given we have chosen r, r', it is possible to apply a simple calculation over $g^{r'-rx}$ to extract the public key g^x. Therefore, assuming a circuit C as black-box, if one runs it on input the pair $(g^r, g^{r'})$ and then performs the computation described above, one can deduce the public key (if the circuit indeed uses a single public key). With respect to our objective however this approach is still too weak: since we bound the extraction algorithm to be independent of the number of marked circuits, having recovered the public-key cannot help us detect whether the key was indeed marked or not. Here comes our second technical observation: we can take advantage of the partial discrete logarithm trapdoor of Paillier's encryption scheme to turn the public-key of a marked scheme to something that the marking algorithm can "decrypt" and identify some internal property of it to assist the extraction algorithm.

To accomplish this plan, the first cryptographic design challenge is to create this hybrid encryption scheme, that behaves like ElGamal from the perspective of the user, while its public-key is akin to ciphertext for a Paillier-like encryption whose secret-key is used during the extraction process. This means that the El-Gamal variant has to operate within $\mathbb{Z}^*_{n^2}$ and specifically within a suitable order subgroup $\langle g_1 \rangle$ where DDH is expected to hold, while the public-key will belong to a suitable coset $\langle g_1 \rangle \cdot (1+n)^v$ with v containing the salient features the extraction algorithm can recover via Paillier decryption. The second design challenge, is that the extraction algorithm needs to check the integrity of the recovered public-key; this is done by incorporating into v a message authentication code (actually a PRF) that tags the ElGamal component of the public-key. The third design challenge is to be able to embed robustly an arbitrary mark within the

public-key without disturbing the functional properties of the public-key itself. To achieve that we utilize an authenticated symmetric encryption that extends the value v with a ciphertext that contains the mark.

The final challenge is to ensure that decryption from the user side works similarly to ElGamal and all the attributes that were inserted by the above requirements into the public-key do not disturb the decryption operation of the PKE.

1.3 Relations to Prior Work

In the first part of the introduction we covered related work in watermarkable cryptographic primitives as well as watermarking in general. In this section we provide a more extensive discussion on how our work differs from related work in terms of definitional model, security properties and efficiency. We focus our attention on related work for watermarkable public key primitives. Finally, we also discuss the relation to traitor tracing.

Definitional Model. In this work, we adopt a model that is based on the definitional framework proposed by Cohen et al. [CHN+16]. Namely, we define a watermarkable PKE scheme where the Mark algorithm is responsible for both generating a key pair instance (pk, sk) for the encryption scheme, as well as marking it at the same time.

Other works [YAL+19, GKM+19, Nis20] impose a stronger requirement where the key generation and marking algorithms are decoupled: (unmarked) keys can be independently generated and, subsequently, marked. For completeness, we point out some differences between the models considered in the aforementioned works. First of all, in contrast to all other works, the model of Goyal et al. [GKM+19] considers both public marking and public extraction. These properties are satisfied by the watermarkable predicate encryption scheme presented in the same paper. The model of [Nis20] considers secret marking and secret extraction while [YAL+19] considers public extraction and secret marking.

We note that the model of coupling key generation and marking is natural in certain applications (i.e. watermarking a VPN client). Most importantly though, coupling key generation and marking *does not trivialize the problem*: If one considered a trivial watermarking scheme where the marking authority simply associates every secret key with the identity of the owner, it will have to store one associated key with each identity. This is completely inefficient, since it results in a stateful watermarkable encryption scheme with both linear state and linear extraction time.

Security Properties. We require watermarkable PKE schemes to preserve IND-CPA security and also support unremovability and unforgeability. In our unremovability definition we consider adversaries that have access to Challenge, Corrupt and Extract oracles. This is a stronger definition than the one considered in [CHN+16]. An important point when defining unremovability is in respect to when we consider that the adversary successfully removed the mark.

Cohen et al. required the adversarial circuit to agree with a decryption circuit on a fraction of inputs. However, as pointed in [GKM+19], this can lead to trivial attacks. We follow the definitional approach of [GKM+19] who in order to address this issue, they require that an adversary should output a "useful" circuit which is at the same time unmarked (or marked with a different than the original message). We capture the notion of "useful" by defining closeness and farness relations to capture the adversarys ability to create circuits that are close or far to marked ones.

Since we focus on the secret-marking setting we also consider unforgeability which guarantees that an adversary cannot create a marked circuit with sufficiently different functionality from that of given marked circuits without access to a marking key. Similar to unremovability, we define unforgeability for adversaries that have access to Challenge, Corrupt and Extract oracles. The constructions of [CHN+18] do not support unforgeability. The more recent work of [Nis20] describes a possible direction to achieve unforgeability. Unforgeability does not make sense as a property in the public marking setting of [GKM+19].

In addition, [YAL+19, GKM+19] consider the notion of collusion-resistance, or else Collusion-resistance w.r.t. watermarking (as coined in [Nis20]). A watermarking scheme is collusion-resistant w.r.t. watermarking if it is unremovable even if adversaries are given many watermarked keys for the same original key. We note that the collusion-resistance property cannot be inherently considered in our model since the user does not choose the initial key and request marked versions embedded with different messages. Also, as noted in [Nis20], this property is not crucial for classic applications of watermarking such as ownership identification.

Efficiency. In this paragraph, we present several efficiency parameters of previous watermarkable public key encryption schemes, in particular [Nis20] and [GKM+19], and provide a comparison with the construction presented in this work. Starting with [GKM+19], a watermarkable predicate encryption scheme can be instantiated from hierarchical functional encryption scheme, which, in turn, according to [AV19] can be constructed from any PKE scheme. In such a construction, the size of ciphertexts is linear in the number of colluding users. In [Nis20],the efficiency parameters of the initial public key encryption scheme are almost preserved. Specifically, the ciphertext size does not change, while the public and secret keys have also a linear dependence on the size of the embedded message and the security parameter. Regarding the complexity of the extraction algorithm, in both [GKM+19, Nis20] we note that the master public key (which is the public key of a user in the case of Watermarkable PKE) is part of the input of the extraction algorithm. This is particularly limiting in applications where one might detect the use of marked circuit in the wild (i.e. a stolen decryption circuit). Excluding the public key from the input of the Extract algorithm, as in our model, implies that Extract would have to search over all the public keys generated so far. The construction presented in this work (cf. Section 4) achieves constant size ciphertexts, constant size public-secret keys and the extraction time is independent of the number of generated marked keys.

Relation to Traitor-Tracing. Watermarkable PKE is also related to the notion of traitor tracing, which was put forth by Chor et al. [CFN94] and studied extensively (see e.g. [KY02, DFKY03, BW06, BSW06, BZ14] and references therein). Briefly speaking, in traitor tracing, an authority delivers keys to a set of users and encrypts content which is intended to be decrypted by all users, or a subset of them in cases where an authority is capable of revoking decryption keys (i.e. trace and revoke schemes). In the occasion where a number of users collude by constructing an even partially working implementation of the decryption function, the authority can identify at least one of the colluding users.

Previous works ([GKM+19, YAL+19, Nis20]), point to the relation between traitor tracing and watermarkable encryption. In particular, Goyal et al. [GKM+19] present a generic construction which shows how to obtain a watermarkable PKE scheme (in the sense of Cohen et al. definition [CHN+16]) from a traitor tracing scheme with embedded identities and a regular public-key encryption. We note that a traitor tracing scheme with embedded identities (cf. [GKW19, NWZ16]) is an extension of the standard traitor tracing where not only the index of a colluding user is traced, but a whole string is extracted, i.e. the identity. Based on this, [GKM+19] argue that "in the particular case where we allow a single algorithm that both generates the public/secret keys together with the watermark" the notion of watermarkable PKE would be entirely subsumed by traitor tracing. While valid, this observation sidesteps the serious disadvantage that the generic construction blows up the complexity of the watermarkable PKE to be at least as much as underlying traitor tracing. As a result, the state of the art in traitor tracing cannot yield efficient watermarkable PKE under standard assumptions, to the best of our knowledge (even the most efficient construction in terms of keys and ciphertexts from [GKW19], which is provable under LWE, will require an extraction algorithm linear in the number of users). Moreover, watermarking seems a simpler primitive than traitor tracing since in the latter, users should share the decryption functionality and apply it to the same ciphertext, while in watermarking users functionalities are entirely decoupled. In our view, this is the key issue in the design of watermarking PKE schemes and we demonstrate that we can obtain constructions where the size of ciphertexts, the size of keys, as well as the extraction complexity is independent of the number of users.

2 Preliminaries

Notation. By λ we denote the security parameter and by $\mathsf{negl}(\lambda)$ we denote a negligible function in λ. By $x||y$, we denote the concatenation of the bitstrings x, y. By $x \xleftarrow{\$} S$ we denote that x is sampled uniformly at random from S. By $\mathsf{poly}(\lambda)$ we denote a polynomial in λ. In addition, we write $\mathcal{D}_1 \overset{c}{\approx} \mathcal{D}_2$ to denote that the distributions $\mathcal{D}_1$, $\mathcal{D}_2$ are computationally indistinguishable.

Assumptions. We first introduce the assumptions which will be necessary for proving the properties of our watermarkable PKE scheme.

Definition 1 (DCR assumption [Pai99]). *No PPT adversary can distinguish between: (i) tuples of the form* $(n, u^n \mod n^2)$*, where* n *is a composite RSA modulus and* $u \xleftarrow{\$} \mathbb{Z}^*_{n^2}$ *and (ii) tuples on the form* (n, v)*, where* $v \xleftarrow{\$} \mathbb{Z}^*_{n^2}$*.*

Definition 2 (DDH for square n-th residues [KTY07]). *Let* n *be a composite RSA modulus, i.e.* $n = pq$*, where* $p = 2p' + 1$*,* $q = 2q' + 1$ *and* p', q' *are also primes. By* $\mathcal{X}_{n^2}$ *we denote the subgroup of* $\mathbb{Z}_{n^2}$ *that contains all square* n*-th residues. The Decisional Diffie Hellman assumption for square* n*-th residues (*$\mathsf{DDH}_{\mathsf{SQNR}}$*) is defined as follows: The distribution* $\langle n, g_1, y, g_1^r, y^r \rangle$*, where* g_1 *generates* $\mathcal{X}_{n^2}$*,* $y \xleftarrow{\$} \mathcal{X}_{n^2}$ *and* $r \xleftarrow{\$} [p'q']$*, is computationally indistinguishable from the distribution* $\langle n, g_1, y, g_1^r, y^{r'} \rangle$*, where* g_1, y, r *defined as above and* $r' \xleftarrow{\$} [p'q']$*.*

In our construction we use a seemingly stronger variant of the above assumption where the factorization of n is also provided as part of the tuples. While this appears to provide more power to the adversary, it is straightforward to see that it reduces, via the Chinese remainder theorem, to the two DDH assumptions for the underlying subgroups $\langle g_1^{q'} \rangle$ and $\langle g_1^{p'} \rangle$ of prime order p', q' respectively.[4] For simplicity and due to its relation to the DDH assumptions in the underlying groups, we will use the same notation $\mathsf{DDH}_{\mathsf{SQNR}}$ to denote this variant.

Cryptographic Primitives. For completeness we recall the definitions of some cryptographic primitives to be used below.

Definition 3 (Pseudorandom function). *Let* $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ *be a keyed function with key space* $\mathcal{K}$*, input space* $\mathcal{X}$ *and output space* $\mathcal{Y}$*. We say that* F *is a pseudorandom function (PRF) if for any PPT distinguisher* D *it holds that*

$$\left| \mathsf{Pr}_{k \xleftarrow{\$} \mathcal{K}}[D^{F(k,\cdot)}(1^n) = 1] - \mathsf{Pr}_{f \xleftarrow{\$} \mathcal{F}}[D^{f(\cdot)}(1^n) = 1] \right| \leq \mathsf{negl}(\lambda),$$

where $\mathcal{F}$ *is the set of all functions with input space* $\mathcal{X}$ *and output space* $\mathcal{Y}$ *and* $\mathcal{X} = \{0,1\}^n$*.*

We consider an authenticated symmetric encryption scheme that satisfies the notion of *integrity of ciphertexts* as defined in [BN00], i.e. an adversary that only has access to a signing oracle, cannot produce any fresh valid ciphertext.

