Watermarking public-key cryptographic functionalities and implementations: The case of encryption and signatures

Foteini Baldimtsi\textsuperscript{1} | Aggelos Kiayias\textsuperscript{2} | Katerina Samari\textsuperscript{3}

Abstract
A watermarking scheme for a public-key cryptographic functionality enables the embedding of a mark in the instance of the secret-key algorithm such that the functionality of the original scheme is maintained, while it is infeasible for an adversary to remove the mark (unremovability) or mark a fresh object without the marking key (unforgeability). A number of works have appeared in the literature proposing different definitional frameworks and schemes secure under a wide range of assumptions. In the previous work [1, 2], the authors proposed a meaningful relaxation of the watermarking model and gave constructions that allow direct watermarking of popular cryptographic schemes (e.g. ElGamal Encryption). A definitional framework for watermarking public-key cryptographic functionalities and implementations which covers both deterministic (e.g. decryption) and probabilistic (e.g. signing) secret-key algorithms is provided. The authors’ work unifies the previous results of [1, 2] where deterministic and probabilistic circuits to be watermarked as separate cases are considered. The constructions of [1, 2] were previously presented as extended abstracts missing rigorous security proofs. The authors prove those constructions secure under their new, unified framework. In the authors’ schemes secret detection of the watermark is provided, and security under minimal hardness assumptions assuming only the existence of one-way functions, is proved.

1 | INTRODUCTION

Watermarking is a powerful tool widely used in practice to secure the copyrighted material. Watermarking digital objects like pictures, video or software is usually achieved by embedding a special piece of information, the mark, into the object so that it is difficult for an adversary to remove it without damaging the object itself (unremovability). At the same time, the embedding of the mark should not result in a significantly different object, or an object with a different functionality. A plethora of watermarking schemes exists in the literature [3–6] (and references therein), most of them focusing on watermarking ‘static’ or else ‘perceptual’ objects (e.g. images) and formal security definitions for watermarking ‘perceptual’ objects have been given by Hopper et al. [7].

Besides perceptual objects, there has been a recent focus on software watermarking and precisely on watermarking cryptographic functionalities. Watermarking cryptographic functions have various real-life applications. Consider for instance, the case of the VPN clients. An organization might wish to distribute VPN clients to its employees where each employee is assigned a public/secret-key pair. Watermarking a VPN client could potentially restrict an employee from sharing its client for the following reason: Due to the unremovability and unforgeability properties, if one is given any client, she would be able to detect to whom the client belongs, assuming that the ID of the user is embedded as the watermark. The first rigorous definitions for software watermarking were given by Barak et al. [8, 9]. Informally speaking, according to [8, 9], a marking algorithm, marks a program/circuit C by producing a new circuit C’, which does not alter the functionality of C. Then, a detection algorithm can deduce whether a circuit is marked or not, for any circuit which is given as an input. The basic security requirement that should be satisfied by a watermarking scheme is fragility/unremovability which requires that no polynomial time adversary should be able to...
remove the ‘mark’ from a marked circuit \( C \), unless it substantially changes its functionality. Having set their definitional framework, in [8, 9], the authors explore the relation of software watermarking with the notion of indistinguishability obfuscation [iO] [8–10] and provide an impossibility result. In particular, they show that if a marked circuit \( C \) has exactly the same functionality as the original circuit \( \tilde{C} \), then under the assumption that iO exists, watermarking is impossible.

Nishimaki [11] (cf. also [12]), inspired by the definitions of watermarking given by Hopper et al. [7], suggests a model for watermarking cryptographic functions (i.e. functions that are characterized by some key), and proposes the first watermarking scheme for Lossy Trapdoor functions [13]. Apart from the notion of non-removability, similarly to [7], the notion of unforgeability is introduced for the case of cryptographic functions. Informally, unforgeability captures that no adversary can mark functions on its own, without having access to the marking key. In [11], the impossibility result of Barak et al. [8, 9] is circumvented by restricting the adversaries in the security games to output functions that preserve a specific format. We note that Naccache et al. [14] considered the problem of ‘copy-righting’ public-key encryption schemes before [8, 9, 11], however, similarly to [11], they also place restrictions to the adversary’s outputs.

Cohen et al. [15, 16] (merged result of [17, 18]) were the first that considered general adversaries. Motivated by the fact that the impossibility result [8, 9] does not hold if a marked circuit is approximately close to the original unmarked one, we propose a watermarking scheme for any family of puncturable pseudorandom functions (PRFs) [19–22]. Their construction satisfies unremovability without placing any restrictions to the adversary’s strategy based on the strong assumption that iO exists. In addition, it supports public detection, which means that any entity can detect whether a circuit is marked or extract a specific message from a circuit based only on public information. We note that the notion of unforgeability is not considered in [15], however some preliminary results related to this notion appeared in [17]. In particular, the construction of [17] for puncturable PRFs satisfies a weaker notion of unforgeability called relaxed unforgeability. Interestingly, a recent work of Yang et al. [23] shows how to achieve standard unforgeability having as a starting point the construction in [17]. Cohen et al. [15, 16] also describe how to construct ‘Watermarkable Public-key Encryption’ and ‘Watermarkable Signatures’. This is accomplished by using the result of Sahai and Waters [19], who construct public-key encryption schemes and digital signature schemes based on iO by setting the decryption and signing algorithms to be essentially just an evaluation of a puncturable PRF.

Boneh et al. [24] construct a watermarkable family of PRFs based on a primitive called private programmable PRFs, which they instantiate by relying on iO. Kim and Wu [25] overcome the burden of relying on a ‘heavy’ assumption like iO and suggest a watermarking construction for a family of PRFs based on standard lattice assumptions. Quach et al. [26] provided a watermarking construction for a family of PRFs which relies on standard assumptions and supports public marking, a property which allows any entity to mark itself a function of the family, instead of relying on a trusted authority. Additional constructions for watermarking PRFs from standard assumptions were given in [27, 28]. Yang et al. [29] introduce the notion of collusion-resistant watermarking, where an adversary is capable of receiving multiple watermarked copies of the same circuit embedded with different messages. The authors provide a watermarking scheme for PRFs and based on that and the constructions of Sahai and Waters [19] propose constructions for primitives like public-key encryption, digital signatures, symmetric-key encryption and message-authentication codes.

Baldimtsi et al. [1] propose an alternative, more relaxed model for watermarking which is focused on watermarking public-key cryptographic functionalities. Based on the proposed model, the authors provide watermarking schemes based on minimal hardness assumptions. In particular, they provide constructions for public-key encryption with private marking and both private and public detection are proposed assuming one-way functions and identity-based encryption (IBE) [30], respectively. In a follow-up work, Baldimtsi et al. [2] modify the model of [1] to be suitable for capturing probabilistic circuits, such as probabilistic signing algorithms, and propose a watermarking construction for a class of digital signature schemes.

On the front of watermarking the public-key cryptographic primitives, Goyal et al. [31] propose two schemes: (1) a private marking and private detection (with message extraction) watermarking scheme for attribute-based encryption (ABE) from mixed functional encryption (FE) and delegatable ABE, which can be instantiated under the LWE assumption, and (2), a public marking and public extraction scheme for predicate encryption from (bounded collusion-resistant) hierarchical FE, which can be instantiated by any PKE. The constructions of [31] are highly inefficient as they rely on heavy tools like mixed FE and hierarchical FE. Very recently, Nishimaki [32] presented a framework to equip the public-key cryptographic schemes with watermarking functionalities without the need for additional security assumptions beyond the ones assumed by the PK primitive, under the limitation that the underlying public-key primitive needs to have an all-but-one security reduction (as defined in [32]) and only provide a selective security (i.e. adversaries must commit a target attribute at the beginning of the security game).

Related notions to watermarking Leakage-deterring public-key cryptography, as defined in [33], which captures the idea that some personal information is embedded to the public key of a user such that, if she decides to share her secret key (or a partial working implementation of her decryption function) the recipient can extract the private information embedded in the public key. Although related to watermarking, leakage-deterring cryptography focuses on private information embedding in a cryptosystem that remains hidden unless the secret key is shared. Privacy is not an issue in watermarking,
thus construction techniques are technically and conceptually different.

Watermarking public-key encryption functionalities is also related to the notion of traitor tracing, which was put forth by Chor et al. [34]. Since then, this primitive has attracted a lot of attention and a multitude of schemes have appeared achieving various properties and secure under different assumptions, for example [35–39]. 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. In our view, the basic difference between the notions of watermarking and traitor tracing is that in traitor tracing schemes, users share functionality as the goal to decrypt the same ciphertext, while in watermarking users should not be able to decrypt the same ciphertexts. This is a key issue in the design of watermarking schemes.

**Our contributions.** The main contribution of our work is to revisit the definitional framework for watermarking public-key cryptographic functionalities considered in our previous works [1, 2] and present a unified definitional model that covers the idea of watermarking both deterministic and probabilistic cryptographic circuits. We present the constructions for watermarking public encryption schemes and digital signatures schemes suggested in [1, 2] and provide a complete and detailed security analysis for the above watermarking constructions under our new, unified definitional model. A rigorous security analysis was not previously included in [1, 2]. Let us recall the watermarking approach of [1]. Motivated by the goal of providing watermarking constructions for public-key cryptographic functionalities (e.g., public-key encryption) without relying on assumptions like iO but only on standard assumptions, Baldimtsi et al. [1] approach cryptographic watermarking, by making a relaxation and refinement to the model considered in previous works (e.g., [15]). First, a small shared state is allowed between the marking and detection algorithms which is publicly available and can be potentially maintained in a distributed ledger. Second, an authority instead of embedding a message in a specific circuit which is given as input (as in [9, 11, 15]), it samples itself a circuit embedded with a message of the client’s choice, or just a marked circuit. Specifically, the watermarking authority embeds a message in the secret key algorithm for the functionality (e.g., decryption algorithm) and returns it to the client together with the corresponding public-key algorithms of the functionality (e.g., encryption algorithm). We note that this formulation is consistent with the specific definitions for ‘Watermarkable Public-Key Encryption’ and ‘Watermarkable Signatures’ previously suggested in [15] and we discuss differences with the more generalized framework of [31] in Section 3.4. We argue that the new definitional framework of [1] maintains all the relevant to practice features that the previous formulations enjoyed, and moreover can be very suitable for some real-world scenarios due to its more refined nature. An important observation that motivates our modelling is that limiting the interaction between the marking system and the recipient of the object in the above fashion is unnecessarily restrictive. In many applications of public-key cryptography, the actual details of the decryption or signing program are not relevant to its user, only its functionality is (which encompasses its correctness and security properties). For instance, in the VPN scenario we described above, the organization (i.e., the marking system) is often the one to sample a key \( K_U \) for its client and provide it along with the VPN client.