[4] To see this, consider A DDH challenge for the two underlying groups $\langle g_1^{q'}, y_\mathsf{l}, G_\mathsf{l}, Y_\mathsf{l} \rangle$ $\langle g_1^{p'}, y_\mathsf{r}, G_\mathsf{r}, Y_\mathsf{r} \rangle$, we can combine them to $\langle g_1, y_\mathsf{l} \cdot y_\mathsf{r}, G_\mathsf{l} \cdot G_\mathsf{r}, Y_\mathsf{l}^{q'} \cdot Y_\mathsf{r}^{p'} \rangle$. Observe that if the challenge pair is DDH distributed then $G_\mathsf{l} \cdot G_\mathsf{r} = g_1^{q' r_\mathsf{l} + p' r_\mathsf{r}}$ and $Y_\mathsf{l}^{q'} \cdot Y_\mathsf{r}^{p'} = y_\mathsf{l}^{q' r_\mathsf{l}} y_\mathsf{r}^{p' r_\mathsf{r}} = g_1^{(q')^2 t_\mathsf{l} r_\mathsf{l} + (p')^2 t_\mathsf{r} r_\mathsf{r}}$. Now observe that $(q' t_\mathsf{l} + p' t_\mathsf{r})(q' r_\mathsf{l} + p' r_\mathsf{r}) = (q')^2 t_\mathsf{l} r_\mathsf{l} + (p')^2 t_\mathsf{r} r_\mathsf{r} \bmod p'q'$. Given that $y_\mathsf{l} \cdot y_\mathsf{r} = g_1^{q' t_\mathsf{l} + p' t_\mathsf{r}}$, this establishes that the combined challenge is DDH distributed. For the other case, when the challenge pair follows the random distribution, then $Y_\mathsf{l}^{q'} \cdot Y_\mathsf{r}^{p'} = y_\mathsf{l}^{q' r'_\mathsf{l}} y_\mathsf{r}^{p' r'_\mathsf{r}} = g_1^{(q')^2 t_\mathsf{l} r'_\mathsf{l} + (p')^2 t_\mathsf{r} r'_\mathsf{r}}$ that can be easily seen to be uniformly distributed over $\mathcal{X}_{n^2}$ and as a result the combined challenge is randomly distributed.

Definition 4. *A symmetric encryption scheme* $(\mathsf{S.Gen}, \mathsf{S.Enc}, \mathsf{S.Dec})$ *satisfies integrity of ciphertexts if for any PPT adversary* $\mathcal{A}$ *it holds that*

$$\mathsf{Pr}[\mathcal{G}_{\mathcal{A}}^{\mathsf{int-ctxt}}(1^\lambda) = 1] = \mathsf{negl}(\lambda).$$

$\underline{\mathcal{G}_{\mathcal{A}}^{\mathsf{int-ctxt}}(1^\lambda):}$

1. The Challenger $\mathcal{C}$ generates a key $k_{\mathsf{e}} \leftarrow \mathsf{S.Gen}(1^\lambda)$. Then initializes the set $S \leftarrow \emptyset$, which will include the ciphertexts that will be returned as answers to $\mathcal{A}$'s queries.
2. $\mathcal{A}$ is allowed to issue a number of queries to the encryption oracle $\mathsf{S.Enc}(k_{\mathsf{e}}, \cdot)$. Upon receiving a message m from $\mathcal{A}$, $\mathcal{C}$ computes $c \leftarrow \mathsf{S.Enc}(k_{\mathsf{e}}, m)$, returns c to $\mathcal{A}$ and sets $S \leftarrow S \cup \{c\}$.
3. $\mathcal{A}$ wins, if there is c' issued as a query to the decryption oracle $\mathsf{S.Dec}(k_{\mathsf{e}}, \cdot)$ s.t. $c' \notin S$ and $m' \neq \perp$ where $m' \leftarrow \mathsf{S.Dec}(k_{\mathsf{e}}, c')$. Then the game returns 1.

Fig. 1: The integrity of ciphertexts game.

Next, we state the definition of real-or-random CPA security for symmetric encryption [BDJR97]. In a high level, an adversary should not be able to distinguish a ciphertext from a random string from the ciphertext space.

Definition 5. *A symmetric encryption scheme* $(\mathsf{S.Gen}, \mathsf{S.Enc}, \mathsf{S.Dec})$ *with plaintext space* $\{0,1\}^\nu$ *and ciphertext space* $\{0,1\}^\mu$ *satisfies real-or-random security against chosen plaintext attacks if for any PPT adversary* $\mathcal{A}$*,*

$$\mathsf{Pr}[\mathcal{G}_{\mathcal{A}}^{\mathsf{ror-cpa}}(1^\lambda) = 1] = \mathsf{negl}(\lambda).$$

$\underline{\mathcal{G}_{\mathcal{A}}^{\mathsf{ror-cpa}}(1^\lambda):}$

1. The Challenger $\mathcal{C}$ generates a key $k_{\mathsf{e}} \leftarrow \mathsf{S.Gen}(1^\lambda)$.
2. $\mathcal{A}$ is allowed to issue queries to the encryption oracle, i.e. $\mathcal{A}$ issues a query m to $\mathcal{C}$, and $\mathcal{C}$ responds to $\mathcal{A}$ with c s.t. $c \leftarrow \mathsf{S.Enc}(k_{\mathsf{e}}, m)$.
3. Challenge phase: $\mathcal{A}$ sends a message m to the challenger. Then $\mathcal{C}$ chooses $b \xleftarrow{\$} \{0,1\}$. If $b = 0$ it computes $c^* \leftarrow \mathsf{S.Enc}(k_{\mathsf{e}}, m)$, otherwise it chooses $c^* \xleftarrow{\$} \{0,1\}^\mu$. $\mathcal{C}$ returns c^* to $\mathcal{A}$.
4. $\mathcal{A}$ outputs b^*. If $b = b^*$ then $\mathcal{A}$ wins and the game returns 1.

Fig. 2: Real or Random CPA Security.

3 Watermarkable Public Key Encryption

We start by introducing the definition for a watermarkable public-key encryption scheme. Our definition follows the framework of [CHN+16, BKS17] by considering a single algorithm, $\mathsf{PKE.Mark}$ responsible for both key-generation of the public-key encryption scheme and marking at the same time (i.e., a mark is embedded into a circuit when generating public and secret encryption keys).

Definition 6 (Watermarkable PKE). *A watermarkable public key encryption scheme with message space $\mathcal{M}$, tag space $\mathcal{T}$ and ciphertext space $\mathcal{CT}$ is a tuple of algorithms* $(\mathsf{WM.Gen}, \mathsf{PKE.Mark}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Extract})$ *with the following syntax:*

- $\mathsf{WM.Gen}(1^\lambda) \to (params, mk, xk)$*: On input the security parameter λ, it outputs the public parameters $params$, a marking key mk, and an extraction key xk. The marking key is private and it is kept by an authority, while the extraction key may be either public or private depending on whether the scheme allows public or private extraction.*
- $\mathsf{PKE.Mark}(params, mk, \tau) \to (pk, sk)$*: On input the marking key mk, $params$, and a tag $\tau \in \mathcal{T}$, it outputs a public-secret key pair (pk, sk).*
- $\mathsf{Enc}(pk, m) \to c$*: On input a public key pk and a plaintext $m \in \mathcal{M}$, it outputs a ciphertext c.*
- $\mathsf{Dec}(sk, c) \to m$*: On input a secret key sk and ciphertext c, it outputs a plaintext m.*
- $\mathsf{Extract}(params, xk, C) \to \tau/\perp$*: On input xk, $params$ and a circuit C which maps $\mathcal{CT}$ to $\mathcal{M}$, it outputs a tag $\tau' \in \mathcal{T}$ or returns a special symbol $\perp$, indicating that the circuit is unmarked.*

Remark 1. We say that a watermarkable PKE scheme supports *public marking* if the marking key is equal to the public watermarking parameters are $params = mk$. Otherwise we say that it only supports *private marking*. Similarly, we say that a scheme supports *public extraction* if $params = xk$ and *private extraction* otherwise. In this work we focus in the private setting.

We now define correctness. In the definitions below, Dec_{sk} denotes a decryption circuit with secret key sk embedded.

Definition 7 (Extraction correctness). *We say that a watermarkable PKE scheme satisfies extraction correctness if for any tag $\tau \in \mathcal{T}$, it holds that:*

$$\mathsf{Pr}\left[\mathsf{Extract}(xk, \mathsf{Dec}_{sk}) \neq \tau \,\middle|\, \begin{array}{c}(params, mk, xk) \leftarrow \mathsf{WM.Gen}(1^\lambda) \\ (pk, sk) \leftarrow \mathsf{PKE.Mark}(params, mk, \tau)\end{array}\right] = \mathsf{negl}(\lambda),$$

A watermarkable PKE scheme should be functionality-preserving, i.e. maintain encryption correctness and IND-CPA security. We first define encryption correctness:

Definition 8 (Encryption Correctness). *We say that a watermarkable PKE scheme satisfies encryption correctness if for any tag $\tau \in \mathcal{T}$ and any plaintext $m \in \mathcal{M}$, it holds that:*

$$\mathsf{Pr}\left[\mathsf{Dec}(sk, c) \neq m \,\middle|\, \begin{array}{c}(params, mk, xk) \leftarrow \mathsf{WM.Gen}(1^\lambda);\\ (pk, sk) \leftarrow \mathsf{PKE.Mark}(params, mk, \tau);\\ c \leftarrow \mathsf{Enc}(pk, m)\end{array}\right] = \mathsf{negl}(\lambda)$$

We now define IND-CPA security for watermarked PKE. We require that IND-CPA should hold even if the adversary gets to see the marking and extraction keys mk, xk.

Definition 9 (IND-CPA security). *We say that a watermarkable PKE scheme satisfies IND-CPA security if for any tag $\tau \in \mathcal{T}$ and for any PPT adversary $\mathcal{A}$,*

$$\mathsf{Pr}\left[\mathcal{A}(c_b) = b \,\middle|\, \begin{array}{c}(params, mk, xk) \leftarrow \mathsf{WM.Gen}(1^\lambda);\\ (pk, sk) \leftarrow \mathsf{PKE.Mark}(params, mk, \tau);\\ (m_0, m_1) \leftarrow \mathcal{A}(1^\lambda, params, mk, xk, pk);\\ b \xleftarrow{\$} \{0,1\}; c_b \leftarrow \mathsf{Enc}(pk, m_b);\end{array}\right] = \frac{1}{2} + \mathsf{negl}(\lambda)$$

Before defining unremovability and unforgeability, we first define a number of oracles, namely the Challenge, Corrupt and Extract oracles, which are crucial part of the definitions of the security games of unremovability and unforgeability.

The Challenge, Corrupt and Extract oracles. The Challenge oracle, on input a tag τ calls the $\mathsf{PKE.Mark}$ algorithm and returns only the (marked) public key as output, along with an index i which shows how many times $\mathsf{PKE.Mark}$ has been invoked so far. We note that in the definitions of unremovability and unforgeability security games the index i will be initialized to 0. The Corrupt oracle, receives an index i as input and outputs the key pair (pk_i, sk_i) generated in the i-th Challenge oracle query by the $\mathsf{PKE.Mark}$ algorithm. Finally, the Extract oracle, receives as input a circuit, simply runs $\mathsf{Extract}$ algorithm on that input and returns the corresponding output of the algorithm. The formal description of the oracles is given below in Figure 3.

Furthemore, we define the notions of *closeness* and *farness* (cf. Definitions 10, 11) which are crucial for correctly capturing the unremovability and unforgeability notions respectively since those notions capture the notion of "useful" circuits as discussed in Section 1.3. Specifically, in the simpler case of watermarking where circuits are either marked or unmarked, when defining unremovability we have to make sure that an adversary cannot create a circuit which is "close" to a marked circuit but it is unmarked, while, unforgeability requires that it should be difficult for an adversary to come up with a circuit which is "far" from a marked circuit but it remains marked.

Definition 10. *We say that a circuit C is ρ-close to a circuit Dec_{sk}, and we denote $C \sim_\rho \mathsf{Dec}_{sk}$, if $\mathsf{Pr}_{m \xleftarrow{\$} \mathcal{M}}[C(\mathsf{Enc}_{pk}(m)) = m] \geq \rho$.*

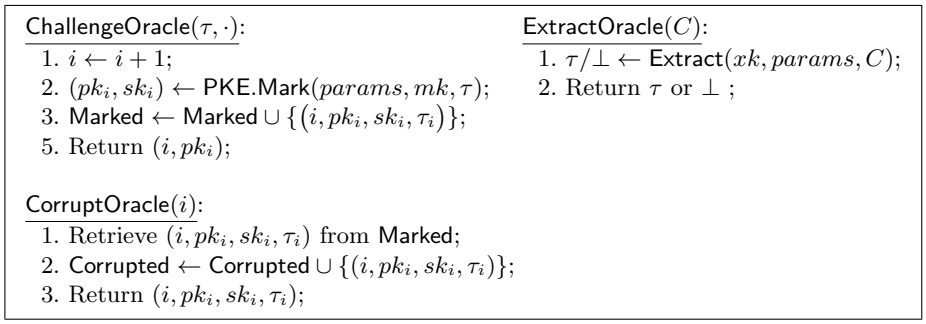

Fig. 3: The Challenge, Corrupt and Extract oracles.

Definition 11. *We say that a circuit C is γ-far from a circuit Dec_{sk}, and we denote $C \nsim_{\gamma} \mathsf{Dec}_{sk}$, if* $\mathsf{Pr}_{m \xleftarrow{\$} \mathcal{M}}[C(\mathsf{Enc}_{pk}(m)) \neq m] \geq \gamma$.

We now proceed by defining ρ-unremovability. In plain words, an adversary should not be able to create a circuit which is ρ-close to a marked circuit generated by the Challenge oracle and at the same time the extraction algorithm returns a different tag or unmarked. As discussed in Section 1.3, our unremovability definition is stronger than that of [CHN+16] as it gives oracle access to the adversary.