A further refinement of the model of [1] is distinguishing between the notions of watermarking cryptographic functionalities and cryptographic implementations. Intuitively, the former notion captures constructions for public-key cryptographic primitives that satisfy the basic properties of watermarking (i.e., detection correctness, functionality property-preserving, unremovability, unforgeability). The latter notion considers a specific cryptographic implementation as a starting point, for example, the El-Gamal encryption scheme, and aims to capture what would it mean to watermark it. In practice, watermarking a specific scheme, which may be widely used may be more useful and make watermarking constructions backwards compatible comparing to constructing new watermarkable schemes.

In a follow-up work [2], we observed that a common characteristic of all previous works is that watermarking constructions concern deterministic circuits. In particular, certain families of PRFs are watermarked [15, 24, 25, 29] while for other primitives like PKE, the decryption circuit is the one which is watermarked [1, 15, 29]. Even for the case of the constructions for digital signatures in [15, 29], the signing circuit is deterministic. In addition, they observe that closeness and farness relations considered in previous works do not seem suitable for the case of probabilistic circuits, as now the output of a circuit is not a specific value but a random variable. Closeness and farness relations are crucial for defining unremovability and unforgeability notions respectively, since when defining the 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. Thus, novel formulations for closeness and farness relations between probabilistic circuits are introduced in [2], that are suitable for defining watermarking security for cryptographic functionalities.

We unify the models of [1, 2] by presenting a framework for watermarking public-key cryptographic functionalities and implementations where the secret key algorithm, which is the watermarked algorithm, may be either deterministic or probabilistic. Then, we present watermarking constructions that show how to watermark any IND-CPA secure public-key encryption scheme and any EUF-CMA secure digital signature scheme that satisfies an additional property called verification soundness. Intuitively, verification soundness captures the verification algorithm will accept exactly the outputs of the
signing algorithm. Interestingly, this property has been considered before but in a different context [40, 41]. Standard digital signature schemes, such as Schnorr signatures, satisfy this property and thus this construction is capable of yielding practical watermarked digital signatures. At a high level, the constructions exploit the notion of a PRF, to create a compact ‘dictionary’ of marked circuits that is subsequently scanned and compared with an adversarial circuit. The proposed constructions are simple, use well-known building blocks and they are proven to satisfy both the unremovability and unforgeability under minimal hardness assumptions. The marking and detection algorithm share a state of logarithmic size and the detection is secret.

2 | PRELIMINARIES

Notation: We first set the notation to be used throughout the paper. By $\lambda \in \mathbb{N}$ we denote the security parameter. The left arrow notation, $x \stackrel{\$}{\leftarrow} X$, denotes that $x$ is chosen uniformly at random from a space $X$. By $x \leftarrow \mathcal{D}$, we denote that $x$ is chosen from the distribution $\mathcal{D}$. The support set of the distribution $\mathcal{D}$ is denoted as $|\mathcal{D}|$. Next, by $C$ we denote an unmarked circuit and by $C$ a watermarked one. By $C(m; r)$ we denote the probabilistic circuit $C$ which runs on input $m$ and randomness $r$. By poly$(\lambda)$, we denote a polynomial in $\lambda$. Finally, the abbreviation PPT stands for the probabilistic polynomial time.

Pseudorandom functions: A function $F : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^t$ is a PRF if for any PPT distinguisher $\mathcal{A}$, $|\Pr[\mathcal{A}^{F(k, \cdot)} = 1] - \Pr[\mathcal{A}^{f(\cdot)} = 1]| \leq \text{negl}(\lambda)$, where $k$ is chosen uniformly at random from $\{0,1\}^k$ and $f$ is chosen uniformly at random from $F = \{ f | f : \{0,1\}^n \rightarrow \{0,1\}^t \}$.

Digital signatures: A digital signature scheme is defined by a triplet of PPT algorithms $(\text{SigGen}, \text{Sign}, \text{Verify})$. The key generation algorithm, $\text{SigGen}$, on the input of the security parameter $1^\lambda$, outputs a pair of signing and verification keys $(sk, vk)$. The signing algorithm, $\text{Sign}$, takes as input a message $m$ from a message space $\mathcal{M}$ and a signing key $sk$ and outputs a signature $\sigma$. Finally, the verification algorithm, $\text{Verify}$, takes as input a verification key $vk$ and a message-signature pair $(m, \sigma)$, and outputs either 0 or 1.

A signature scheme satisfies correctness if for any $(sk, vk)$ output by $\text{SigGen}(1^\lambda)$ and any message $m \in \mathcal{M}$ it holds that $\Pr[\text{Verify}_{vk}(m, \text{Sign}_{sk}(m)) = 1] = 1$.

The standard definition of security for digital signatures is existential unforgeability under chosen message attacks (EUF-CMA):

Definition 1 (EUF-CMA). A signature scheme $(\text{SigGen}, \text{Sign}, \text{Verify})$ is existentially unforgeable under an adaptive chosen-message attack, if for any PPT adversary $\mathcal{A}$, it holds that:

$$\Pr\left[(sk, vk) \leftarrow \text{Gen}(1^\lambda); (m^*, \sigma^*) \leftarrow \mathcal{A}^{\text{Sign}_{sk}(\cdot)}; \text{Verify}_{vk}(m^*, \sigma^*) = 1 \land m^* \notin Q\right] = \text{negl}(\lambda),$$

where $Q$ is the set of queries of $\mathcal{A}$ to the signature oracle $\text{Sign}_{sk}(\cdot)$.

For the security analysis of our watermarking construction (Section 5.2), we require signature schemes to satisfy the property defined below which is called the verification soundness.

Definition 2 (Verification soundness). Let $(\text{SigGen}, \text{Sign}, \text{Verify})$ be a digital signature scheme with message space $\mathcal{M}$, randomness space $\mathcal{R}$ and signature space $\mathcal{S}$. For any pair $(vk, sk)$ output by $\text{SigGen}(1^\lambda)$, we define the relation $R_{sk} = \{(vk, (m, \sigma)) | m \in \mathcal{M}, \exists r \in \mathcal{R} \text{ s.t. } \sigma \leftarrow \text{Sign}_{sk}(m; r)\}$, where $\text{Sign}_{sk} : \mathcal{M} \times \mathcal{R} \rightarrow \mathcal{S}$. We say that a digital signature scheme satisfies verification soundness if for any $(vk, sk) \leftarrow \text{SigGen}(1^\lambda)$ it holds that for any PPT adversary $\mathcal{A}$,

$$\Pr\left[(m, \sigma) \leftarrow \mathcal{A}(vk, sk); (vk, (m, \sigma)) \notin R_{sk} \land \text{Verify}_{vk}(m, \sigma) = 1\right] = \text{negl}(\lambda)$$

Informally, this property forces rejection of a signature that was not generated by the signing algorithm, that is, a PPT adversary given $(vk, sk)$ cannot output $(m, \sigma)$ that verifies correctly when the signature was not computed using the signing algorithm for this particular message and signing key. This stronger requirement was previously considered in the literature in for example [40, 41] and it also holds for other standard signature schemes like Schnorr [42], the case of which we show in the Appendix. We note that verification soundness is incomparable to the EUF-CMA notion for a number of reasons. First, notice that in the verification soundness security game the adversary is also given as input the secret key. Furthermore, according to the EUF-CMA security, the adversary wins if it outputs a valid signature pair for a message that was never queried to the Signing Oracle. However, it is not required that the forged signature/message pair belongs to the relation $R_{sk}$, that is a EUF-CMA adversary could output a winning forged signature that is not a possible output of the signing algorithm for the chosen message.

2.1 | Definitions for cryptographic functionalities and implementations

We now define the notions of cryptographic functionalities and implementations. The goal of the cryptographic functionality definition is to capture a cryptographic object such as an encryption scheme or a PRF, in an abstract ideal way, thus focusing on the properties it should satisfy. On the other hand, the notion of a cryptographic implementation is used to describe a specific scheme that satisfies the properties of the cryptographic functionality (i.e. the ElGamal encryption scheme [43] is an implementation of the public-key encryption functionality).
Definition 3 (Cryptographic functionality). A cryptographic functionality $C_F$ consisting of $m$ algorithms is defined by a set of $n$ properties and their corresponding probabilities $(\mathcal{G}^{\text{prop}}_A, \pi_i)_{i=1}^n$. The first component, $\mathcal{G}^{\text{prop}}_A$, is a security game, which can be seen as a Turing Machine, and takes the following inputs: (1) a special input which among possibly other values includes the security parameter, (2) the description of a PPT algorithm called the adversary and (3) the description of a tuple of $m$ circuits. The security game $\mathcal{G}^{\text{prop}}_A$ outputs either 0 or 1. We say that an adversary ‘breaks’ a property or ‘wins’ a security game if the output of the game is 1. The second component is a value $\pi_i \in [0, 1]$, which is a fixed ideal lower bound under which the property cannot be guaranteed.

Remark 1 In Definition 3, some properties may also hold for super-polynomial or even unbounded adversaries (e.g. correctness-related properties), however, for simplicity we opt to define all the properties with respect to PPT adversaries. In the text below, we will make a clear note when a property is also satisfied with respect to super-polynomial adversaries.

Example: Consider the public-key encryption functionality as an example, which can be defined as a pair of algorithms (Gen, Enc, Dec) that should satisfy the properties of correctness and IND-CPA security. Correctness can be defined as a security game, though this is not traditionally the case, where an adversary is challenged to provide an encryption of a message $M$ which is decrypted to a message different than $M$. In this case, the adversary need not be restricted to run in the polynomial time. The IND-CPA security property is defined in the standard way. Corresponding to our definitions, the games will receive as input the encryption/decryption algorithms for a specific key pair. Given that the definition of a cryptographic functionality describes an ‘ideal’ scenario, correctness would always hold with the probability 0, while in the IND-CPA property, the adversary would have a probability of success of exactly 1/2.

Definition 4 (Cryptographic Implementation). Let $C_F$ be a cryptographic functionality with $m$ algorithms and $n$ properties $(\mathcal{G}^{\text{prop}}_A, \pi_i)_{i=1}^n$. An implementation of the cryptographic functionality $C_F$ consists of an $m$-tuple of algorithms $(\text{Gen}, C_1, \ldots, C_m)$ such that, for every security parameter $\lambda$ and each property $\pi_i$ for $i \in \{1, \ldots, n\}$ and for any corresponding PPT adversary $\mathcal{A}$, it holds that $\Pr[\mathcal{G}_A(1^\lambda) = 1] = \pi_i + \negl(\lambda)$.

3 | WATERMARKING CRYPTOGRAPHIC FUNCTIONALITIES

In this section, we define the notion of watermarking a cryptographic functionality $C_F$. We start by providing the syntax of a watermarking scheme, in Section 3.1, and we proceed by defining the properties of a watermarking scheme. Specifically, since each property will be defined by employing a security game between a challenger and an adversary, as the first step, in Section 3.2, we describe the oracles which will be given access to an adversary. Then, we present the correctness and security properties of a watermarking scheme in Sections 3.3 and 3.4 respectively.

3.1 | Syntax of a watermarking scheme

Let $C_F$ be a cryptographic functionality defined by $m$ algorithms/circuits and $n$ properties $(\mathcal{G}^{\text{prop}}_A, \pi_i)_{i=1}^n$ and let $(E_i)_{i \in \mathbb{N}}$ be the space of the messages to be embedded in a circuit. The entities that are involved in a watermarking scheme are a set of clients, and an entity called the ‘Marking Service’.

Definition 5 (Stateful watermarking scheme). A stateful watermarking scheme for a cryptographic functionality $C_F$, consists of three probabilistic polynomial time algorithms $(\text{WGen}, \text{Mark}, \text{Detect})$ whose input/output behavior has as follows:

- $\text{WGen}$: On input $1^\lambda$, it outputs public parameters $\text{param}$ and a pair of keys $(mk, dk)$, where $mk$ is the marking key and $dk$ is the detection key. It also initializes a public variable state which can be accessed by all the parties.
- $\text{Mark}$: On input $mk, \text{param}$, a message $msg \in E_i$ (which is sent by a client to the Marking Service) and state, it outputs a tuple of circuits $(C_1, C_2, \ldots, C_m)$, an efficiently sampleable and representable distribution $D$ on the inputs of the circuit $C_i$.
- $\text{Detect}$: On input $dk, \text{param}$, state and a circuit $C_i$, it outputs a message $msg'$ or unmarked.

Notice that although the Marking Service outputs a tuple of circuits (as many as the algorithms of $C_F$), however, only one circuit among them is considered marked. By convention, we consider this circuit to be the first circuit of a tuple returned by Mark. This definition could be easily extended to capture cases where more than one circuits are considered marked. Second, a stateless watermarking scheme can also be described in the same way, by either assuming that the variable state remains constantly the empty string or by modifying Definition 5 so that state does not exist. Third, notice that a new feature of our definition of watermarking is that the Mark algorithm outputs a distribution $D$ on the inputs of a marked circuit. As it will be seen in Section 3.4, this distribution is relevant to our definitions of closeness and farness between circuits (cf. Definitions
8, 9) as it essentially defines on which inputs we expect similar or dissimilar circuits.

3.2 | Oracles

For our properties, we define the Challenge, Detect and Corrupt oracles in Figure 1. The Challenge oracle, on input \( \text{msg} \) calls the Mark algorithm and returns the tuple circuits output by Mark without including the first circuit of the tuple, that is the marked circuit. The tuple of circuits is returned to the adversary along with an index \( i \) that shows how many times Mark is invoked so far. The Corrupt oracle, given specific index \( i \) as input, outputs the whole tuple of circuits generated by Mark algorithm for the index \( i \) (including the first circuit) and works for values of \( i \) which were previously returned to the Challenge oracle. Finally, the Detect oracle, given as input a circuit, it simply runs Detect algorithm on that input. Note that all oracles have access to the public variable state, however state can only be modified by the challenger every time Mark is invoked.

**Remark 2** Notice that for marked (but not corrupted tuples) the adversary does not have access to the marked circuit \( C_1 \). This might be restrictive for certain schemes and properties. Consider for instance the case of CCA security for a public-key encryption scheme. Then, the marked algorithm would be the decryption one. Although the adversary should not receive \( \text{Dec}_{sk} \), he should still be able to query it on ciphertexts of his choice. Thus, we could define one more oracle name QueryOracle that would take as input an index \( i \) and an input \( x \) and would return the output of the \( i \)-th watermarked circuit produced by ChallengeOracle.

3.3 | Correctness properties

The correctness properties of a watermarking scheme are detection correctness and functionality property preserving.

At a high level, detection correctness states that if a circuit is marked with a message \( \text{msg} \) by running Mark algorithm, then if Detect is invoked on input that circuit, it will return \( \text{msg} \) with overwhelming probability. Our definition guarantees that any update of the state, after each execution of Mark, does not affect the output of Detect for previously marked circuits.

**Definition 6** (Detection correctness). We say that a watermarking scheme satisfies the detection correctness if for any PPT adversary \( \mathcal{A} \), it holds that

\[
\Pr[\text{det-corr}(1^\lambda) = 1] = \negl(\lambda),
\]

where the corresponding game is defined in Figure 2.

The functionality property-preserving notion captures the natural requirement that a watermarking scheme for a cryptographic functionality should primarily preserve the fundamental properties of the functionality. For example, a watermarking scheme for the PKE functionality should first and foremost be correct and IND-CPA secure. We formalize this notion by defining for each property \( \mathcal{F} \) of a functionality \( \mathcal{C}_x \) the corresponding security game \( \mathcal{G}_{\mathcal{A}}^{\text{wm-prop}}(1^\lambda) \), which is described in Figure 3. In the game of Figure 3, the adversary is allowed to issue a number of queries to Challenge, Corrupt and Detect oracles and then he chooses for which tuple generated by the Challenge oracle he will run the game \( \mathcal{G}_{\mathcal{A}}^{\text{prop}} \) with the challenger \( \mathcal{C} \).

**Definition 7** (Functionality property-preserving). Let \( \mathcal{C}_x \) be a cryptographic functionality defined by \( m \) algorithms and \( n \) properties \( \{ (\mathcal{G}_{\mathcal{A}}^{\text{prop}}(\pi_j)^{n}_j) \}_{j=1}^{n} \). We say that a watermarking scheme is functionality property preserving with respect to \( \mathcal{C}_x \) with error \( \varepsilon \) if for any property \( (\mathcal{G}_{\mathcal{A}}^{\text{prop}}(\pi_j), \pi_j) \), where \( j \in \{1, \ldots, n\} \), and it holds that for any PPT adversary \( \mathcal{A} \),

\[
\Pr[\mathcal{G}_{\mathcal{A}}^{\text{wm-prop}}(1^\lambda) = 1] \leq \pi_j + \varepsilon.
\]

**Remark 3** Regarding Definition 7, one could consider an alternative, stronger definition where the adversary \( \mathcal{A} \) is also supplied with the marking key \( mk \). The adversary could mark objects by himself and then run the property game using the circuits returned by Mark for a value of state of his choice. Second, we could define functionality

\[
\mathcal{G}_{\mathcal{A}}^{\text{det-corr}}(1^\lambda):
\]

1. The Challenger \( \mathcal{C} \) runs \( \text{WGen}(1^\lambda) \) and outputs \( (\text{param}, (mk, dk), \text{state}) \). It gives \( \text{param} \) to the adversary \( \mathcal{A} \) and if detection is public, \( \mathcal{C} \) also gives \( dk \) to \( \mathcal{A} \). \( \mathcal{A} \) has also access to the public variable state. Then, \( \mathcal{C} \) sets \( \text{Mark} \leftarrow \emptyset \), \( \text{Corrupt} \leftarrow \emptyset \) and \( i \leftarrow 0 \).
2. \( \mathcal{A} \) issues queries to \( \text{DetectOracle}, \text{ChallengeOracle} \) and \( \text{CorruptOracle} \).
3. \( \mathcal{A} \) outputs an index \( j \).
4. \( \mathcal{A} \) wins the game if \( (j, (C_1^j \ldots), D_j, \text{msg}) \notin \text{Marked} \) and \( \text{Detect}(dk, \text{param}, C_1^j, \text{msg}) \neq \text{msg} \).
\[ G_A^{\text{wmp\prop}}(1^\lambda) : \]

1. The Challenger runs \( \text{WGen}(1^\lambda) \) which outputs \((\text{param}, (mk, dk), \text{state})\). It gives \text{param} to the adversary \( \mathcal{A} \). \( \mathcal{A} \) also has access to the public variable state. If detection is public, \( \mathcal{A} \) also receives \( dk \) from the Challenger. The Challenger initializes the sets \( \text{Marked} \) and \( \text{Corrupt} \) as empty and \( i \leftarrow 0 \).
2. \( \mathcal{A} \) can make queries to \( \text{DetectOracle} \), the \( \text{ChallengeOracle} \) and the \( \text{CorruptOracle} \).
3. \( \mathcal{A} \) sends an index \( i \) to denote the \( i \)-th tuple generated by the \( \text{ChallengeOracle} \).
4. Then, \( \mathcal{A} \) and \( C \) interact exactly as in the game \( G_A^{\text{prop}} \), with the following differences:
   (a) \( C \) does not send to \( \mathcal{A} \) a subset of the keys generated by the key-generation algorithm as in \( G_A^{\text{prop}} \), but sends the corresponding circuits returned by the \( i \)-th query to the \( \text{ChallengeOracle} \) instead.
   (b) Depending on the property, if \( \mathcal{A} \) is not allowed to have knowledge on the marked circuit \( C^i \), but \( \mathcal{A} \) had previously obtained it through a query to the \( \text{CorruptOracle} \), then the game stops.