Definition 12 (ρ-unremovability). *We say that a watermarking scheme satisfies ρ-unremovability if for any PPT adversary $\mathcal{A}$ participating in the game defined in Fig. 4 it holds that:*

$$\mathsf{Pr}[\mathcal{G}^{\mathsf{unrmv}}_{\rho,\mathcal{A}}(1^{\lambda}) = 1] = \mathsf{negl}(\lambda).$$

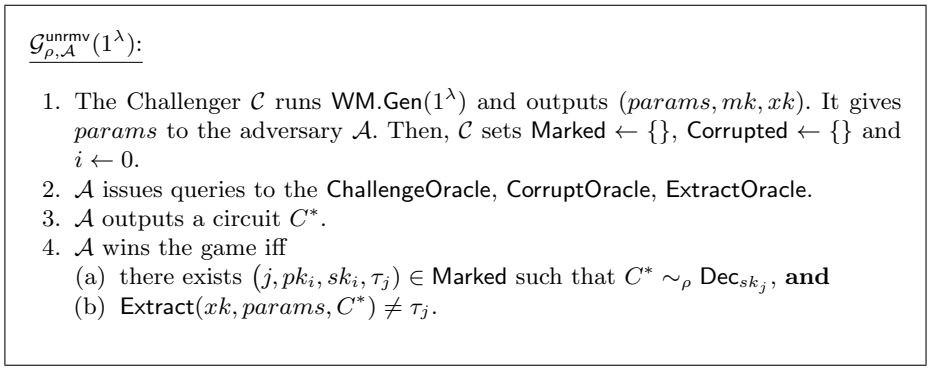

Fig. 4: The ρ-unremovability game.

Finally, we consider the notion of γ-unforgeability. At a high level, this property requires that an adversary cannot create a marked circuit that is γ-far with respect to any of the given marked circuits and at the same is marked.

Definition 13 (γ-unforgeability). *We say that a watermarking scheme satisfies γ-unforgeability if for any PPT adversary $\mathcal{A}$ participating in the game defined in Fig. 5, it holds that:*

$$\mathsf{Pr}[\mathcal{G}^{\mathsf{unforge}}_{\gamma,\mathcal{A}}(1^\lambda) = 1] = \mathsf{negl}(\lambda).$$

$\underline{\mathcal{G}^{\mathsf{unforge}}_{\gamma,\mathcal{A}}(1^\lambda)}$:

1. The Challenger $\mathcal{C}$ runs $\mathsf{WM.Gen}(1^\lambda)$ and outputs $(params, mk, xk)$. It gives $params$ to the adversary $\mathcal{A}$.Then, $\mathcal{C}$ sets $\mathsf{Marked} \leftarrow \{\}$, $\mathsf{Corrupted} \leftarrow \{\}$ and $i \leftarrow 0$.
2. $\mathcal{A}$ issues queries to $\mathsf{ExtractOracle}$, $\mathsf{ChallengeOracle}$ and $\mathsf{CorruptOracle}$.
3. $\mathcal{A}$ outputs a circuit C^*.
4. $\mathcal{A}$ wins the game iff
   (a) For all j s.t. $\big(j, pk_j, sk_j, \tau_j\big) \in \mathsf{Corrupted}$ it holds that $C^* \nsim_\gamma \mathsf{Dec}_{sk_j}$, **and**
   (b) $\mathsf{Extract}\big(params, xk, C^*\big) \neq \perp$.

Fig. 5: The γ-unforgeability game.

4 Our Watermarkable PKE Scheme

As discussed in the introduction, our construction is based on a hybrid encryption scheme for which the public-key has a similar structure to a Pailler ciphertext, while the rest of the scheme (from the point of view of the user) behaves like an ElGamal encryption scheme. We exploit the structure of $\mathbb{Z}^*_{n^2}$, where $n = pq$ and p, q are primes. In our construction, p, q are *safe primes*, meaning that they are of the form $p = 2p' + 1$, $q = 2q' + 1$ where p', q' are also primes. Wen also require a pseudorandom function F, an authenticated symmetric encryption scheme $(\mathsf{S.Gen}, \mathsf{S.Enc}, \mathsf{S.Dec})$, and a collision resistant hash function H.

The tuple of $(\mathsf{WM.Gen}, \mathsf{PKE.Mark}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Extract})$ algorithms that comprise our waterkable PKE scheme is presented below.

$\underline{\mathsf{WM.Gen}:}$ On input 1^λ,

- Run $\mathsf{Param}(1^\lambda)$: Choose safe primes $p = 2p' + 1$ and $q = 2q' + 1$, where p, q are of size at least $\lfloor \lambda/2 \rfloor + 1$. Sample $g \xleftarrow{\$} \mathbb{Z}^*_{n^2}$ and compute $g_1 = g^{2n} \mod n^2$. Return $params = (n, g_1)$.

We denote as $\mathcal{X}_{n^2} = \langle g_1 \rangle$, the subgroup that contains all square n-th residues modulo n^2. Observe that the order of $\mathcal{X}_{n^2}$ is $p'q'$. Note that all elements in $\mathbb{Z}_{n^2}$ can be written in a unique way as $g_1^r(1+n)^v(-1)^\alpha(p_2p - q_2q)^\beta$, where $r \in [p'q']$, $v \in [n]$, $\alpha, \beta \in \{0,1\}$ and $p_1p^2 \equiv 1 \mod q^2$ and $q_2q^2 \equiv 1 \mod p^2$. With $\mathcal{Q}_{n^2}$ we denote the subgroup of quadratic residues modulo n^2 which can be seen that they contain all elements of the form $g_1^r(1+n)^v$, where $r \in [p'q']$ and $v \in \mathbb{Z}_n$. The order of $\mathcal{Q}_{n^2}$ is $np'q'$, one fourth of the order of $\mathbb{Z}^*_{n^2}$. Recall that the order of $\mathbb{Z}^*_{n^2}$ is $n\phi(n) = 4p'q'n$.

- Let $F : \mathcal{K} \times \mathcal{Y} \to \{0,1\}^{\lambda/2}$ be a PRF and a key $k_{\mathsf{p}} \xleftarrow{\$} \mathcal{K}$.
- Let $(\mathsf{S.Gen}, \mathsf{S.Enc}, \mathsf{S.Dec})$ be an authenticated symmetric encryption scheme with security parameter κ_1 and message space $\mathcal{T} = \{0,1\}^{\kappa_2}$ and ciphertext space $\{0,1\}^{\lambda/2}$. Run $k_{\mathsf{e}} \leftarrow \mathsf{S.Gen}(1^{\kappa_1})$.
- Let $H : \{0,1\}^* \to \{0,1\}^{\kappa_3}$ be a hash function.
- Set the marking key as $mk = (k_{\mathsf{p}}, k_{\mathsf{e}})$, the extraction key as $xk = (k_{\mathsf{p}}, k_{\mathsf{e}}, p', q')$ and the public parameters $params = (n, g_1)$.
- Choose $\delta \geq 1/\mathsf{poly}(\lambda)$.

We assume that the parameters $\kappa_1, \kappa_2, \kappa_3$ and λ are compatible.

PKE.Mark: On input $mk = (k_{\mathsf{p}}, k_{\mathsf{e}})$, and a tag $\tau \in \mathcal{T}$,

- Choose $x \xleftarrow{\$} [n/4]$.
- Compute $g_1^x \mod n^2$ and $v_1 = F(k_{\mathsf{p}}, (g_1^x, \tau))$.
- Compute $v_2 = \mathsf{S.Enc}(k_e, H(v_1)||\tau)$.
- Concatenate v_1 and v_2, i.e. compute $v = v_1||v_2$. Compute $h = g_1^x(1+n)^v$.
- Return $pk = h = g_1^x(1+n)^v$, $sk = (x, v)$.

Enc: On input $m \in \mathcal{M}$ and $pk = (n, g_1, h)$, choose $r \xleftarrow{\$} [n^2/4]$ and compute a ciphertext $\psi = \langle g_1^r \mod n^2, (1+n)^r \mod n^2, h^r \cdot m \mod n^2 \rangle$.

Dec: On input $\psi = \langle a, b, c \rangle$ and $sk = (x, v)$, compute $\hat{m} = (a^x b^v)^{-1} c$.

Extract: On input $xk = (k_{\mathsf{p}}, k_{\mathsf{e}}, p', q')$ and a (decryption) circuit C,

(D1). $\mathsf{Count} \leftarrow [\,]$
(D2). Set $\ell = \frac{\lambda}{\delta^2}$ and $\ell^* = \frac{\lambda}{2\delta^2}$.
(D3). For $i = 1$ to ℓ:
  (a) Choose $r \xleftarrow{\$} [n/4]$, $r' \xleftarrow{\$} [n/4]$, $s, s' \xleftarrow{\$} \mathbb{Z}_n$.
  Compute $\psi_i = \langle g_1^r, (1+n)^s, g_1^{r'}(1+n)^{s'} \rangle$.
  (b) Run the algorithm G_{ext} of Figure 6 with inputs C and ψ_i.
  (c) If $G_{\mathsf{ext}}(C, \psi_i)$ returns $\big((y_i, \tau_i), v_{i,1}\big)$
    – if there is record $\big((y_i, \tau_i), v_{i,1}, \mathsf{count}_i\big)$ in Count, set $\mathsf{count}_i \leftarrow \mathsf{count}_i + 1$. If no such record exists initialize $\mathsf{count}_i = 1$ and insert to the table Count a record $\big((y_i, \tau_i), v_{i,1}, \mathsf{count}_i\big)$.
(D4). If there is a record $\big((y_i, \tau_i), v_{i,1}, \mathsf{count}_i\big)$ in Count s.t. $\mathsf{count}_i \geq \ell^*$ then return τ_i, else, return $\mathsf{unmarked}$.

G_{ext}: On input xk, C, ψ_i.

1. Run $C(\psi_i)$. On output $\hat{m}_i$ s.t. $(\hat{m}_i \neq \perp \wedge \hat{m}_i \in \mathcal{Q}_{n^2})$ compute $\hat{c} = \hat{m}_i{}^{\phi(n)} \mod n^2$.
2. $\hat{z} = (\hat{c} - 1)/n$.
3. $z = \hat{z} \cdot \phi(n)^{-1} \mod n$.
4. $v_i = -s^{-1}(z - s') \mod n$.
5. $f = g_1^{-nr'} \hat{m}_i^n \mod n^2$.
6. $y_i = \left(f^{[n^{-1} \mod p'q'][r^{-1} \mod p'q']}\right)^{-1} \mod n^2$.
7. Split the bit representation of v into two parts of $\lambda/2$ bits each, i.e. $v_i = v_{i,1}||v_{i,2}$. Run $h_i||\tau_i \leftarrow \mathsf{S.Dec}(k_{\mathsf{e}}, v_{i,2})$
8. If $\left(h_i||\tau_i \neq \perp \ \wedge H(v_{i,1}) = h_i \ \wedge F(k_{\mathsf{p}}, (y_i, \tau_i)) = v_{i,1}\right)$ return $\left((y_i, \tau_i), v_{i,1}\right)$. Otherwise return $\perp$.

Fig. 6: The G_{ext} algorithm.

5 Security Analysis

In this section, we prove that the scheme $(\mathsf{WM.Gen}, \mathsf{PKE.Mark}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Extract})$ presented in Section 4 is a watermarkable PKE scheme according to the model presented in Section 3.

Theorem 1. *The scheme* $(\mathsf{WM.Gen}, \mathsf{PKE.Mark}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Extract})$ *of Section 4 is a watermarkable PKE scheme assuming (1) that the* DCR *assumption holds, (2) the* $\mathsf{DDH}_{\mathsf{SQNR}}$ *assumption holds, (3) F is a PRF, (4)* $(\mathsf{S.Gen}, \mathsf{S.Enc}, \mathsf{S.Dec})$ *is an authenticated encryption scheme that additionally satisfies indistinguishability between real and random ciphertexts and the hash function H is collision resistant.*

Proving Theorem 1 requires to prove that the properties defined in Section 3 are satisfied. We start in subsection 5.1 by proving that the watermarkable PKE scheme satisfies encryption correctness and IND-CPA security. Then, we proceed in Sections 5.2 by proving extraction correctness property, and finally, in Section 5.3 we prove that ρ-unremovability and γ-unforgeability properties are satisfied, where $\rho \geq 1/2 + 1/\mathsf{poly}(\lambda)$ and $\gamma \leq 1/2 - 1/\mathsf{poly}(\lambda)$. Regarding the ρ-unremovability notion, we note that the lower bound of Cohen et al. [CHN+16] applies to our message-embedding construction. In particular, Cohen et al. [CHN+16] showed that message-embedding watermarking schemes satisfy ρ-unremovability only if $\rho \geq 1/2 + 1/\mathsf{poly}(\lambda)$.

Before proceeding to the detailed proofs, we present the following well-known propositions which will be required for our analysis, i.e. Propositions 1, 2.

Proposition 1. *Let $\mathcal{X}_{n^2} = \langle g_1 \rangle$, the subgroup that contains all square n-th residues modulo n^2, with order $p'q'$. Let $\mathcal{D}_1, \mathcal{D}_2$ the following distributions.$\mathcal{D}_1$: (n, g_1^r) where $r \leftarrow [p'q']$, $\mathcal{D}_2 : (n, g_1^r)$, where $r \leftarrow [n/4]$. $\mathcal{D}_1, \mathcal{D}_2$ are statistically indistinguishable.*

Proposition 2. *Let $\mathcal{X}_{n^2} = \langle g_1 \rangle$, the subgroup that contains all square n-th residues modulo n^2, with order $p'q'$. Let $\mathcal{D}_3, \mathcal{D}_4$ the following distributions. $\mathcal{D}_3$: (n, g_1^r) where $r \leftarrow [np'q']$, $\mathcal{D}_4 : (n, g_1^r)$, where $r \leftarrow [n^2/4]$. $\mathcal{D}_3, \mathcal{D}_4$ are statistically indistinguishable.*

5.1 Encryption Correctness and IND-CPA security

Lemma 1 (Encryption Correctness). *The Watermarkable PKE scheme presented in Section 4 satisfies the encryption correctness property.*

Proof. We prove that for any $m \in \mathcal{X}_{n^2}$, and any (pk, sk) generated by the $\mathsf{PKE.Mark}$ algorithm, it holds that $\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m$. It can be easily seen that $(g_1^{rx}(1+n)^{rv})^{-1} g_1^{rx}(1+n)^{rv} m = m$. □

Lemma 2 (IND-CPA security). *The Watermarkable PKE scheme of Section 4 is IND-CPA secure under the $\mathsf{DDH}_{\mathsf{SQNR}}$ assumption.*

Proof. We will prove this lemma by defining a sequence of games. By $\mathcal{G}_0$ we denote the IND-CPA security game for the watermarkable PKE scheme.

Game $\mathcal{G}_0$: On input 1^λ,

1. $(n, g_1) \leftarrow \mathsf{Param}(1^\lambda)$; $k_\mathsf{p} \xleftarrow{\$} \mathcal{K}$; $k_\mathsf{e} \xleftarrow{\$} \mathsf{S.Gen}(1^{\kappa_1})$; $mk = (k_\mathsf{p}, k_\mathsf{e})$; $xk = (p', q', k_\mathsf{p}, k_\mathsf{e})$;
2. $x \xleftarrow{\$} [n/4]$; $v_1 = F(k_\mathsf{p}, (g_1^x, \tau))$; $v_2 = \mathsf{S.Enc}(k_\mathsf{e}, H(v_1)||\tau)$; $v = v_1||v_2$; $h = g_1^x(1+n)^v$; $pk = (n, g_1, h)$; $sk = (x, v)$;
3. $(m_0, m_1) \leftarrow \mathcal{A}(pk, mk, xk)$;
4. $b \xleftarrow{\$} \{0,1\}$; $r \xleftarrow{\$} [n^2/4]$; $c = \langle g_1^r, (1+n)^r, h^r \cdot m_b \rangle$;
5. $b^* \leftarrow \mathcal{A}(c)$;

In the game $\mathcal{G}_0$, $c = \langle g_1^r, (1+n)^r, g_1^{rx}(1+n)^{rv} \cdot m_b \rangle$.

Game $\mathcal{G}_1$: $\mathcal{G}_1$ is the same as $\mathcal{G}_0$, except the following: At Step 4 of the game, the Challenger samples $r_1 \xleftarrow{\$} [n/4]$, $s_1 \xleftarrow{\$} [n]$ and computes $c = \langle g_1^{r_1}, (1+n)^{s_1}, g_1^{r_1 x}(1+n)^{s_1 v} m_b \rangle$.

Game $\mathcal{G}_2$: The game $\mathcal{G}_2$ is the same as $\mathcal{G}_1$ except the following: At Step 4 of game $\mathcal{G}_2$, the Challenger chooses $r^* \xleftarrow{\$} [n/4]$ and computes $c = \langle g_1^{r_1}, (1+n)^{s_1}, g_1^{r^*}(1+n)^{s_1 v} \rangle$.

Analysis.
• $\mathcal{G}_1$ is indistinguishable from $\mathcal{G}_0$ due to Propositions 1, 2: By Proposition 2, recall that sampling r uniformly from $[n^2/4]$ is indistinguishable from sampling r uniformly from $[np'q']$. If r is sampled from $[np'q']$ and $r_1 = r \mod p'q'$, $s_1 = r \mod n$, then r_1 is uniformly distributed $[p'q']$ and s_1 is uniformly distributed in $[n]$. When r appears over g_1 which is of order $p'q'$, it can be substituted by r_1. Similarly, when r appears over an element of order n, it can be substituted with s_1. By Proposition 1, sampling r_1 uniformly from $[p'q']$ is statistically

indistinguishable from sampling r_1 from $[n/4]$. Therefore, $\mathcal{G}_1$ is statistically indistinguishable from $\mathcal{G}_0$, which means that

$$\left|\mathsf{Pr}[\mathcal{G}_1 = 1] - \mathsf{Pr}[\mathcal{G}_0 = 1]\right| = \mathsf{negl}(\lambda). \tag{1}$$

• $\mathcal{G}_1$ and $\mathcal{G}_2$ are computationally indistinguishable under the $\mathsf{DDH}_{\mathsf{SQNR}}$ assumption: Assuming that there is a PPT adversary $\mathcal{A}$ which distinguishes between the games $\mathcal{G}_1$ and $\mathcal{G}_2$ with non-negligible probability α, we construct a PPT adversary $\mathcal{B}$ which breaks the $\mathsf{DDH}_{\mathsf{SQNR}}$ assumption as follows.

$\mathcal{B}$: On input $\langle n, p', q', g_1, y, g_1^r, v^* \rangle$,

(a) Run the Steps 1,2 as in the games $\mathcal{G}_1, \mathcal{G}_2$.

(b) Upon receiving m_0, m_1 from $\mathcal{A}$, choose $b \xleftarrow{\$} \{0,1\}$ and $s \xleftarrow{\$} \mathbb{Z}_n$. Compute $c = \langle g_1^r, (1+n)^s, v^*(1+n)^{sv} \cdot m_b \rangle$ and give c and the factorization of n to $\mathcal{A}$.

(c) $\mathcal{A}$ outputs b^* and then $\mathcal{B}$ outputs b^* as well.

First, we note that since $y \xleftarrow{\$} \mathcal{X}_{n^2}$, $y = g_1^x$, for some $x \in [p'q']$. We consider the following cases regarding the value of v^*:

- $v^* = y^r$ where $r \xleftarrow{\$} [p'q']$: In this case, the ciphertext c which is provided to $\mathcal{A}$ is of the form $c = \langle g_1^r, (1+n)^s, g_1^{xr}(1+n)^{sv} \cdot m_b \rangle$. By proposition 1, $\langle g_1^r, (1+n)^s, g_1^{xr}(1+n)^{sv} \cdot m_b \rangle$ statistically indistinguishable from $\langle g_1^r, (1+n)^s, g_1^{xr}(1+n)^{sv} \cdot m_b \rangle$ where $r \xleftarrow{\$} [n/4]$. Therefore, in this case, $\mathcal{B}$ interacts with $\mathcal{A}$ as in the game $\mathcal{G}_1$ and thus

$$\mathsf{Pr}[\mathcal{B}(n, g_1, y, g_1^r, y^r) = 1] = \mathsf{Pr}[\mathcal{G}_1 = 1].$$

- $v^* = y^{r'}$: In this case, c is of the form $\langle g_1^r, (1+n)^s, y^{r'}(1+n)^{sv} \cdot m_b \rangle$, i.e. $c = \langle g_1^r, (1+n)^s, g_1^{xr'+m'}(1+n)^{sv} \rangle$, where $m_b = g_1^{m'}$ for some $m' \in [p'q']$. Since r' is uniformly chosen and it is independent from the view of the adversary, we have that $xr' + m'$ is uniformly distributed in $[p'q']$. This means that $\mathcal{B}$ interacts with $\mathcal{A}$ as in the game $\mathcal{G}_2$. Thus, $\mathsf{Pr}[\mathcal{B}(n, g_1, y, g_1^r, y^{r'}) = 1] = \mathsf{Pr}[\mathcal{G}_2 = 1]$.

By the $\mathsf{DDH}_{\mathsf{SQNR}}$ assumption, we have that for any PPT adversary $\mathcal{A}$,

$$\left|\mathsf{Pr}[\mathcal{G}_2 = 1] - \mathsf{Pr}[\mathcal{G}_1 = 1]\right| \leq \mathsf{negl}(\lambda). \tag{2}$$

• For any PPT adversary $\mathcal{A}$, it holds that $\mathsf{Pr}[\mathcal{G}_2 = 1] = 1/2$: This holds because the ciphertext c in game $\mathcal{G}_2$ is independent of the messages m_0, m_1 chosen by the adversary.

Therefore, it holds that for any PPT adversary $\mathcal{A}$, $\left|\mathsf{Pr}[\mathcal{G}_0 = 1] - 1/2\right| \leq \mathsf{negl}(\lambda)$. This completes our proof. □

5.2 Extraction Correctness

Lemma 3 (Extraction correctness). *The Watermarkable PKE scheme of Section 4 satisfies extraction correctness, under the following assumptions: (1) correctness is satisfied (cf. Lemma 1), (2) F is a pseudorandom function and (3) the symmetric encryption scheme* $(\mathsf{S.Gen}, \mathsf{S.Enc}, \mathsf{S.Dec})$ *is correct.*

Proof. Consider a pair $(pk, sk) \leftarrow \mathsf{PKE.Mark}(params, mk, \tau)$, where $sk = (x, v)$ with $v = v_1||v_2$ s.t. $v_1 = F(k_\mathsf{p}, (g_1^x, \tau))$ and $pk = (n, g_1, h)$ with $h = g_1^x(1+n)^v$. We will prove that if $\mathsf{Extract}$ receives as input Dec_{sk}, it always returns τ. Let $\psi_i = \langle g_1^r, (1+n)^s, g_1^{r'}(1+n)^{s'} \rangle$ be a ciphertext generated as described at Step (D3)a of the $\mathsf{Extract}$ algorithm. Following the steps 1-8 of the G_ext algorithm of Figure 6, we prove that $\mathsf{Extract}$ on input Dec_{sk} and ψ_i returns $((g_1^x, \tau), v_1)$.

Step 1: $\mathsf{Dec}_{sk}(\psi_i) = g_1^{r'-xr}(1+n)^{s'-sv}$. Since $\hat{m}_i = g_1^{r'-xr}(1+n)^{s'-sv} \in \mathcal{Q}_{n^2}$, we compute $\hat{c} = \hat{m}_i^{\phi(n)} = g_1^{\phi(n)(r'-xr)}(1+n)^{\phi(n)(s'-sv)} = (1+n)^{\phi(n)(s'-sv)} = 1 + n[\phi(n)(s'-sv) \mod n]$.

Step 2: $\hat{z} = \frac{\hat{c}-1}{n} = \phi(n)(s'-sv) \mod n$.

Step 3: $z = \phi(n)^{-1}\hat{z} \mod n = \phi(n)^{-1}\phi(n)(s'-sv) = (s'-sv) \mod n$.

Step 4: $-s^{-1}(z-s') \mod n = -s^{-1}(s'-sv-s') \mod n = v$.

Step 5: $f = g_1^{-nr'}\hat{m}_i^n \mod n^2 = g_1^{-nr'}g_1^{n(r'-xr)}(1+n)^{n(s'-sv)} = g_1^{-nr'}g_1^{n(r'-xr)} = g_1^{-nxr}$.

Step 6: $\left(f^{[n^{-1} \mod p'q'][r^{-1} \mod p'q']}\right)^{-1} \mod n^2 = \left(g_1^{-nxr[n^{-1} \mod p'q'][r^{-1} \mod p'q']}\right)^{-1} = g_1^x$.

Step 7: Split the bit representation of v into two parts of $\lambda/2$ bits each, i.e. $v = v_1||v_2$. Due to the correctness property of the symmetric encryption scheme, it holds that $H(v_1)||\tau \leftarrow \mathsf{S.Dec}(k_\mathsf{e}, v_2)$.

Step 8: It holds that $v_1 = F(k_\mathsf{p}, (g_1^x, \tau))$ and therefore G_ext returns $((g_1^x, \tau), v_1)$.

Since G_ext returns $((g_1^x, \tau), v_1)$ for any of the ciphertexts $\psi_1, \ldots, \psi_\ell$, which are generated by $\mathsf{Extract}$ at Step (D3)a, then $\mathsf{Extract}$ returns the message τ.

5.3 Proving unremovability and unforgeability properties

As it will become clear in the proofs of unremovability and unforgeability properties in this section, it is crucial that our watermarkable PKE scheme satisfies a property called *ciphertext indistinguishability*. At a high-level, no PPT adversary should be able to distinguish between the ciphertexts constructed as described at Step (D3)a of the $\mathsf{Extract}$ algorithm and standard encrypted plaintexts, under any public key pk. For the simplicity, we will refer to the ciphertexts computed at Step (D3)a as "extraction ciphertexts". We intuitively explain why this property is essential in proving unremovability and unforgeability by presenting below some simple scenarios where it is assumed that a potential attacker could distinguish between standard ciphertexts and extraction ciphertexts. The examples below refer to the simpler case of watermarking where circuits are either marked or unmarked.