**FIGURE 3** The Functionality the property-preserving game for a property \( \text{prop} \),

property-preserving with respect to specific parameters as upper bounds on the number of queries that can be issued to the Challenge, Corrupt and Detect oracles. However, we omit referring to such parameters in Definition 7 to simplify the presentation. The same holds for the definitions of detection correctness, unremovability and unforgeability (i.e. Definitions 6, 10, 11).

**Remark 4** The notion of functionality-preserving was defined in the literature but with a different meaning than in our setting. In [11], it was used to capture that for any input \( x \) the outputs of the unmarked and the corresponding marked circuits should remain the same (i.e. \( C(x) = C(x) \)). The notion was also used in a similar way in [17].

### 3.4 Security properties

The security properties of a watermarking scheme, called *unremovability* and *unforgeability*, can be intuitively described as follows. Unremovability requires that one should not be able to create a circuit which is ‘close’ to a marked circuit but it is unmarked. On the other hand, unforgeability puts forth that it should be difficult to come up with a circuit which is ‘far’ from a marked circuit but it remains marked. In other words, unforgeability essentially requires that an adversary should not be able to produce marked circuits on its own without having access to the marking key. These properties will be captured by the unremovability and unforgeability security games. Informally speaking, in both games, at a first step, an adversary \( \mathcal{A} \) will be allowed to obtain a number of marked circuits by making queries to the \( \text{CorruptOracle} \). Then, in the unremovability game, \( \mathcal{A} \) tries to remove a mark while at the same time is kept close to a marked circuit \( C \), whereas in the unforgeability game \( \mathcal{A} \) attempts to produce a circuit \( C^* \) that remains marked while it ‘diverges’ from a marked circuit \( C \). It becomes clear that a crucial step towards defining formally unremovability and unforgeability is to define proximity relations between circuits. Below, we define ‘closeness’ and ‘farness’ relations which are coupled with unremovability and unforgeability notions accordingly.

**Relations between circuits**: For our definitions, we consider probabilistic circuits where the coins of some circuit \( C \) are chosen uniformly at random from a space \( R \). Deterministic circuits can be captured by setting \( R \) to be a singleton.

As a convention, in the case of deterministic circuits we set \( R = \{ \varepsilon \} \), where \( \varepsilon \) is the empty string. For a circuit \( C \), with \( C(x; r) \) we denote the output of \( C \) on input \( x \) with randomness \( r \in R \), while by \( \text{Out}(C(x)) \) we denote the output space of \( C \) for the input \( x \), that is \( \text{Out}(C(x)) = \{ C(x; r) \}_{r \in R} \). Thus, an element \( y \) belongs to \( \text{Out}(C(x)) \), that is \( y \in \text{Out}(C(x)) \), if there is \( r' \in R \) such that \( y = C(x; r') \).

**Definition 8** (\( (\rho, \delta) \)-closeness). Let \( C_1, C_2 \) be the two probabilistic circuits with input space \( \mathcal{X} \) and randomness spaces \( R_1 \) and \( R_2 \). We will say that a circuit \( C_1 \) is \( (\rho, \delta) \)-*close* to a circuit \( C_2 \) with respect to a distribution \( D \), and we will denote it as \( C_1 \sim_{(\rho, \delta), D} C_2 \), if

\[
\Pr_{x \sim D} \left[ \Pr_{r \sim R_1} \left[ C_1(x; r) \in \text{Out}(C_2(x)) \right] \geq \delta \right] \geq \rho.
\]

**Definition 9** (\( (\gamma, \tau) \)-farness). Let \( C_1, C_2 \) be the two probabilistic circuits with input space \( \mathcal{X} \) and randomness spaces \( R_1 \) and \( R_2 \). We will say that a circuit \( C_1 \) is \( (\gamma, \tau) \)-*far* from a circuit \( C_2 \) with respect to a distribution \( D \), and we will denote it as \( C_1 \sim_{(\gamma, \tau), D} C_2 \), if

\[
\Pr_{x \sim D} \left[ \Pr_{r \sim R_1} \left[ C_1(x; r) \in \text{Out}(C_2(x)) \right] \leq \gamma \right] \geq \tau.
\]

Observe that Definitions 8 and 9 examine how a circuit approximates or diverges from another circuit that can be seen as a reference point (e.g. a marked circuit \( C \)), which is consistent with the way unremovability and unforgeability will be defined. We demonstrate the plausibility of Definitions 8, 9 by providing an example.

**Example**: Consider a digital signature scheme (SigGen, Sign, Verify) where the signing algorithm is probabilistic with the randomness space \( R \). Let \( \text{Sign}_{sk} \) be the signing algorithm, and let \( C^* \) be an adversarially created circuit, where \( C^* \) works as follows: For a large fraction of inputs \( x \) in the message space, \( C^* \) returns the same output as \( \text{Sign}_{sk}(x) \) only with small probability, but still \( C^* \) always returns an output in the image...
of \( \text{Sign}_k(x) \). Intuitively, one would expect that for a carefully chosen parameter, \( C^* \) would not be ‘close’ to \( \text{Sign}_k(x) \). On the other hand, for a large fraction of messages \( x \) in the message space, since the signatures returned by \( C^* \) belong to the image of \( \text{Sign}_k(x) \), they are valid signatures for \( x \). Based on that, in our view, \( C^* \) should be considered ‘close’ to \( \text{Sign}_k(x) \). Definition 8 captures that scenario, as for certain parameters \( \rho, \delta, \tau \), it may require that there is a non-negligible fraction of messages \( x \). For each of those messages, denoted as \( x \), there is a non-negligible fraction of randomness values \( r \). The output of the circuit on input of the message and the randomness value belongs in the image of \( \text{Sign}_k(x) \).

Proposition 1 Let \( C_1, C_2 \) be two circuits with input space \( X \) and randomness spaces \( R_1 \) and \( R_2 \) respectively. Let \( \gamma, \delta, \rho, \tau \in (0, 1) \). If \( \gamma > 1 - \rho \) and \( \delta > \tau \), \( C_1 \) cannot be \((\rho, \delta)\)-close to \( C_2 \) and \((\gamma, \tau)\)-far from \( C_2 \) at the same time according to Definitions 8 and 9.

Proof Assume that \( C_1 \) is \((\gamma, \tau)\)-far from \( C_2 \) and \( \gamma > 1 - \rho \). Then, \( \Pr_{x \in X, r \in R_1} \{ C_1(x; r) \in \text{Out}(C_2(x)) \} \leq \gamma > 1 - \rho \). Thus, \( \Pr_{x \in X, r \in R_1} \{ C_1(x; r) \in \text{Out}(C_2(x)) \} > \gamma \) and for any \( \delta > \tau \), \( \Pr_{x \in X, r \in R_1} \{ C_1(x; r) \in \text{Out}(C_2(x)) \} \geq \delta < \rho \).

Definitions of unremovability and unforgeability: Given Definitions 8, 9, we define \((\rho, \delta)\)-unremovability and \((\gamma, \tau)\)-unforgeability with the aid of the games of Figures 4 and 5. The definitions below refer to the more general model where messages beyond ‘marked’ and ‘unmarked’ are embedded in circuits.

Definition 10 \((\rho, \delta)\)-unremovability. We say that a watermarking scheme satisfies the \((\rho, \delta)\)-unremovability if for any PPT adversary \( \mathcal{A} \), it holds that \( \Pr_{\rho, \delta, \mathcal{A}}[w_{\text{unremv}}^\mathcal{A}(1^\lambda) = 1] = \text{negl}(\lambda) \).

Definition 11 \((\gamma, \tau)\)-unforgeability. We say that a watermarking scheme satisfies \((\gamma, \tau)\)-unforgeability if for any PPT adversary \( \mathcal{A} \), it holds that \( \Pr_{\gamma, \tau, \mathcal{A}}[w_{\text{unforge}}^\mathcal{A}(1^\lambda) = 1] = \text{negl}(\lambda) \).

Remark 5 In [31] the authors proposed a new definitional framework by giving separate definitions for the notions of ‘Watermarkable Digital Signatures’ and ‘Watermarkable Predicate encryption’ which is a generalized definition that includes a public-key encryption. Our approach is to provide a combined/unified definitional framework that simultaneously captures both public-key encryption and signatures. In addition, in [31] the unremovability definition applies to so called ‘useful’

circuits. A ‘useful’ signing circuit w.r.t. a specific verification key is considered the one which outputs valid signature for a non-negligible fraction of messages, while a ‘useful’ decryption circuit w.r.t. a specific public key is the one that decrypts correctly a non-negligible fraction of ciphertexts, encrypted under this public key. Then, unremovability requires that an adversary should output a ‘useful’ circuit which is at the same time unmarked or marked with a different original message, rather than indicating that the adversarial circuit is required to agree with a decryption/signing circuit on a fraction of inputs (cf. [15]).

Notice that our model captures the notion of a ‘useful’ circuit in a different way by defining closeness and farness relations w.r.t. a distribution \( \mathcal{D} \) which defines the inputs that we aim to compare two circuits. As it will become clear in our construction Section 5.1, for the case of PKE our notions coincide as the distribution \( \mathcal{D} \) refers to valid ciphertexts under a specific public key.

Remark 6 In the literature (i.e. [9, 11, 15, 18]), meaningfulness was considered to be a core security property of a watermarking scheme, and it requires that the
vast majority of circuits be unmarked. As observed in [17] though, meaningfulness is implied by unforgeability: if an adversary can sample a circuit that is marked with good probability, then it can directly forge a circuit without making any oracle queries.

4 | WATERMARKING CRYPTOGRAPHIC IMPLEMENTATIONS

A watermarking scheme for a cryptographic functionality as defined in Section 3, refers essentially to a scheme that satisfies correctness and security properties defined in Sections 3.3 and 3.4. Such a watermarking scheme can potentially be created from scratch without necessarily having as a starting point a specific implementation. Thus, the notion of watermarking a cryptographic implementation aims to define what it would mean to watermark a specific implementation (e.g. ElGamal encryption scheme).