– Assume an attacker $\mathcal{A}$ against the ρ-unremovability game which has obtained a pair (pk_i, sk_i) by issuing a $\mathsf{CorruptOracle}$ query. If $\mathcal{A}$ could distinguish "extraction ciphertexts" from valid ciphertexts under pk_i, then it could construct a circuit C^* which runs Dec_{sk_i} when receving as input an encrypted plaintext under the key pk_i and returns $\perp$ when receiving as input a "extraction ciphertext". Therefore, $\mathcal{A}$ wins the ρ-unremovability game since C^* is "close" (specifically 1-close) to Dec_{sk_i} and $\mathsf{Extract}$ on input C^* returns $\perp$, i.e. *unmarked*.
– Assume an attacker $\mathcal{A}'$ against the γ-unforgeability game which has obtained a pair (pk_i, sk_i) through a query to the $\mathsf{CorruptOracle}$. If $\mathcal{A}'$ could distinguish "extraction ciphertexts" from valid ciphertexts under pk_i, then it could construct a circuit C^* which returns $\perp$ when receiving as input a valid ciphertext under pk_i and runs Dec_{sk_i} when receiving as input a "extraction ciphertext". In that case, $\mathcal{A}'$ the attacker managed to break γ-unforgeability since C^* which is "far" (specifically 1-far) from Dec_{sk_i} but $\mathsf{Extract}$ would decide that C^* is *marked* the decryption circuit under the key sk which is at the same time *marked*.

Before proving *unremovability* and *unforgeability*, we present some intermediate lemmas which will be necessary in our proofs. First,we show that for any public key pk, even if the adversary is given the corresponding secret key sk, the adversary is not able to distinguish between ciphertexts encrypted under pk from ciphertexts prepared under $\mathsf{Extract}$ algorithm.

Lemma 4. *Let $\tau \in \mathcal{T}$ and $pk = (n, g_1, h)$, $sk = (x, v)$ returned by $\mathsf{PKE.Mark}(mk, \tau)$, where $x \xleftarrow{\$} [n/4]$, $v = v_1||v_2$, $v_1 = F(K, (g_1^x, \tau))$, $v_2 = \mathsf{S.Enc}(k_e, H(v_1)||\tau)$ and $h = g_1^x(1+n)^v$. Assuming that the DCR assumption holds, F is a PRF and the symmetric encryption scheme $(\mathsf{S.Gen}, \mathsf{S.Enc}, \mathsf{Dec})$ satisfies real-or-random security against chosen plaintext attacks (cf. Definition 5), it holds that*

$$\langle n, g_1, x, v, g_1^r, (1+n)^r, g_1^{xr}(1+n)^{rv} \cdot m \rangle \overset{c}{\approx} \langle n, g_1, x, v, g_1^{r_1}, (1+n)^{s_1}, g_1^{r_2}(1+n)^{s_2} \rangle,$$

where $r \xleftarrow{\$} [n^2/4]$, $r_1, r_2, x \xleftarrow{\$} [n/4]$, $s_1, s_2 \xleftarrow{\$} \mathbb{Z}_n$ and $m \xleftarrow{\$} \mathcal{X}_{n^2}$.

The proof of the above is included in Appendix A.

Next, we proceed with the proof of Lemma 5, which shows that if G_{ext} on input dk, C^* and "extraction ciphertext" ψ, at Step 4 outputs a value $v \in \mathbb{Z}_n$ and at Step 6 it outputs $y = g_1^x \mod n^2$, then this implies that C^* has run $\mathsf{Dec}_{sk}(\psi)$, where $sk = (x, v)$.

Lemma 5. *Let $\psi = \langle g_1^r, (1+n)^s, g_1^{r'}(1+n)^{s'} \rangle$ where $r \xleftarrow{\$} [n/4]$, $r' \xleftarrow{\$} [n/4]$, $s, s' \xleftarrow{\$} \mathbb{Z}_n$ and let C^* be a circuit which on input ψ returns $\hat{m}$ s.t. (1) G_{ext} at Step 4 outputs v, and (2) G_{ext} and Step 6 outputs g^x. Then, $\hat{m} = g_1^{r'-xr}(1+n)^{s'-sv} \mod n^2$.*

The proof of the above Lemma is included in Appendix A.

Test algorithm
Input: Two circuits C^* and Dec_{sk} and $\mathcal{M}$ the input space of $\mathsf{Enc}_{pk}(\cdot)$.
Parameters: $\delta \geq 1/\mathsf{poly}(\lambda), \rho \geq 1/2 + \delta$.

1. Set $\mathsf{cnt} = 0$ and $\ell = \lambda/\delta^2$.
2. Sample $m_1, \ldots, m_\ell$ independently and uniformly at random from $\mathcal{M}$.
3. For $j = 1, \ldots, \ell$, compute $c_j = \mathsf{Enc}_{pk}(m_j)$.
4. For $j = 1, \ldots, \ell$, do
   (a) Compute $C^*(c_j) = m_j$ set $\mathsf{cnt} = \mathsf{cnt} + 1$.
5. If $\mathsf{cnt} \geq \frac{\lambda}{2\delta^2}$ return 1, otherwise return 0.

Fig. 7: Test algorithm.

Testing closeness between circuits. Below, we include standard bounds which are utilized for testing closeness and farness between circuits.

Proposition 3. *Let* $\delta \geq \frac{1}{\mathsf{poly}(\lambda)}$, $\rho > \frac{1}{2} + \delta$ *and* $\gamma \leq \frac{1}{2} - \delta$.

- *For any* $\delta \geq 1/\mathsf{poly}(\lambda)$, *if* $C^* \sim_\rho \mathsf{Dec}_{sk}$, *then* $\mathsf{Pr}[\mathsf{cnt} < \frac{\lambda}{2\delta^2}] = \mathsf{negl}(\lambda)$.
- *For any* $\delta \geq 1/\mathsf{poly}(\lambda)$, *if* $C^* \nsim_\gamma \mathsf{Dec}_{sk}$, *then* $\mathsf{Pr}[\mathsf{cnt} < \frac{\lambda}{2\delta^2}] = 1 - \mathsf{negl}(\lambda)$.

Proposition 3 holds by Chernoff bounds.

Lemma 6 (ρ-Unremovability). *The watermarkable PKE scheme* (WM.Gen, PKE.Mark, Enc, Dec, Extract) *satisfies ρ-Unremovability under the following assumptions: (1) correctness property is satisfied (cf. Lemma 1), (2) F is a pseudorandom function and (3) ciphertext indisinguishability holds (cf. Lemma 4), (4) the symmetric encryption scheme* (S.Gen, S.Enc, S.Dec) *satisfied integrity of ciphertexts and (5) the hash function H is collision resistant.*

Proof idea. We now provide the general idea behind our unremovability proof and give the full proof in Appendix B.

Recall by the definition of the ρ-unremovability game that an adversary is allowed to obtain a number of (watermarked) public-secret key pairs as well as a number of public keys by issuing queries to the Corrupt and Challenge oracles respectively. Given that, our first goal is to prove that an adversary is not able to create a new valid watermarked secret key, e.g. possibly by combining the secret keys that he possesses. In more detail, we prove that if an adversary issues a circuit C as a query to the Extract oracle (or outputs C in the end of the game) and the Extract oracle returns a tag τ, then this means that C implements the decryption algorithm under one of the secret keys generated previously by the challenger. We prove this via a sequence of hybrid games in the detailed proof.

In each of the hybrid games the algorithm G_{ext} is gradually altered so that in the game $\mathcal{G}_4$, the modified G_{ext} (renamed as G^4_{ext}) performs as follows: On input a circuit C and a extraction ciphertext ψ, if a value $y = g_1^x$ is computed at Step 6

and a value v is computed at Step 4 it simply checks whether there is a secret key (x, v) generated previously by the Challenge oracle. Recall by the definition of the ρ-unremovability game that the challenger stores every public-secret key pair which is generated together with the corresponding tag (such information is stored at a table called Marked). Therefore, if such secret key $sk = (x, v)$ exists, the tag τ associated with the secret key sk would be returned as the output of the $\mathsf{Extract}$ algorithm (assuming that for the *majority* of extraction ciphertexts given as input to G_{ext}, the same (x, v) will be computed). Based on Lemma 5, this implies that the decryption algorithm under the key $sk = (x, v)$ has been run by the circuit C for the majority of ciphertexts. Given that, in the hybrid game $\mathcal{G}_5$ the computation of the (modified) G_{ext} algorithm, i.e. Steps 1-6, are replaced by an algorithm which pre-computes how the extraction ciphertext which is received as input is decrypted by each secret key that has been generated so far. Then, by running the circuit with input the extraction ciphertext, and matching the output with the previous results, the algorithm infers which secret key has been used for the decryption (assuming there is one).

Then, since the first winning condition of the ρ-unremovability game is that the circuit C^* output by the adversary should be ρ-close to a marked decryption circuit, the challenger can guess such a circuit (cf. hybrid game $\mathcal{G}_6$). Based on that and the *ciphertext indistinguishability* property of the watermarkable PKE scheme (cf. Lemma 4) the challenger can gradually substitute extraction ciphertexts sampled in the Step (D3)a with encrypted plaintexts under the marked public key guessed by the challenger. We note that the plaintexts which are encrypted at this stage are chosen uniformly at random from the plaintext space. The above change is performed in the hybrid games $\mathcal{G}_{7,1}$- $\mathcal{G}_{7,\ell}$ and in the game $\mathcal{G}_{7,\ell}$ the adversarial circuit C^* in run on input ℓ encrypted plaintexts under the aforementioned public key and it is checked how which portion of such ciphertexts is decrypted correctly by C^* in order to decide whether C^* is marked or not. Last, by utilizing a Chernoff bound, we prove that if C^* is ρ-close to a decryption circuit then it cannot decrypt correctly less than ℓ ciphertexts except with negligible probability. To put it differently, C^* will be detected as marked with the tag τ which was initially related with this specific public-secret key pair (i.e. was given as input to the Mark algorithm).

Lemma 7 (γ-Unforgeability). *The watermarkable PKE scheme* $(\mathsf{WM.Gen}, \mathsf{PKE.Mark}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Extract})$ *satisfies γ-unforgeability under the following assumptions: (1)ciphertext indistinguishability holds (cf. Lemma 4), (2) IND-CPA security holds , (3) F is a pseudorandom function.*

Proof sketch. Without loss of generality, we assume that an adversary $\mathcal{A}$ interacting with a Challenger in the unforgeability game $\mathcal{G}^{\mathsf{unforge}}_{\gamma,\mathcal{A}}$ issues q_1 queries to the $\mathsf{ChallengeOracle}$, q_2 queries to the $\mathsf{CorruptOracle}$ and q_3 queries to the $\mathsf{ExtractOracle}$. This means that q_1 public-secret key pairs have been generated by $\mathsf{PKE.Mark}$, i.e. $(pk_1, sk_1), \ldots, (pk_{q_1}, sk_{q_1})$, the adversary has obtained all the public keys, but also $\mathcal{A}$ has obtained q_2 public-secret key pairs, i.e. (pk_{j_1}, sk_{j_1})

$\ldots$, $(pk_{j_{q_2}}, sk_{j_{q_2}})$. Recall that $\mathcal{A}$ wins if (1) for any $\mathsf{Dec}_{sk_i} \in \mathsf{Corrupted}$, C^* is γ-far from Dec_{sk_i}, and (2) $\mathsf{Extract}(params, xk, C^*) = \tau \neq \perp$.

This proof is similar to the unremovability proof. Let us first provide some intuition on a potential scenario where unforgeability is broken. Assume an adversary which holds at least two secret keys could combine their components (i.e. the values $v_{i,1}, v_{i,2}$ of the component $v = v_{i,1}||v_{i,2}$) and obtain a new valid "watermarked" secret key, then the decryption algorithm would be far from any decryption algorithm in the set $\mathsf{Corrupted}$ and $\mathsf{Extract}$ would return would extract a tag indicating that the circuit is marked. Intuitively, this is prevented due to the fact that the components $v_{i,1}, v_{i,2}$ are related between each other as $v_{i,2}$ encrypts $H(v_{i,1})||\tau$. In addition, by requiring that the symmetric encryption scheme is authenticated, an adversary cannot create new valid ciphertexts on its own. This combination essentially ensures that new valid watermarked secret keys cannot be easily created. We follow the same sequence of games as described in the unremovability game.

In particular, in the sequence of games $\mathcal{G}_0$-$\mathcal{G}_5$ it has been proved that if an adversary issues a circuit C as a query to the Extract oracle (or outputs C in the end of the game) that is *marked* with a tag τ, then this means that C implements the decryption algorithm under one of the secret keys generated previously by the challenger. Due to Lemma 5, in the hybrid game $\mathcal{G}_5$ the computation of the (modified) G_{ext} algorithm, i.e. Steps 1-6, are replaced by an algorithm which precomputes how the extraction ciphertext which is received as input is decrypted by each secret key that has been generated so far. Then, due to ciphertext indistinguishability "extraction ciphertexts" can be replaced by standard ciphertexts under a chosen public key (chosen uniformly at random by the challenger). Since for any $\mathsf{Dec}_{sk_i} \in \mathsf{Corrupted}$, C^* should be γ-far from Dec_{sk_i}, by Proposition 3, C^* decrypts correctly ℓ^* out of ℓ ciphertexts only with negligible probability. Therefore, the only chance that an adversary wins the unforgeability game is breaking the IND-CPA security of the watermarkable PKE scheme (cf. Lemma 2), as the adversary should decrypt correctly at least ℓ^* out of ℓ ciphertexts under a secret key that it does possess, i.e. it does not belong to the set $\mathsf{Corrupted}$.

References

AV19. Prabhanjan Ananth and Vinod Vaikuntanathan. Optimal bounded-collusion secure functional encryption. In *TCC*, 2019.