Consider a scheme (Gen, C1, ..., Cm−1) that implements a functionality C_F defined by m algorithms and n properties (g_{prop}^{A_i}, \pi_i)_{i=1}^q. First, we note that a watermarking scheme (WGen, Mark, Detect) for (Gen, C1, ..., Cm−1) maintains the syntax of Section 3.1. Second, the definitions of the properties of detection-correctness, (\rho, \delta)-unremovability and (\gamma, \tau)-unforgeability remain the same. What differentiates the notion of ‘watermarking a cryptographic implementation’ from the notion of ‘watermarking a cryptographic functionality’ is only the implementation property-preserving notion. Implementation property-preserving does not only require that the watermarking scheme preserves the properties of the functionality, but also it requires that it preserves the properties of the initial scheme except with some small error. Intuitively, by this notion we aim to capture that if there is a PPT adversary which breaks a property of a watermarked implementation, then, there will be a PPT adversary which breaks the corresponding property for the initial non-watermarked implementation. For example, if we consider a watermarked ElGamal encryption scheme, if an adversary manages to break IND-CPA security of the watermarked ElGamal encryption scheme, then it will break the IND-CPA security of the original ElGamal encryption scheme.

Definition 12 captures formally the implementation property-preserving notion. Since the syntax of a watermarking scheme for cryptographic implementations remains the same with the case of the functionality, the game G_{S^\text{prop}^m}^{\text{prop}^m}((1^t)) refers to the security game described in Figure 3. The game G_{A}^{\text{prop}^m}((1^t)) refers to the j-th property of the functionality.

Definition 12 (Implementation property-preserving). A watermarking scheme is implementation property-preserving with error \epsilon for a cryptographic implementation (Gen, C1, ..., Cm−1) if for any property (g_{prop}^{A_i}, \pi_i) it holds that: For any PPT adversary S there is a PPT adversary A such that |Pr[G_{S^\text{prop}^m}^{\text{prop}^m}((1^t)) = 1] - Pr[G_{A}^{\text{prop}^m}((1^t)) = 1]| \leq \epsilon.

5 | WATERMARKING PUBLIC KEY ENCRYPTION AND DIGITAL SIGNATURE IMPLEMENTATIONS

We now present two watermarking constructions, one for public-key encryption and one for digital signature schemes. Our constructions share the same idea and can be viewed as compilers which take as input an existing public-key encryption scheme (or digital signature scheme) and convert it into a watermarked public-key encryption scheme (or digital signature scheme). Our constructions support private key detection and embed to the encryption/signing circuits one-bit messages, in the sense that the circuits are either marked or unmarked. Before proceeding to a formal description of our constructions, we first provide some intuition by referring to a simpler construction, which supports public-key detection and has a state of size linear to the number of keys generated by the marking service.

Public-key detection via linear size state versus secret-key detection via logarithmic size state: Consider a PKE scheme (Gen, Enc, Dec). Given that the shared state can be represented as a public table, a watermarked PKE scheme could work as follows: For any marking request, Mark generates a fresh pair of keys (pk, sk) using Gen algorithm, that is the key generation algorithm of the public-key encryption scheme that is being watermarked. Then, it stores the generated public key pk to the state table and returns (Enc_{pk}, Dec_{sk}) to the client. Thus, state will hold all the public keys generated by Mark at a certain point in time. Now, how does Detect work given access to the state table? When Detect receives as input a (decryption) circuit C, it will check for any public key stored in the public table state, whether the circuit can correctly decrypt a number of ciphertexts which is above a certain threshold. How to determine the maximum number of ciphertexts to check this threshold is not trivial and depends on the parameters of the properties of the scheme.

Such a construction could be proven to be a secure watermarking scheme for public-key encryption however the use of a state that grows linearly to the number of markings is not very appealing in practice especially for implementations where the public keys are large. Furthermore, such a construction could be extended to a message-embedding construction, where the message selected by the user will be stored in the public table together with the public key. Such a construction will also depend linearly to the size of the message. A similar construction could be used for digital signatures, where the generated verification keys will be stored on the state table. In that case, roughly speaking, when Detect is given as input a (signing) circuit C, it will check whether there is a verification key in the state table that can verify a certain number of signatures generated by C.

We overcome the problem of linear-size state by focusing on watermarking schemes with private key detection. In Sections 5.1 and 5.2, we suggest watermarking schemes where the state is of a logarithmic size (in the number of marked circuits)
and private key detection is supported, with the same key being used by Mark and Detect algorithms.

**Overview of our constructions:** In our constructions, WGen algorithm chooses uniformly at random a key $K$ for a pseudorandom function $F$. The key $K$ is used both as a marking and detection key and is kept private by the Marking Service. This algorithm also initializes a public variable state as empty. For the case of a PKE scheme (Gen, Enc, Dec), whenever Mark is run, it sets state = state + 1 and computes public-secret key pair $(pk, sk)$ by running Gen$(1^n)$ with randomness $F(K, state + 1)$. Then, it returns $(Enc_{pk}, Dec_{sk})$ to the client. Observe that the state is essentially a counter of the marked circuits issued by Mark. Detect algorithm identifies whether a decryption circuit is marked or not, by re-generating the public-secret key pairs using the PRF $F(K, i)$, for $i \leq$ state. For each produced $pk_i$, it chooses uniformly at random a number of plaintexts, encrypts them under $pk_i$ and checks whether $C$ decrypts correctly a certain fraction of them. If this holds for at least one public key, Detect returns *marked*, otherwise it returns *unmarked*. Along the same lines, for a digital signature scheme $(SigGen, Sign, Verify)$, Mark will compute a verification-signing key pair $(vk, sk)$ by running SigGen$(1^n)$ with randomness $F(K, state + 1)$ and Detect will perform in a similar way.

Although the watermarking constructions for PKE schemes and digital signature schemes are very similar, the security analysis for those constructions has some significant differences. First, our security analysis for watermarking PKE schemes as described above holds for any PKE scheme that is correct and IND-CPA secure. On the contrary, the original properties of the digital signature schemes do not suffice for proving the security properties of our watermarking scheme. In particular, the underlying digital signature scheme should satisfy a property called verification soundness which is described in Section 2. This property is satisfied by standard signature schemes such as Schnorr’s scheme and RSA signatures. In the Appendix, we prove that Schnorr’s signature scheme satisfies verification soundness.

The constructions and the security analysis are presented in full detail in Sections 5.1 and 5.2.

**A note about state:** Note that the state information in our construction is public and it should be immutable for the system to work in practice. A potential solution for storing the state would be by using a public bulletin board or a blockchain system. For example, every time the state is updated, the marking service signs it and posts a new transaction in the blockchain. This system works in a similar way.

**5.1 A watermarking scheme for implementations of PKE**

In Figure 6, we present a watermarking scheme (WGen, Mark, Detect) for a PKE scheme (Gen, Enc, Dec). We prove that the scheme of Figure 6 satisfies the properties of a watermarking scheme as defined in Sections 3.3 and 3.4 by proving Theorem 1. We note that in our analysis, we consider the key-generation algorithms which create their random tape by choosing keys uniformly at random. This aligns with the key generation algorithms of all the well-known encryption schemes.

**Theorem 1** Let $(Gen, Enc, Dec)$ be an implementation of the public-key encryption functionality that has plaintext space $M$ of exponential size (in the security parameter) and satisfies perfect correctness and IND-CPA security. Let $F : \mathcal{K} \times \{0,1\}^n \rightarrow \{0,1\}^\lambda$ be a PRF, where $\mathcal{K}$ is the key space. Then, the scheme in Figure 6 is a watermarking scheme for the implementation $(Gen, Enc, Dec)$. Namely, it satisfies detection correctness, implementation property-preserving, $(\rho, \delta)$-unremovable and $(\gamma, \tau)$-unforgeable, where $\varepsilon_{pf}$ is the security of the PRF, $\rho = \delta = 1/poly(\lambda)$, $\gamma \geq 1 - \frac{\rho^3}{\tau^3}$ and $\tau < \frac{\rho^4}{\gamma^3}$.

We prove Theorem 1 by proving separately each property of a watermarking scheme in Lemmas 1, 2, 3 and 5. We note that although we assume that $(Gen, Enc, Dec)$ is perfectly correct, our proofs can also be extended for implementations which have a negligible decryption error.

**Lemma 1 (Implementation property-preserving).** If the public-key encryption scheme $(Gen, Enc, Dec)$ satisfies IND-CPA security, perfect correctness, and

- $WGen$: On input $1^\lambda$, it chooses uniformly at random a key $K$ for a pseudorandom function $F : \mathcal{K} \times \{0,1\}^n \rightarrow \{0,1\}^\lambda$. It outputs $mk = dk = K$, it initializes the public variable state ← 0 and sets $i ← 0$. It chooses parameters $\rho, \delta$ such that $\rho = \delta = 1/poly(\lambda)$.
- $Mark$: On input $K$, state, marked, it sets $i ← state + 1$ and runs $Gen(1^\lambda)$ with randomness $F(K, i)$. The output is a public-secret key pair $(pk_i, sk_i)$ and the algorithm returns a pair of circuits $(Dec_{pk_i}, Enc_{sk_i})$. It sets as $D_i$ the distribution of the ciphertexts that correspond to plaintexts chosen uniformly from the plaintext space. Then, it updates the value of state by setting state ← state + 1.
- $Detect$: On input $K$, a circuit $C$ and state, it runs as follows.
  For $i = 1$ to state:
  - Run $Gen(1^\lambda)$ with randomness $F(K, i)$ (as the Mark algorithm does) in order to obtain $(pk_i, sk_i)$.
  - Choose $k = \frac{\lambda}{\rho^2}$ plaintexts uniformly at random from the plaintext space $M$, i.e., $m_1, \ldots, m_k$. Then encrypt them under the public key $pk_i$ by computing the ciphertexts $(c_j = Enc(pk_i, m_j))_{i=1}^k$.
  - For $j = 1$ to $k$ check whether $C(c_j) = m_j$. If $C$ decrypts correctly at least $\frac{\lambda}{\rho^2}$ ciphertexts out of $k = \frac{\lambda}{\rho^2}$ ciphertexts $c_1, \ldots, c_k$, return marked.
  Otherwise, return unmarked.
$F: \mathcal{K} \times \{0,1\}^n \rightarrow \{0,1\}^\ell$ is a PRF, then the watermarking scheme of Figure 6 is the implementation property-preserving with a negligible error according to Definition 12.

**Proof:** We prove that the watermarking scheme of Figure 6, preserves the IND-CPA security and correctness, in Claims 1.1 and 1.2, respectively.

**Claim 1.1:** [IND-CPA security] Assuming that $F$ is a PRF then, for any PPT adversary $\mathcal{A}$, there is a PPT adversary $\mathcal{S}$ such that

$$\left| \Pr [\mathcal{G}_0] - \Pr [\mathcal{G}_1] \right| \leq \text{negl}(\lambda).$$

**Proof of Claim 1.1:** This claim is proven by using a sequence of games. We define $\mathcal{G}_0 = \mathcal{G}_{\text{wm}}$ (for a fixed $\mathcal{S}$). Then we define $\mathcal{G}_1$ by substituting any call to the PRF function $F$ for a key $K$ to generate tuples $(\text{Dec}_{pk}, \text{Enc}_{pk})$ by a call to a random function $f: \{0,1\}^n \rightarrow \{0,1\}^\ell$. Owing to the security of the PRF function $F$, we stress that $\Pr [\mathcal{G}_0] - \Pr [\mathcal{G}_1] \leq \text{negl}(\lambda)$. If this did not hold, then a PRF distinguisher by making oracle queries and run Gen with randomness the output of each oracle query, could distinguish between a PRF and a random function with non-negligible probability.

We now denote as $\mathcal{G}_2 = \mathcal{G}_{\text{wm}}^{\text{IND-CPA}}(\lambda)$ and we will show that $\Pr [\mathcal{G}_1] - \Pr [\mathcal{G}_2] \leq \text{negl}(\lambda)$. By assumption, we have the $(\text{Gen}, \text{Enc}, \text{Dec})$ as IND-CPA secure and thus, for any PPT adversary $\mathcal{A}$, $\Pr [\mathcal{G}_2] \leq 1/2 + \text{negl}(\lambda)$. We will show that $\Pr [\mathcal{G}_1] \leq 1/2 + \text{negl}(\lambda)$, as well. Assume that there is a PPT adversary $\mathcal{A}$ breaking $\mathcal{G}_1$ with non-negligible advantage $\alpha$, then we can construct $\mathcal{A}'$ breaking $\mathcal{G}_2$ with the probability of at least $\frac{\alpha}{n}$, where $n$ is the number of public keys obtained through oracle queries. $\mathcal{A}'$, upon receiving a public key $pk$ from the IND-CPA challenger of $\mathcal{G}_2$, simulates $\mathcal{G}_1$ by responding honestly to the queries of $\mathcal{A}'$ and plugging $pk$ into one of the $n$ queries by guessing randomly the ’attacked query position’ of $\mathcal{A}$.

Therefore, we have that $\Pr [\mathcal{G}_1] = 1 - \Pr [\mathcal{G}_2] \leq \text{negl}(\lambda)$, and thus $\Pr [\mathcal{G}_1] \leq 1/2 + \text{negl}(\lambda)$, which completes our proof. (End of Claim 1.1)

**Claim 1.2:** [Correctness] For any PPT adversary $\mathcal{S}$, there is a PPT adversary $\mathcal{A}$ such that $\Pr [\mathcal{G}_0] - \Pr [\mathcal{G}_1] = 0$, under the assumption that the encryption scheme $(\text{Gen}, \text{Enc}, \text{Dec})$ is perfectly correct.

**Proof of Claim 1.2:** By assumption, we have that $\Pr [\mathcal{G}_0] = 1$, and if we replace Gen with an algorithm $\text{Gen}'$ with output space which is a subset of the output space of Gen, then the correctness is preserved with no error. This holds for $\text{Gen}'$ of the watermarking scheme. Hence, $\Pr [\mathcal{G}_1] = 1$. (End of Claim 1.2)

**Lemma 2** (Detection correctness). The watermarking scheme of Figure 6 satisfies detection correctness under the assumption that the underlying encryption scheme $(\text{Gen}, \text{Enc}, \text{Dec})$ satisfies the perfect correctness.

**Proof** Notice that Detect algorithm of the scheme of Figure 6 reconstructs exactly the same circuits one-by-one already produced by the Mark algorithm. By Lemma 1, our watermarking scheme preserves perfect correctness of the scheme $(\text{Gen}, \text{Enc}, \text{Dec})$ with no error. Therefore, there is at least one marked circuit $C_j$, for $j \leq \lambda$, which decrypts correctly all the ciphertexts generated by Detect using $pk$. This means that Detect returns marked.

**Lemma 3** $(\rho, \delta)$-unremovability. The scheme of Figure 6 satisfies $(\rho, \delta)$-unremovability, where $\rho = \delta = 1/\text{poly}(\lambda)$, under the assumption that the underlying public-key encryption scheme $(\text{Gen}, \text{Enc}, \text{Dec})$ satisfies perfect correctness.

**Proof** We prove this lemma using counting arguments independent of the adversary’s strategy. Specifically, first, we fix $(i, (\text{Dec}_{pk}, \text{Enc}_{pk}), D_i) \in \text{Marked}$ and we assume that $C^\ast \sim (\rho, \delta, D_i) \text{ Dec}_{sk}$. Then, we prove that if this holds, Detect returns unmarked only with negligible probability. Let $R$ be the randomness space of the adversarial circuit $C^\ast$. We define the random variable $X_{ij}$ as follows:

$$X_{ij}(c, r) = \begin{cases} 1, & \text{if } C^\ast (c, r) = \text{Dec}_{sk}(c) \\ 0, & \text{otherwise} \end{cases}$$

where $(c, r) \in [D_i] \times R$. It holds that

$$\Pr [X_{ij} = 1] = \Pr _{\ell \sim [\ell], r \sim R} [C^\ast (c, r) = \text{Dec}_{sk}(c)].$$

By the assumption regarding $C^\ast$ and Definition 8, we have

$$\Pr _{\ell \sim [\ell], r \sim R} [C^\ast (c, r) = \text{Dec}_{sk}(c)] \geq \delta \geq \rho.$$
Pr[X_{ij} = 1] \geq \Pr_{r \sim \mathcal{R}}[C(c; r) = \text{Dec}_{sk}(c) | c \in \text{Good}] \Pr[c \in \text{Good}]
\geq \rho \delta = \rho^2.
\quad (4)

Then, we define the random variable \( X_i = \sum_{j=1}^{\lambda/\rho^2} X_{ij} \), where the random variables \( X_{ij} \) for \( j = 1, \ldots, \lambda/\rho^2 \) are independent. Thus, we have
\[
\mu = E[X_i] = \sum_{i=1}^{\lambda/\rho^2} E[X_{ij}] \geq \sum_{i=1}^{\lambda/\rho^2} \rho^2 = \frac{\lambda}{\rho^2}.
\quad (5)
\]

By Lemma 1, we have that the watermarking scheme preserves perfect correctness of the encryption scheme (Gen, Enc, Dec) without introducing any decryption error. This means that the random variable \( X_i \) counts the number of ciphertexts from \( \mathcal{D}_i \) (out of \( k = \lambda/\rho^2 \)) which are correctly decrypted by \( C^* \). We will compute an upper bound on \( \Pr[X_i < \frac{\lambda}{\rho^2}] \) using the following Chernoff bound:
\[
\Pr[X_i \leq (1 - \epsilon)\mu] \leq e^{-\epsilon^2/2}, \text{ for all } 0 < \epsilon < 1.
\quad (6)
\]

First, by Equation (5), we have that
\[
\Pr[X < \frac{\lambda}{\rho^2}] \leq \Pr[X_i \leq \frac{\lambda}{\rho^2}] \leq \Pr[X_i \leq \mu/2].
\]
If we set \( \epsilon = 1/2 \) in Equation (6), we have that
\[
\Pr[X_i \leq \mu/2] \leq e^{-\mu/8} \leq e^{-\mu/2} = e^{-\text{neg} (\lambda)},
\quad (7)
\]
which is negligible in \( \lambda \). Thus, \( \Pr[X_i \geq \frac{\lambda}{\rho^2}] \geq 1 - \text{neg} (\lambda) \), meaning that \( C^* \) decrypts correctly at least \( \frac{\lambda}{\rho^2} \) ciphertexts except with negligible probability. Therefore, Detect returns marked except with negligible probability.

To simplify the presentation of our proof for \((\gamma, \tau)\)-unforgeability, we first prove an intermediate lemma, that is Lemma 4.

**Lemma 4** Let \((\text{Gen}, \text{Enc}, \text{Dec})\) be a public-key encryption scheme that satisfies perfect correctness and IND-CPA security. Let \( \text{Dec}_{sk} \) be a decryption circuit where, \((pk, sk) \leftarrow \text{Gen}(1^\lambda)\) and \( \mathcal{D}_i \) are the distribution of ciphertexts that correspond to plaintexts chosen uniformly at random from the plaintext space \( M \) and encrypted under \( pk \). Assume a circuit \( C^* \) such that \( C^*_{\gamma, \tau, \mathcal{D}_i} \text{Dec}_{sk} \), where \( \gamma \geq 1 - \frac{\delta}{\rho} + \tau \) and \( \tau < \frac{\delta}{\rho} \), \( \rho = \delta = 1/\text{poly}(\lambda) \). Then, \( C^* \) decrypts correctly at least \( \frac{\lambda}{\rho^2} \) out of \( \frac{\lambda}{\rho^2} \) ciphertexts chosen from \( \mathcal{D}_i \) only with negligible probability.