BCG+17. Zvika Brakerski, Nishanth Chandran, Vipul Goyal, Aayush Jain, Amit Sahai, and Gil Segev. Hierarchical functional encryption. In *8th Innovations in Theoretical Computer Science Conference, ITCS*, pages 8:1–8:27, 2017.

BDJR97. Mihir Bellare, Anand Desai, E. Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *FOCS '97*, pages 394–403, 1997.

BGI+01. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *CRYPTO 2001*, pages 1–18, August 2001.

BGI+12. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.

BKS17. Foteini Baldimtsi, Aggelos Kiayias, and Katerina Samari. Watermarking public-key cryptographic functionalities and implementations. *ISC 2017*, pages 173–191, November 2017.

BLW17. Dan Boneh, Kevin Lewi, and David J. Wu. Constraining pseudorandom functions privately. *PKC 2017*, pages 494–524, March 2017.

BN00. Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *ASIACRYPT*, pages 531–545, 2000.

BSW06. Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. *EUROCRYPT 2006*, pages 573–592, May 2006.

BW06. Dan Boneh and Brent Waters. A fully collusion resistant broadcast, trace, and revoke system. *CCS 2006*, pages 211–220, November 2006.

BZ14. Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. *CRYPTO 2014*, pages 480–499, August 2014.

CFN94. Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. *CRYPTO 1994*, pages 257–270, August 1994.

CHN+16. Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. *STOC 2016*, pages 1115–1127, June 2016.

CHN+18. Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. *SIAM J. Comput.*, 47(6):2157–2202, 2018.

DFKY03. Yevgeniy Dodis, Nelly Fazio, Aggelos Kiayias, and Moti Yung. Scalable public-key tracing and revoking. *PODC 2003*, pages 190–199, July 2003.

GKM+19. Rishab Goyal, Sam Kim, Nathan Manohar, Brent Waters, and David J. Wu. Watermarking public-key cryptographic primitives. In *CRYPTO 2019*, pages 367–398, August 2019.

GKW18. Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In *STOC*, pages 660–670, 2018.

GKW19. Rishab Goyal, Venkata Koppula, and Brent Waters. New approaches to traitor tracing with embedded identities. In Dennis Hofheinz and Alon Rosen, editors, *TCC*, 2019.

KTY07. Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Group encryption. In *ASIACRYPT*, pages 181–199, 2007.

KW17. Sam Kim and David J. Wu. Watermarking cryptographic functionalities from standard lattice assumptions. *CRYPTO 2017*, pages 503–536, August 2017.

KW19. Sam Kim and David J. Wu. Watermarking prfs from lattices: Stronger security via extractable prfs. In *CRYPTO 2019*, pages 335–366, 2019.

KY02. Aggelos Kiayias and Moti Yung. Traitor tracing with constant transmission rate. *EUROCRYPT 2002*, pages 450–465, May 2002.

Nis13. Ryo Nishimaki. How to watermark cryptographic functions. *EUROCRYPT 2013*, pages 111–125, May 2013.

Nis20. Ryo Nishimaki. Equipping public-key cryptographic primitives with watermarking (or: A hole is to watermark). *Theory of Cryptography (TCC)*, 2020.

NWZ16. Ryo Nishimaki, Daniel Wichs, and Mark Zhandry. Anonymous traitor tracing: How to embed arbitrary information in a key. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT*, 2016.

Pai99. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - EUROCRYPT '99,*, pages 223–238, 1999.

QWZ18. Willy Quach, Daniel Wichs, and Giorgos Zirdelis. Watermarking prfs under standard assumptions: Public marking and security with extraction queries. *TCC 2018*, pages 669–698, November 2018.

SW14. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC 2014*, pages 475–484, 2014.

YAL+19. Rupeng Yang, Man Ho Au, Junzuo Lai, Qiuliang Xu, and Zuoxia Yu. Collusion resistant watermarking schemes for cryptographic functionalities. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT*, 2019.

YAYX20. Rupeng Yang, Man Ho Au, Zuoxia Yu, and Qiuliang Xu. Collusion resistant watermarkable prfs from standard assumptions. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO*, pages 590–620, 2020.

YF11. Maki Yoshida and Toru Fujiwara. Toward digital watermarking for cryptographic data. *IEICE Transactions*, 94-A(1):270–272, 2011.

Supplementary Material

A Proof of Lemmas 4 and 5

Proof of Lemma 4. Due to the fact that F is a PRF, and $(\mathsf{S.Gen}, \mathsf{S.Enc}, \mathsf{Dec})$ satisfies real-or-random security against chosen plaintext attacks , the distribution $\langle n, g_1, x, v, g_1^r, (1+n)^r, g_1^{xr}(1+n)^{rv}\cdot m\rangle$ is computationally indistinguishable from the following distribution $\mathcal{D}_1$. We denote as $\mathcal{D}_1$ the distribution:

$$\langle n, g_1, x, v, g_1^r, (1+n)^r, g_1^{xr}(1+n)^{rv}\cdot m\rangle,\ g_1 \xleftarrow{\$} \mathcal{X}_{n^2}, x \xleftarrow{\$} [n/4], v \xleftarrow{\$} [n], r \xleftarrow{\$} [n^2/4], m \xleftarrow{\$} \mathcal{X}_{n^2}.$$

By Proposition 2, we have that $\mathcal{D}_1$ is *statistically indistinguishable* from $\mathcal{D}_2$, where $\mathcal{D}_2$ is the following distribution:

$$\langle n, g_1, x, v, g_1^r, (1+n)^r, g_1^{xr}(1+n)^{rv}\cdot m\rangle,\ g_1 \xleftarrow{\$} \mathcal{X}_{n^2}, x \xleftarrow{\$} [n/4], v \xleftarrow{\$} [n], r \xleftarrow{\$} [np'q'], m \xleftarrow{\$} \mathcal{X}_{n^2}.$$

Based on the Chinese Remainder Theorem, $\mathcal{D}_2$ can be rewritten as follows. We denote as $\mathcal{D}_3$ the following distribution:

$$\langle n, g_1, x, v, g_1^{r_1}, (1+n)^{s_1}, g_1^{xr_1}(1+n)^{s_1v}\cdot m\rangle,\ g_1 \xleftarrow{\$} \mathcal{X}_{n^2}, x \xleftarrow{\$} [n/4], s_1, v \xleftarrow{\$} [n], r_1 \xleftarrow{\$} [p'q'], m \xleftarrow{\$} \mathcal{X}_{n^2}.$$

DCR assumption implies that no PPT adversary can distinguish between a random quadratic residue and a random square n-th residue. Therefore, by DCR assumption, we have that $\mathcal{D}_3$ is *computationally indistinguishable* from $\mathcal{D}_4$, where $\mathcal{D}_4$ is defined as follows:

$$\langle n, g_1, x, v, g_1^{r_1}, (1+n)^{s_1}, g_1^{xr_1}(1+n)^{s_1v}\cdot m\rangle,\ g_1 \xleftarrow{\$} \mathcal{X}_{n^2}, x \xleftarrow{\$} [n/4], s_1, v \xleftarrow{\$} [n], r_1 \xleftarrow{\$} [p'q'], m \xleftarrow{\$} \mathcal{Q}_{n^2}.$$

Since $m \in \mathcal{Q}_{n^2}$, it can be written as $m = g^{r^*}(1+n)^{s^*}$ for some $r^* \in [p'q']$ and $s^* \in [n]$. Based on that, $\mathcal{D}_4$ can be rewritten as $\mathcal{D}_5$:

$$\langle n, g_1, x, v, g_1^{r_1}, (1+n)^{s_1}, g_1^{xr_1+r^*}(1+n)^{s_1v+s^*}\rangle,\ g_1 \xleftarrow{\$} \mathcal{X}_{n^2}, x \xleftarrow{\$} [n/4], s^*, s_1, v \xleftarrow{\$} [n], r^*, r_1 \xleftarrow{\$} [p'q'].$$

Since r^*, s^* are independent of the view of the adversary and are uniformly distributed in $[p'q']$ and in $[n]$ respectively, $\mathcal{D}_5$ is *statistically* indistinguishable from $\mathcal{D}_6$:

$$\langle n, g_1, x, v, g_1^{r_1}, (1+n)^{s_1}, g_1^{r_2}(1+n)^{s_2}\rangle,\ g_1 \xleftarrow{\$} \mathcal{X}_{n^2}, x \xleftarrow{\$} [n/4], s_1, s_2, v \xleftarrow{\$} [n], r_1, r_2 \xleftarrow{\$} [p'q'].$$

By Proposition 1, it holds that $\mathcal{D}_6$ is *statistically* indistinguishable from $\mathcal{D}_7$:

$$\langle n, g_1, x, v, g_1^{r_1}, (1+n)^{s_1}, g_1^{r_2}(1+n)^{s_2}\rangle,\ g_1 \xleftarrow{\$} \mathcal{X}_{n^2}, x \xleftarrow{\$} [n/4], s_1, s_2, v \xleftarrow{\$} [n], r_1, r_2 \xleftarrow{\$} [n/4].$$

This completes our proof. □

Next, we proceed with the proof of Lemma 5, which is utilized in the unremovability and unforgeability proofs.

Proof of Lemma 5. First, we follow Steps 1-4 in reverse order to see what it means $\hat{m}$ that $\mathsf{Extract}$ extracts v at Step 4.

Step 4: $v = -s^{-1}(z - s') \mod n$. Note that s, s' are chosen by $\mathsf{Extract}$ to create ψ_i. Since s^{-1} is unique, $z = s' - sv_i$.
Step 3: $z = \hat{z}\phi(n)^{-1} \mod n$. This implies that $\hat{z} = z\phi(n) = (s' - sv)\phi(n) \mod n$.
Step 2: $\hat{z} = (\hat{c} - 1)/n$. Thus, $\hat{c} = 1 + [\phi(n)(s' - sv) \mod n] \cdot n$.
Step 1: $\hat{m}^{\phi(n)} = 1 + [\phi(n)(s' - sv) \mod n] \cdot n \mod n^2 = (1+n)^{[\phi(n)(s'-sv) \mod n]} \mod n^2$.

Next, we follow Steps 5-6 in reverse order:

Step 6: $g^x = \left(f^{[n^{-1} \mod p'q'][r^{-1} \mod p'q']}\right)^{-1} \mod n^2$. This means that $f = g_1^{-nxr}$.
Step 5: $f = g_1^{-nr'}\hat{m}^n \mod n^2$. Thus, $g_1^{-nxr} = g_1^{-nr'}\hat{m}^n \mod n^2$, and $\hat{m}^n = g_1^{nr'-nxr}$.

Therefore, regarding $\hat{m}$, we conclude that

$$(1)\ \hat{m}^n = g_1^{nr'-nxr} \text{ and } (2)\ \hat{m}^{\phi(n)} = 1 + [\phi(n)(s' - sv) \mod n] \cdot n \mod n^2.$$

Since at Step 1, G_{ext} checks whether $\hat{m} \in \mathcal{Q}_{n^2}$, we have that $\hat{m} = g^{r^*}(1+n)^{s^*}$, for some $r^* \in [p'q']$ and $s^* \in [n]$. Therefore, $\hat{m}^n = g_1^{nr^*}$. By condition (1) , we have that

$$g_1^{nr^*} = g_1^{nr'-nxr}.$$

Therefore, $nr^* = n(r' - xr) \mod p'q'$. Since $gcd(n, \phi(n)) = 1$, we have that $r^* = (r' - xr) \mod p'q'$. Next, $\hat{m}^{\phi(n)} = (1+n)^{s^*\phi(n)} \mod n^2$. By (2), we have that

$$(1+n)^{s^*\phi(n)} \mod n^2 = (1+n)^{[\phi(n)(s'-sv) \mod n]} \mod n^2.$$

Therefore, $s^*\phi(n) = \phi(n)(s' - sv) \mod n$. Since $gcd(n, \phi(n)) = 1$, this holds only if $s^* = s' - sv \mod n$. □

B Proof of Lemma 6

Proof. We structure our proof by using a sequence of games. Without loss of generality, we assume that $\mathcal{A}$ makes q_1 queries to the $\mathsf{ChallengeOracle}$, $q_2 \leq q_1$ queries to the $\mathsf{CorruptOracle}$ and q_3 queries to the $\mathsf{ExtractOracle}$.

Game $\mathcal{G}_0$: $\mathcal{G}_0$ is the game $\mathcal{G}^{\mathsf{unrmv}}_{\rho,\mathcal{A}}(1^\lambda)$.
Game $\mathcal{G}_1$: $\mathcal{G}_1$ is the same as $\mathcal{G}_0$ except the following: After the adversary outputs a circuit C^* at Step 3 of the unremovability game, $\mathcal{G}_0$ the challenger runs as follows: (a) For $j \in \{1, \ldots, q_1\}$, the challenger runs the Test algorithm of Fig. 7 on inputs C^* and Dec_{sk_j} and checks if there is j s.t. Test algorithm returns 1. (b) Checks whether $\mathsf{Extract}(xk, C^*) \neq \tau_j$. If both conditions (a), (b), are satisfied then the adversary wins.

Game $\mathcal{G}_2$: $\mathcal{G}_2$ is the same as $\mathcal{G}_1$ except that the PRF function F is substituted by a random function in the following way: Whenever $v_1 = F(k_{\mathsf{p}}, (g_1^x, \tau))$ is computed for the first time in $\mathcal{G}_0$, the Challenger chooses $v_1 \xleftarrow{\$} \mathbb{Z}_n$ and stores $\big((g_1^x, \tau), v_1\big)$, so that in any subsequent call of $F(k_{\mathsf{p}}, (g_1^x, \tau))$ the only correct answer is v_1. In addition to the table Marked which stores the public-secret key pairs generated through $\mathsf{ChallengeOracle}$ queries, $\mathsf{WM.Gen}$ initializes a table Marked' as empty. Below we describe how the simulation of the adversary's queries is modified.

- $\mathsf{ChallengeOracle}$ queries: On the t-th query to the $\mathsf{ChallengeOracle}$,for a tag $\tau_t \in \mathcal{T}$, the Challenger chooses $x_t \xleftarrow{\$} [n/4]$, $v_{t,1} \xleftarrow{\$} \{0,1\}^{\lambda/2}$ and computes $v_{t,2} = \mathsf{S.Enc}(k_{\mathsf{e}}, H(v_{t,1})||\tau_t)$. Then, the Challenger computes $v_t = v_{t,1}||v_{t,2}$ and $pk_t = g_1^{x_t}(1+n)^{v_t}$. It stores (pk_t, sk_t) in the t-entry of the table Marked and also it stores $\big((y_t, \tau_t), v_{t,1}, v_{t,2}\big)$ in the table Marked', where $y_t = g_1^{x_t}$. It returns pk_t to $\mathcal{A}$.
- $\mathsf{CorruptOracle}$ queries: On the t-th query to the $\mathsf{CorruptOracle}$, the Challenger retrieves the t-th entry of the table Marked and returns (pk_t, sk_t, τ_t) to $\mathcal{A}$. Then, it stores $\big((y_t, \tau_t), v_{t,1}, v_{t,2}\big)$ to the table $\mathsf{Corrupted}'$.
- $\mathsf{ExtractOracle}$ queries: On input a circuit C, instead of running $\mathsf{Extract}(xk, C)$, it runs $\mathsf{Extract}_1(xk, C)$ which works in the same way as $\mathsf{Extract}$ except that G_{ext} is substituted by a modified algorithm G^1_{ext} which performs as G_{ext} except for the Step 8:
  Step 8 of G^2_{ext}: If $\big(h_i||\tau_i \neq \perp \ \wedge H(v_{i,1}) = h_i \ \wedge \exists\big((y_i, \tau_i), v_{i,1}\big) \in \mathsf{Marked}'\big)$ return $\big((y_i, \tau_i), v_{i,1}\big)$, otherwise return $\perp$.

Game $\mathcal{G}_3$: $\mathcal{G}_3$ is the same as $\mathcal{G}_2$ except the following. First, in the beginning a set S is initialized as empty. For any query τ_t to the $\mathsf{ChallengeOracle}$, the value $v_{t,2}$ generated by the Challenger is added to the set S. Second, $\mathsf{Extract}_2$ is substituted by $\mathsf{Extract}_3$ with the the following modifications. Step 8 of the G^2_{ext} algorithm (which is now renamed as G^3_{ext}) is split into two steps as follows:
Step 8 of G^3_{ext}:
(8a) If $\big(h_i||\tau_i \neq \perp \ \wedge v_{i,2} \notin S\big)$ return $\perp$.
(8b) If $\big(H(v_{i,1}) = h_i \wedge \exists\big((y_i, \tau_i), v_{i,1}\big) \in \mathsf{Marked}'\big)$ return $\big((y_i, \tau_i), v_{i,1}\big)$. Otherwise return $\perp$.

Game $\mathcal{G}_4$: Game $\mathcal{G}_4$ performs exactly as $\mathcal{G}_3$ except that $\mathsf{Extract}_3$ algorithm is substituted by the $\mathsf{Extract}_4$ algorithm where a modified algorithm G^4_{ext} is run. In plain words, Step 8 of $G^4_{\mathsf{ext}}(xk, C, \psi)$ is omitted and at Step 7, decryption of the value $v_{i,2}$ is not performed. Instead, it is simply checked whether the extracted record exists in the table Marked', i.e. whether it corresponds to one of the issued secret keys. Step 7 is described below:
Step 7 of G^4_{ext}: Split the bit representation of v_i into two parts of $\lambda/2$ bits each, i.e. $v_i = v_{i,1}||v_{i,2}$. If there is a record $\big((y_i, \tau_i), v_{i,1}, v_{i,2}\big)$ in Marked', return $\big((y_i, \tau_i), v_{i,1}\big)$.

Game $\mathcal{G}_5$: Game $\mathcal{G}_5$ is the same as $\mathcal{G}_4$ except the following: Any call to $\mathsf{Extract}_4$ algorithm is substituted by a call to an algorithm $\mathsf{Extract}_5$, where G^4_{ext}

is substituted by an algorithm G^5_{ext} described in Figure 8. Recall that q_1 denotes the number of queries issued to the ChallengeOracle and that each public-secret key pair (pk_i, sk_i), where $i \in \{1, \ldots, q_1\}$, which is generated through a ChallengeOracle query is stored in the table Marked. The algorithm G^5_{ext}, on input a circuit C and a "extraction ciphertext" ψ_i, computes in advance how ψ_i is decrypted by each secret key and then checks if the output $C(\psi_i)$ matches any of the decrypted values.

$\underline{G^5_{\mathsf{ext}}}$: On input C and ψ_i,
(1) $\mathsf{Temp} \leftarrow [\,]$.
(2) For $j \in \{1, \ldots, q_1\}$
  – Compute $\mathsf{Dec}_{sk_j}(\psi_i) = \hat{m}_{i,j}$ and set $\mathsf{Temp}[j] = \hat{m}_{i,j}$.
(3) Run $C(\psi_i) = m^*$. If there is $j \in \{1, \ldots, q_1\}$ s.t. $m^* = \hat{m}_{i,j}$ return $(g_1^{x_j}, v_j)$.

Fig. 8: The modified algorithm G^5_{ext}.

Game $\mathcal{G}_6$: $\mathcal{G}_6$ is the same as $\mathcal{G}_5$ except the following. After the completion of the query phase where the Challenger responds to the queries of the adversary, the Challenger chooses $j^* \xleftarrow{\$} \{1, \ldots, q_1\}$, where q_1 is the number of ChallengeOracle queries. The winning conditions are modified as follows. An adversary $\mathcal{A}$ wins if it outputs C^* s.t.
– C^* is ρ-close to $\mathsf{Dec}_{sk_{j^*}}$ and
– $\mathsf{Extract}_6(xk, C^*) \neq \tau_{j^*}$.
We note that $\mathsf{Extract}_6$ algorithm is the same as $\mathsf{Extract}_5$, however we change the notation for consistency in the description of the hybrid games.

Game $\mathcal{G}_7$: The game $\mathcal{G}_7$ is the same as $\mathcal{G}_6$ except the following. Upon receiving C^* from an adversary after the query phase, the Challenger instead of running $\mathsf{Extract}_6$ on input C^*, it runs $\mathsf{Extract}_7$ which first, instead of sampling ℓ extraction ciphertexts, it samples 2ℓ extraction ciphertexts. For the first ℓ extraction ciphertexts it checks if there is $sk_j \neq sk_{j^*}$ that decrypts correctly at least ℓ^* ciphertexts of those ciphertexts. For the remaining ℓ extraction ciphertexts, it checks if at least ℓ^* ciphertexts are correctly decrypted under the key sk_{j^*}. $\mathsf{Extract}_7$ is defined below.
$\underline{\mathsf{Extract}_7}$: On input a circuit C,
(D1). $\mathsf{Count} \leftarrow [\,]$.
(D2). Set $\ell = \frac{\lambda}{\delta^2}$ and $\ell^* = \frac{\lambda}{2\delta^2}$.
(D3). For $i = 1$ to ℓ:
  i. Choose $r, r' \xleftarrow{\$} [n/4]$, $s, s' \xleftarrow{\$} \mathbb{Z}_n$. Compute $\psi_i = \langle g_1^r, (1+n)^s, g_1^{r'}(1+n)^{s'} \rangle$.
  ii. Choose $r_1, r_1' \xleftarrow{\$} [n/4]$, $s_1, s_1' \xleftarrow{\$} \mathbb{Z}_n$. Compute $\psi_i' = \langle g_1^{r_1}, (1+n)^{s_1}, g_1^{r_1'}(1+n)^{s_1'} \rangle$.
  iii. Run G^7_{ext} of Figure 9 on inputs C,ψ_i, ψ_i'.

iv. If G^7_{ext} returns (y_i, v_i)
  – If there is a record $\big((y_i, v_i), \mathsf{count}_i\big)$ in the table Count, set $\mathsf{count}_i \leftarrow \mathsf{count}_i + 1$, otherwise insert a record $\big((\hat{y}_i, \hat{v}_i), \mathsf{count}_i\big)$ with $\mathsf{count}_i = 1$.

(D4). If there is a record $\big(v_i, y_i), \mathsf{count}_i\big)$ in Count s.t. $\mathsf{count}_i \geq \ell^*$, then return marked, else, return $\mathsf{unmarked}$.

$\underline{G^7_{\mathsf{ext}}}$: On input C and ψ_i,
(a) $\mathsf{Temp} \leftarrow [\,]$.
(b) For $j \in \{1, \ldots, q_1\} \setminus \{j^*\}$
  – Compute $\mathsf{Dec}_{sk_j}(\psi_i) = \hat{m}_{i,j}$ and set $\mathsf{Temp}[j] = \hat{m}_{i,j}$.
(c) Compute $\mathsf{Dec}_{sk_{j^*}}(\psi'_i) = \hat{m}_{i,j^*}$.
(d) Run $C(\psi_i) = m^*$. If $m^* \neq \perp$ and there is $j \in \{1, \ldots, q_1\} \setminus \{j^*\}$ s.t. $m^* = \hat{m}_{i,j}$ return $(g_1^{x_j}, v_j)$. Otherwise, run $C(\psi'_i) = m'$. If $m' = \hat{m}_{i,j^*}$ return $(g_1^{x_{j^*}}, v_{j^*})$.

Fig. 9: The modified algorithm G^7_{ext}.

Next, we define a sequence of ℓ games. For any $i \in \{1, \ldots, \ell\}$, $\mathcal{G}_{7,i}$ is defined as follows. By $\mathcal{G}_{7,1}$ we denote the game $\mathcal{G}_7$ described above.

Game $\mathcal{G}_{7,i}$: $\mathcal{G}_{7,i}$ differs with $\mathcal{G}_{7,i-1}$ only in the way the ciphertext at i-th ciphertext of Step 7((D3))ii is selected. In the game $\mathcal{G}_{7,i}$, the i-th ciphertext is an encryption of a random plaintext under the key pk_j while in the game $\mathcal{G}_{7,i-1}$ it is an "extraction ciphertext".

Security Analysis.

Claim 7.1: *If there is a PPT adversary breaking the game $\mathcal{G}_0$ with non-negligible probability, then there is a PPT adversary breaking the game $\mathcal{G}_1$ with non-negligible probability.*

Proof of Claim 7.1: Assuming that an adversary $\mathcal{A}$ wins the game $\mathcal{G}_0$ with non-negligible probability α, then $\mathcal{A}'$ breaks the game $\mathcal{G}_1$ with probability $\alpha - \mathsf{negl}(\lambda)$, based on Proposition 3. This is straightforward since the Test algorithm of Fig. 7 will return 1 if $\mathsf{cnt} \geq \frac{\lambda}{2\delta^2}$ and based on Proposition 3, if $C^* \sim_\rho \mathsf{Dec}_{sk}$ (which was the original winning condition of $\mathcal{G}_0$), then $\mathsf{Pr}[\mathsf{cnt} < \frac{\lambda}{2\delta^2}] = \mathsf{negl}(\lambda)$.
(End of Claim 7.1) ⊣

Claim 7.2: *If there is a PPT adversary breaking the game $\mathcal{G}_1$ with non-negligible probability, then there is a PPT adversary breaking the game $\mathcal{G}_2$ with non-negligible probability.*

Proof of Claim 7.2:

We will prove that if there is a PPT adversary A winning the game $\mathcal{G}_1$, with non-negligible probability then there is an adversary that wins $\mathcal{G}_2$ with non-negligible probability unless the PRF assumption is violated. Then, we construct a PPT distinguisher D which distinguishes between a PRF and a random function, thus breaking Definition 3 as follows.

Distinguisher D with oracle access to an oracle $\mathcal{O} : \mathcal{X} \to \{0,1\}^{\lambda/2}$:

- D runs an adversary $\mathcal{A}$ playing the role of the challenger in the game $\mathcal{G}_1$: When simulating a query to the $\mathsf{ChallengeOracle}$, e.g. the t-th query, D chooses a value $x_t \xleftarrow{\$} [n/4]$, computes $g_1^x \mod n^2$ and issues a query $(g_1^{x_t}, \tau_t)$ to the oracle $\mathcal{O}$. Upon receiving a value $v_{t,1}$, D proceeds as described in the game $\mathcal{G}_1$. When D simulates a query to the $\mathsf{ExtractOracle}$ for a circuit C, it runs $\mathsf{Extract}(params, xk, C)$ with the following difference: At Step 8 of the $G_{\mathsf{ext}}(params, xk, C, \psi_i)$ algorithm, where $i \in \{1, \ldots, \ell\}$, if a record $\big((g_1^{x_i}, \tau_i), v_{i,1}\big)$ has been extracted and $\big((g_1^{x_i}, \tau_i), v_{i,1}\big) \notin \mathsf{Marked}'$, then D issues a query $\mathcal{O}(g_1^{x_i}, \tau_i)$ which returns $v_{i,1}^*$. If $v_{i,1}^* = v_{i,1}$, it sets $\mathsf{count}_i = 1$ and inserts $\big((g_1^{x_i}, \tau_i), v_{i,1}, \mathsf{count}_i\big)$ to the table Count. Otherwise G_{ext} returns $\perp$.
- $\mathcal{A}$ outputs C^*.
- It runs Steps (a),(b) of the algorithm $\mathcal{G}_0'$ If both conditions are satisfies then D outputs 1, else it outputs 0.