**Proof** Let \( \mathcal{R} \) be the randomness space of \( C^* \). Similarly to the proof of Lemma 3, we define the random variable
\[
X_{ij}(c, r) = \begin{cases} 1, & \text{if } C^*(c; r) = \text{Dec}_{sk}(c) \\ 0, & \text{otherwise} \end{cases}
\quad (8)
\]
where \((c, r) \in [\mathcal{D}_i] \times \mathcal{R} \). It holds that
\[
\Pr[X_{ij} = 1] = \Pr_{c \sim \mathcal{D}_i, r \sim \mathcal{R}}[C^*(c; r) = \text{Dec}_{sk}(c)].
\quad (9)
\]

By the assumption regarding \( C^* \) and Definition 9, we have that
\[
\Pr_{c \sim \mathcal{D}_i, r \sim \mathcal{R}}[C^*(c; r) = \text{Dec}_{sk}(c)] \leq \tau \geq \gamma.
\quad (10)
\]

We consider the set
\[
\text{Good} = \{c \in [\mathcal{D}_i] \mid \Pr_{r \sim \mathcal{R}}[C^*(c; r) = \text{Dec}_{sk}(c)] \leq \tau \}.
\]

Given the above set, by Equation (9), we have
\[
\Pr[X_{ij} = 1] = \Pr_{c \sim \mathcal{D}_i, r \sim \mathcal{R}}[C^*(c; r) = \text{Dec}_{sk}(c)] \cdot \Pr[c \in \text{Good}] + \Pr_{c \sim \mathcal{D}_i, r \sim \mathcal{R}}[C^*(c; r) = \text{Dec}_{sk}(c)] \cdot \Pr[c \in \text{Good}]
\leq \tau + (1 - \gamma).
\quad (11)
\]

Then we define the random variable \( X_i = \sum_{j=1}^{\lambda/\rho^2} X_{ij} \), where the random variables \( X_{ij} \) are independent. Thus, we have
\[
\mu = E[X_i] = \sum_{j=1}^{\lambda/\rho^2} \Pr[X_{ij} = 1] \leq \frac{\lambda}{\rho^2} (1 - \gamma + \tau).
\quad (12)
\]

Since \( \gamma \geq 1 - \frac{\delta}{\rho} + \tau \) for \( \tau < \frac{\delta}{\rho} \), we have that \( \mu \leq \frac{\lambda}{\rho^2} \). We utilize the following Chernoff bound
\[
\Pr[X_i \geq (1 + \epsilon)\mu] \leq e^{\frac{\epsilon^2}{2}/3}, \text{ for all } 0 < \epsilon < 1.
\quad (13)
\]

If we set \( \epsilon = 1/2 \), we have that \( \Pr[X_i \geq \frac{\lambda}{\rho^2}] \leq e^{\frac{\lambda^2}{2\rho^2}}, \text{ which is negligible in } \lambda.\)
negligible probability $\beta$. The algorithm $B$ runs $A$ by playing the role of the Challenger in the $(y, r)$-unforgeability game and takes advantage of $A$'s output $C^*$ to break the IND-CPA security. By the security of the PRF $F$, for $K \leftarrow \mathcal{K}$, $B$ can simulate any query to the Challenge Oracle, by running $\text{Gen}(1^\ell)$ with randomness chosen uniformly at random $\{0, 1\}^\ell$ instead of calling $F(K, i)$ for an index $i$. The reduction is described in detail in Figure 7. Without the loss of generality, we assume that $A$ makes $m_1$ queries to the Challenge Oracle, $m_2$ queries to the Corrupt Oracle and $m_3$ queries to the Detect Oracle.

**Remark 7** Notice that in step 5 of the reduction of Figure 7 there is a probability for messages in the same positions of the two tuples to be equal, that is $M_0$.

$\Pr F$ breaks the IND-CPA security. By the security of the PRF, $F$, for $K \leftarrow \mathcal{K}$, $B$ can simulate any query to the Challenge Oracle, by running $\text{Gen}(1^\ell)$ with randomness chosen uniformly at random $\{0, 1\}^\ell$, instead of calling $F(K, i)$ for an index $i$. The reduction is described in detail in Figure 7. Without the loss of generality, we assume that $A$ makes $m_1$ queries to the Challenge Oracle, $m_2$ queries to the Corrupt Oracle and $m_3$ queries to the Detect Oracle.

**Description of $B$:**

1. $B$ receives $pk$ from the Challenger of the IND-CPA security game.
2. $B$ chooses $i^* \leftarrow \{1, \ldots, m_1\}$ and initializes state $\leftarrow 0$, $i \leftarrow 0$ and Marked $\leftarrow 0$. $B$ sets $pk_e = pk$.
3. $B$ provides $A$ with $\text{param}$ and simulates $A$'s oracle queries as follows:

**Challenge Oracle queries:** When $A$ makes a query to the Challenge Oracle, $B$ sets $i \leftarrow i + 1$ and state $\leftarrow + 1$.

1. If $i \neq i^*$, $B$ chooses $r_i \leftarrow \{0, 1\}^\ell$, runs $\text{Gen}(1^\lambda)$ with randomness $r_i$ and generates a pair $(pk_i, sk_i)$. Then, $B$ stores $(i, (\text{Dec}_{ik_i}, \text{Enc}_{pk_i}), D_i)$ to the table Marked and sends $(i, \text{Enc}_{pk_i}, D_i)$ to $A$. If $i = i^*$, then $B$ returns $(i^*, \text{Enc}_{pk_i}, D_i)$ to $A$.

**Corrupt Oracle queries:** When $A$ makes a query $i < m_1$ to the Corrupt Oracle, if $i = i^*$, since $B$ does not own the secret key that corresponds to the public key $pk_{i^*}$, $B$ outputs a random bit $b \in \{0, 1\}$ and the game stops. If $i \neq i^*$, $B$ sends $(i, (\text{Dec}_{ik_i}, \text{Enc}_{pk_i}), D_i)$ to $A$.

**Detect Oracle queries:** When $A$ makes a query $C$, $B$ runs the algorithm Detect by utilizing the encryption functions stored at the table Marked. Each message of the tuple is chosen by $B$ uniformly at random from the message space.

1. Upon receiving $(c_{0,1}, \ldots, c_{0,\lambda/\rho^3})$ from the Challenger, where $c_{0,j} = \text{Enc}(pk, M_0)$, $B$ initializes three counters $\text{count}_0, \text{count}_1, \text{count}_\perp$ to 0. Then $B$ runs Detect on input $C^*$ in a way that at round $i^*$, it uses the ciphertexts $(c_{0,1}, \ldots, c_{0,\lambda/\rho^3})$ given by the Challenger. Then, for $j = 1, \ldots, \lambda/\rho^3$, $B$ does the following:
   - If $C^*(c_{0,j}) = M_0$, it sets $\text{count}_0 \leftarrow 1$.
   - If $C^*(c_{0,j}) = M_1$, it sets $\text{count}_\perp \leftarrow 1$.
   - If $C^*(c_{0,j})$ is not equal to $M_0$ or $M_1$, $B$ outputs 0 or 1 respectively. Otherwise, it outputs a random bit $b$, based on the conditions of step 6.

**Figure 7** Unforgeability to IND-CPA reduction
the plaintexts in the two tuples are chosen uniformly at random from a plaintext space \( \mathcal{M} \) of exponential size.

To sum up, for the case where \( B \) does not correctly guess \( j^* \), \( B \) wins with probability \( 1/2 \). Thus, \( \Pr[\text{\small B wins} | \neg\text{\small Ecorr} \land \neg\text{\small Guess}] \Pr[\text{\small A wins}] = (1 - \operatorname{negl}(\lambda))\Pr[\neg\text{\small Guess}] \).

Based on the winning probabilities above, we have that: \( \Pr[B \text{ wins } \mid \text{ A wins}] \geq \frac{1}{2} + \frac{\lambda}{2m_0} - \operatorname{negl}(\lambda) \). We now move to the case where \( \mathcal{A} \) does not win \((\gamma, \tau)\)-unforgeability game. One possible case is that \( \mathcal{A} \) makes a CorruptOracle query for the index \( i^* \). Since \( \mathcal{B} \) cannot provide the corresponding secret key, it aborts by returning a random bit. Supposing that \( \mathcal{B} \) does not abort, two possible scenarios can take place considering the \((\gamma, \tau)\)-unforgeability game: (1) Either \( \text{Detect} \) returns unmarked or (2) there is a decryption function \( \text{Dec}_{\text{sk}_e} \in \text{Corrupted} \) such that \( C^* \) is not \((\gamma, \tau)\)-far from \( \text{Dec}_{\text{sk}_e} \). In the former case, \( B \) outputs a random bit. In the latter case, that is when case (2), splits in two subcases, (2a) there is \( \text{Dec}_{\text{sk}_e} \) such that \( C^* \) is not \((\gamma, \tau)\)-far from \( \text{Dec}_{\text{sk}_e} \) and \( \text{Detect} \) returns unmarked, and (2b) there is \( \text{Dec}_{\text{sk}_e} \) such that \( C^* \) is not \((\gamma, \tau)\)-far from \( \text{Dec}_{\text{sk}_e} \) and \( \text{Detect} \) returns marked. In case (2a), \( B \) will output a random bit. In case (2b), we should examine how \( C^* \) behaves on inputting the tuple of ciphertexts \((c_{k,1}, \ldots, c_{k,n})\) given by the Challenger. Using a similar analysis with the case where \( \mathcal{A} \) wins and \( \neg\text{\small Guess} \) is described in the previous paragraph, we stress that in case (2b), \( B \) wins with probability of at least \( 1/2 \). By the arguments above, we have \( \Pr[B \text{ wins } \mid \neg\text{\small A wins}] \geq 1/2 \). We conclude that \( \Pr[B \text{ wins }] \geq \frac{1}{2} + \frac{\lambda}{2m_0} - \operatorname{negl}(\lambda) \), where \( \alpha \) is the probability that \( \mathcal{A} \) breaks the \((\gamma, \tau)\)-unforgeability. Therefore, \( B \) breaks the IND-CPA security with the non-negligible probability.

### 5.2 A watermarking scheme for digital signatures schemes

In this section, in a similar spirit with the watermarking construction of Section 5.1, we present a watermarking construction \((\text{WGen, Mark, Detect})\) for a digital signature scheme \((\text{SigGen, Sign, Verify})\). The detailed construction can be seen in Figure 8. In Theorem 2, we prove that the construction of Figure 8 satisfies the properties of a watermarking scheme. As it will become clear below, our security analysis requires that the underlying digital signature scheme satisfies a property called verification soundness (see Definition 2).