Now, assume that D has access to the PRF oracle $F(k_{\mathsf{p}}, \cdot)$, where $k_{\mathsf{p}} \xleftarrow{\$} \mathcal{K}$. Then it is easy to see that D interacts with $\mathcal{A}$ exactly as in the game $\mathcal{G}_1$.

$$\mathsf{Pr}_{k_{\mathsf{p}} \xleftarrow{\$} \mathcal{K}}[D^{F(k_{\mathsf{p}}, \cdot)}(1^\lambda) = 1] = \mathsf{Pr}[\mathcal{G}_1 = 1]. \tag{3}$$

If D has oracle access to a random function, this means that in each run of G_{ext} algorithm, the adversary can only guess the same $v_{i,1}$ value sampled by the oracle $\mathcal{O}$ with probability $1/2^{\lambda/2}$. Since $q_3 + 1$ queries are issued to the $\mathsf{ExtractOracle}$, the probability that a "valid" $v_{i,1}$ is guessed by $\mathcal{A}$ equals $\frac{(q_3+1)\ell}{2^{\lambda/2}}$, which is negligible in λ. This means that

$$\mathsf{Pr}_{k_{\mathsf{p}} \xleftarrow{\$} \mathcal{K}}[D^{f(\cdot)}(1^\lambda) = 1] = \mathsf{Pr}[\mathcal{G}_2 = 1] + \frac{(q_3 + 1)\ell}{2^{\lambda/2}}. \tag{4}$$

By the Definition of a PRF, we have that for any PPT adversary $\mathcal{A}$ it holds that $\big|\mathsf{Pr}[\mathcal{G}_1 = 1] - \mathsf{Pr}[\mathcal{G}_2 = 1]\big| \leq \mathsf{negl}(\lambda)$. *(End of Claim 7.2)* $\dashv$

Claim 7.3: The games $\mathcal{G}_2$, $\mathcal{G}_3$ are computationally indistinguishable if the symmetric encryption scheme $(\mathsf{S.Gen}, \mathsf{S.Enc}, \mathsf{S.Dec})$ satisfies integrity of ciphertexts (cf. Definition 4).

Proof of Claim 7.3: Due to the structure of the games $\mathcal{G}_2$, $\mathcal{G}_3$, the only way that a PPT adversary $\mathcal{A}$ distinguishes between $\mathcal{G}_2$ and $\mathcal{G}_3$ is if the following event takes place:

Bad = "$\mathcal{A}$ issues a circuit C s.t. $\mathsf{Extract}_2(xk, C) = \tau_i$ but $\mathsf{Extract}_3(xk, C) = \perp$."

Recall (1) that τ_i is one of the tags issued previously as input for a $\mathsf{ChallengeOracle}$ query and (2) that we have assumed that $\mathcal{A}$ issues q_3 queries to the $\mathsf{ExtractOracle}$ and in the end of the (modified) unremovability games it outputs a circuit C^*. We will prove that $\mathsf{Pr}[\mathsf{Bad}] = \mathsf{negl}(\lambda)$. Let $\mathcal{A}$ be a PPT adversary against game $\mathcal{G}_2$, we will construct a PPT adversary $\mathcal{B}$ against the $\mathcal{G}_{\mathcal{B}}^{\mathsf{int-ctxt}}$ game (cf. Figure 1) as follows:

1. $\mathcal{B}$ generates all the parameters of the watermarking scheme by running the $\mathsf{WM.Gen}(1^\lambda)$ excluding the symmetric key k_e. $\mathcal{B}$ chooses $i^* \xleftarrow{\$} \{1, \ldots, q_3+1\}$ and provides to $\mathcal{A}$ the public parameters $params = (n, g_1)$.
2. When $\mathcal{A}$ queries the $\mathsf{ChallengeOracle}$ on input a tag τ_i, $\mathcal{B}$ runs the $\mathsf{PKE.Mark}$ algorithm with the difference that it computes the value $v_{i,2}$ by issuing a query to the Challenger of the integrity of ciphertexts game with input $H(v_{i,1})||\tau_i$. $\mathcal{B}$. Upon receiving a ciphertext $v_{i,2}$ from the Challenger, it stores $v_{i,2}$ to a set S', computes the public key pk_i returns it to $\mathcal{A}$.
3. Regarding the queries of $\mathcal{A}$ to the $\mathsf{ExtractOracle}$, they are answered as follows: On input C, $\mathcal{B}$ runs the algorithm $\mathsf{Extract}_2$ with the following differences: For any ψ_j sampled at Step (D3)a, $\mathcal{B}$ runs $G_\mathsf{ext}(xk, C, \psi_j)$ and at Step 7, $\mathcal{B}$ checks:
   (a) if $v_{i,2} \in S'$ then returns to $\mathcal{A}$ the corresponding plaintext, previously issued as a query the integrity of ciphertexts game.
   (b) if $v_{i,2} \notin S'$ and this is not the i^*-th query to the $\mathsf{ExtractOracle}$ stop the game.
   (c) if $v_{i,2} \notin S'$ and this is the i^*-th query to the $\mathsf{ExtractOracle}$ then $\mathcal{B}$ returns $v_{i,2}$ as the ciphertext/output of the integrity of ciphertexts game.

We denote as Bad' the event that $v_{i,2} \notin S$ is extracted by the G_ext algorithm in some of the $(q_3+1)\ell$ runs. It is easy to see that only if the event Bad' takes place, the event Bad can happen. Based on the above, $\mathsf{Pr}[\mathcal{B} \text{ wins}] \geq \mathsf{Pr}[\mathsf{Bad}']/(q_3+1)$. Based on that, $\mathsf{Pr}[\mathsf{Bad}'] \leq \mathsf{negl}(\lambda)$. *(End of Claim 7.3)* $\dashv$

Claim 7.4: The games $\mathcal{G}_3$, $\mathcal{G}_4$ are computationally indistinguishable assuming that the hash function H is modeled as a random oracle.

Proof of Claim 7.4: An adversary could distinguish between the games $\mathcal{G}_3$, $\mathcal{G}_4$ two queries on input the same tag τ, are issued to the challenge oracle, e.g. the i-th and the j-th query, and at the same time $H(v_{i,1}) = H(v_{j,1})$. Due to the fact that H is collision resistant, this probability equals $1/2^{\kappa_3}$.

(End of Claim 7.4) $\dashv$

Claim 7.5: *The games $\mathcal{G}_4$, $\mathcal{G}_5$ are indistinguishable due to Lemma 5.*
Proof of Claim 7.5: Recall that if G_ext (and consequently G^4_ext) on input a circuit C and a extraction ciphertext ψ_i returns a pair $(g_1^{x_j}, v_j)$, this means that $(g_1^{x_j}, v_j) \in \mathsf{Marked}^*$ and therefore $sk_j = (x_j, v_j)$ has been previously generated through a query to the $\mathsf{ChallengeOracle}$. Lemma 5 shows that if C on input ψ_i returns a pair $(g_1^{x_j}, v_j)$, then it holds that $C(\psi_i) = \mathsf{Dec}_{sk_j}(\psi_i)$. Based on that, instead of applying Steps 1-7 of G^4_ext, it is equivalent to check whether there is sk_j that has been previously generated s.t. $C(\psi_i) = \mathsf{Dec}_{sk_i}(\psi_i)$. This is the check that is performed by G^5_ext algorithm (cf. Figure 8) which is run by $\mathsf{Extract}_5$ algorithm in the game $\mathcal{G}_5$. *(End of Claim 7.5)* $\dashv$

Claim 7.6: *If there is a PPT adversary breaking the game $\mathcal{G}_5$ with non-negligible probability, then there is a PPT adversary breaking the game $\mathcal{G}_6$ with non-negligible probability.*
Proof of Claim 7.6: Since the index j^* is chosen uniformly at random, $\mathcal{C}$ guesses the correct index j targeted by the adversary for the "unremovability attack"

with probability $1/q_1$ where q_1 is the number of queries to the $\mathsf{ChallengeOracle}$. *(End of Claim 7.6)* $\dashv$

Given Claim 7.6, it suffices to prove that for any PPT adversary $\mathcal{A}$, $\mathsf{Pr}[\mathcal{G}_6 = 1] = \mathsf{negl}(\lambda)$. We will prove this by the following claims. First, note that the games $\mathcal{G}_6$, $\mathcal{G}_{7,0}$ are indistinguishable since we assume that every circuit is reset after producing an output on a particular input. Therefore, since the inputs sampled by $\mathsf{Extract}_{7,0}$ follow the same distribution with the inputs sampled by $\mathsf{Extract}_6$, an adversary is not able to distinguish between the games $\mathcal{G}_4$ and $\mathcal{G}_{7,0}$.

Claim 7.7: *For any $i \in \{1, \ldots, \ell\}$, the games $\mathcal{G}_{7,i}$, $\mathcal{G}_{7,i-1}$ are computationally indistinguishable due to ciphertext indistinguishability (i.e. Lemma 4).*
Proof of Claim 7.7: We will show that if $\mathcal{G}_{7,i}$, $\mathcal{G}_{7,i-1}$ were not computationally indistinguishable, then Lemma 4 does not hold. Specifically, we will construct a PPT distinguisher $\mathcal{B}$ which distinguishes the tuples $\langle n, g_1, x, v, g_1^r, (1+n)^r, h^r \cdot m\rangle$, $\langle n, g_1, x, v, g_1^{r_1}, (1+n)^{s_1}, g_1^{r_2}(1+n)^{s_2}\rangle$.

$\mathcal{B}$: On input $\langle n, g_1, x, v, a, b, c\rangle$,

1. Compute $pk = g_1^x(1+n)^v$ and set $sk = (x, v)$.
2. $\mathcal{B}$ chooses $j \xleftarrow{\$} \{1, \ldots, q_1\}$ and set $pk_j = pk$ and $sk_j = sk$.
3. $\mathcal{B}$ interacts with a PPT adversary $\mathcal{A}$ by playing the role of the Challenger as in the modified unremovability game $\mathcal{G}_{7,i}$. $\mathcal{B}$ responds to the queries of $\mathcal{A}$ in by performing the same computations as the Challenger in the game $\mathcal{G}_{7,i}$, except that in j-th query to the $\mathsf{ChallengeOracle}$, $\mathcal{B}$ stores the pair (pk_j, sk_j) to the table Marked and returns pk_j to $\mathcal{A}$.
4. Upon receiving C^* from $\mathcal{A}$, $\mathcal{B}$ runs $\mathsf{Extract}_{7,i}$ in input C^* but with the following difference: At the i-th iteration of the Step 7((D3))ii $\mathcal{B}$ sets $\psi_i' = \langle a, b, c\rangle$ and provides ψ_i' as input to C^*. If the above algorithm returns marked, then $\mathcal{B}$ returns 1, otherwise $\mathcal{B}$ returns 0.

Analysis of the reduction. Assume that $\mathcal{B}$ receives an input of the form $\langle n, g_1, x_j, v_j, g_1^r, (1+n)^r, h_j^r \cdot m\rangle$. This implies that $\mathcal{B}$ simulates perfectly the game $\mathcal{G}_{7,i}$ in the eyes of $\mathcal{A}$. Hence,

$$\mathsf{Pr}[\mathcal{B}(n, g_1, x_j, v_j, g_1^r, (1+n)^r, h_j^r \cdot m) = 1] = \mathsf{Pr}[\mathcal{G}_{7,i} = 1]. \tag{5}$$

If $\mathcal{B}$ receives an input of the form $\langle n, g_1, x_j, v_j, g_1^{r'}, (1+n)^s, g_1^{r''}(1+n)^{s''}\rangle$, this means that $\mathcal{B}$ simulates the game $\mathcal{G}_{7,i-1}$ in the eyes of $\mathcal{A}$. Therefore,

$$\mathsf{Pr}[\mathcal{B}(n, g_1, x_j, v_j, g_1^{r'}, (1+n)^s, g_1^{r''}(1+n)^{s''}) = 1] = \mathsf{Pr}[\mathcal{G}_{7,i-1} = 1]. \tag{6}$$

By (5), (6), it follows that $|\mathsf{Pr}[\mathcal{G}_{7,i} = 1] - \mathsf{Pr}[\mathcal{G}_{7,i-1} = 1]| \leq \mathsf{negl}(\lambda)$.
(End of Claim 7.7) $\dashv$

Claim 7.8: *For any PPT adversary $\mathcal{A}$, it holds that $\mathsf{Pr}[\mathcal{G}_{7,\ell} = 1] = \mathsf{negl}(\lambda)$.*
Proof of Claim 7.8: This claim holds due to Proposition 3. *(End of Claim 7.8)* $\dashv$