**Theorem 2** Let \((\text{SigGen, Sign, Verify})\) be a digital signature scheme which satisfies correctness, EUF-CMA security and verification soundness. Let \( F : \mathcal{K} \times \{0,1\}^n \rightarrow \{0,1\}^\ell \) be a PRF, with key space \( \mathcal{K} \). Then, the watermarking scheme of Figure 8 satisfies the detection correctness, implementation property-preserving with error \( \varepsilon_{\text{prf}}(\rho, \delta) \)-unremovability and \((\gamma, \tau)\)-unforgeability, where \( \varepsilon_{\text{prf}} \) is the security of the PRF, \( \rho = \delta = 1/\operatorname{poly}(\lambda), \gamma \geq 1 - \frac{\rho \delta}{\lambda} + \tau \) and \( \tau < \frac{\delta^2}{\gamma} \).

- **WGen**: On input \( 1^\lambda \), it chooses uniformly at random a key \( K \) for a pseudorandom function \( F : \mathcal{K} \times \{0,1\}^n \rightarrow \{0,1\}^\ell \). It outputs \( mk = dk = K \), initializes the public variable \( \text{state} \leftarrow 0 \) and sets \( i \leftarrow 0 \). It chooses \( \rho, \delta \) such that \( \rho = \delta = 1/\operatorname{poly}(\lambda) \).
- **Mark**: On input \( K \), state, marked, compute \( i = \text{state} + 1 \) and run \( \text{SigGen}(1^\lambda) \) with randomness \( F(K, i) \). The output is a public-secret key pair \((\text{sk}_i, \text{pk}_i)\) and the algorithm returns a pair of circuits \((\text{Sign}_{\text{sk}_i}, \text{Verify}_{\text{sk}_i})\). Set as \( D_i \) the uniform distribution over the message space \( \mathcal{M} \). Then, set \( \text{state} \leftarrow \text{state} + 1 \).
- **Detect**: On input \( K \), a circuit \( C \) and state, for \( i = 1 \) to state:
  - Run \( \text{SigGen}(1^\lambda) \) with randomness \( F(K, i) \) (as the Mark algorithm does) in order to obtain \((\text{sk}_i, \text{pk}_i)\).
  - Choose \( k = \frac{1}{\rho^2} \) messages independently and uniformly at random from the message space \( \mathcal{M} \), i.e. \( m_1, \ldots, m_k \), and for any \( j \in \{1, \ldots, k\} \) run circuit \( C \) on input \( m_j \). \( C \) outputs \( \sigma_1, \ldots, \sigma_k \).
  - For \( j = 1 \) to \( k \) check whether \( \text{Verify}_{\text{sk}_i}(\sigma_j, m_j) = 1 \). If this is true for at least \( \frac{3}{4} \) messages, return marked. Otherwise, return unmarked.

**FIGURE 8** Watermarked signature scheme

To avoid repetition, due to the similarities in the proof of Theorem 2 with the proof of Theorem 1, below we provide a discussion on differences and similarities in the way each property of the watermarking scheme for digital signatures is proven, comparing to the corresponding one in Theorem 1.

**Lemma 6** If the digital signature scheme \((\text{SigGen}, \text{Sign}, \text{Verify})\) satisfies the EUF-CMA security, perfect correctness, verification soundness and \( F : \mathcal{K} \times \{0,1\}^n \rightarrow \{0,1\}^\ell \) is a PRF, then the watermarking scheme of Figure 8 is the implementation property-preserving according to Definition 12.

**Sketch**: The EUF-CMA security of the watermarked scheme relies on the security of the PRF. Given that \((\text{SigGen}, \text{Sign}, \text{Verify})\) satisfies EUF-CMA, a PRF distinguisher distinguishes a random function from a PRF by emulating the EUF-CMA game for either \((\text{SigGen}, \text{Sign}, \text{Verify})\) or the watermarked scheme depending on the oracle function. Correctness and verification soundness are preserved with no error since they hold for any pair of keys generated by \( \text{SigGen}(1^\lambda) \) which implies that they hold for the subset of pairs generated by \( \text{SigGen}(1^\lambda) \) using as randomness a PRF output.

**Lemma 7** The watermarking scheme of Figure 8 satisfies the detection correctness under the assumption that the underlying digital signature scheme \((\text{SigGen}, \text{Sign}, \text{Verify})\) satisfies perfect correctness.

Detection Correctness is trivial and can be proved as in Lemma 2.
**Lemma 8** ($\rho, \delta$)-unremovability. The scheme of Figure 8 satisfies $\gamma$ ($\tau, r$)-unremovability, where $\rho = \delta = 1 / \text{poly} (\lambda)$, under the assumption that the underlying digital signature scheme ($\text{SigGen}, \text{Sign}, \text{Verify}$) satisfies the perfect correctness.

**Sketch:** Similar to the proof of Lemma 3, this property is proven using counting arguments independent of the adversary's strategy relying only on the correctness of the underlying signature scheme. In particular, we fix $(i, (\text{Sign}_{sk}, \text{Verify}_{vk})) \in \text{Marked}$ and assume that $C^*$ is $(\gamma, \tau)$-close to $\text{Sign}_{sk}$. Then, we define a random variable which counts the number of messages (chosen uniformly at random from $\mathcal{M}$) that addressed the query position of $\text{Sign}_{sk}$ and, therefore, $C^*$. By a Chernoff bound, we show that at least $\frac{\gamma}{2}$ messages are checked correctly with overwhelming probability and thus, Detect returns unmarked only with negligible probability.

**Lemma 9** ($\gamma, \tau$)-unforgeability. If the function $F : K \times \{0, 1\}^n \rightarrow \{0, 1\}^r$ is a PRF function and the digital signature scheme ($\text{SigGen}, \text{Sign}, \text{Verify}$) satisfies the EUF-CMA security, perfect correctness and verification soundness, then the scheme of Figure 8 satisfies ($\gamma, \tau$)-unforgeability, where $\gamma \geq 1 - \frac{\rho^3}{3} + \tau$ and $\gamma < \frac{\rho^3}{3}$.

**Sketch:** We prove ($\gamma$, $\tau$)-unforgeability by reducing it to the EUF-CMA security of the watermarked signature scheme. Specifically, we construct a PPT adversary $B$ which breaks EUF-CMA by utilizing a PPT adversary $A$ against a modified unforgeability game where the PRF is substituted by a random function. We assume that $A$ wins with a non-negligible probability. The general idea of the reduction is similar to that of Figure 7 in Lemma 5. The adversary $B$ plays the verification key received by the EUF-CMA Challenger into one of the ChallengeOracle queries of $A$ by randomly guessing the 'attacked query position' of $A$. When $A$ outputs a circuit $C^*$, $B$ chooses $\rho^3 / 3$ messages uniformly at random from the space and provides them as inputs to $C^*$. Then, it checks if at least one message-signature pair output by $C^*$ is verified correctly under the verification key of the EUF-CMA Challenger. If there is such a message-signature pair $(m^*, \sigma^*)$, $B$ wins the EUF-CMA security game by returning the pair $(m^*, \sigma^*)$ as 'forgery'.

To verify the definition of ($\gamma, \tau$)-unforgeability (Definition 11), that $A$ is allowed to obtain a number of signing keys (in the form of Signing circuits) by making queries to the CorruptOracle. $A$ wins if $C^*$ is ($\gamma, \tau$)-far from all the signing circuits in the Corrupted set and at the same time, Detect returns marked. Similarly to the proof of Lemma 5, we first show that the reason that Detect returns marked is because $\frac{\gamma}{2}m = \frac{\gamma}{2}m + \frac{\gamma}{2}m$ message-signatures have passed the verification test under a verification key for some non-corrupted signing circuit. This is implied by Claim 9.1.

**Claim 9.1:** Assume that $C^*$ is ($\gamma, \tau$)-far from a signing circuit $\text{Sign}_{sk}$, where $\text{Sign}_{sk} \in \text{Corrupted}$, and that we choose uniformly at random $\lambda / \rho^3$ messages. Then.Verify for the corresponding key $\text{Verify}_{vk}$ returns 1 for more than $\frac{\gamma}{2}$ outputs/signatures of $C^*$ with negligible probability.

**Proof of Claim 9.1:** Let $R_1$ be the randomness space of $\text{CMA}$ and $R_2$ be the randomness space of $\text{Sign}_{sk}$. For $(m, r) \in \mathcal{M} \times R_1$, we define the random variable

\[
X_{ij} (m, r) = \begin{cases} 1, & \text{if } \text{Verify}_{vk}(m, C^*(m; r)) = 1 \\ 0, & \text{otherwise} \end{cases}
\]

We have that

\[
\Pr_{r \in R_1 \leftarrow \lambda} [X_{ij} = 1] = \Pr_{r \in R_1 \leftarrow \lambda} [\text{Verify}_{vk}(m, C^*(m; r)) = 1].
\]

By Definition 2 of verification soundness, a relation $R_{sk}$ is defined as $R_{sk} = \{ (vk_i, (m, \sigma)) \mid m \in \mathcal{M}, \exists r \in R_2 \text{ s.t. } \sigma = \text{Sign}_{sk}(m; r) \}$. By Lemma 6, the watermarking scheme is implementation property-preserving, which means that it satisfies verification soundness. Hence,

\[
\Pr_{r \in R_1 \leftarrow \lambda} [X_{ij} = 1 \land (vk_i, (m, C^*(m; r)) \notin R_{sk}] = \text{negl}(\lambda).
\]

Since the watermarking scheme preserves the correctness property (Lemma 6), \(\Pr[X_{ij} = 1 \mid (vk_i, (m, C^*(m; r)) \in R_{sk}] = 1\). Given that and Equation (17),

\[
\Pr[X_{ij} = 1] = \Pr_{r \in R_1 \leftarrow \lambda} [X_{ij} = 1 \land (vk_i, (m, C^*(m; r)) \in R_{sk}] + \text{negl}(\lambda).
\]

By the assumption on $C^*$, Definition 9 translates to this case as follows: There is $\mathcal{M}' \subseteq \mathcal{M}$ with $|\mathcal{M}'| \geq \gamma |\mathcal{M}|$ such that $\forall m \in \mathcal{M}'$, $\Pr_{r \in R_2 \leftarrow \lambda} [C^*(m; r)] = \text{Sign}_{sk}(m; r) \leq \tau$. Thus, $\forall m \in \mathcal{M}'$, $\Pr [(vk_i, (m, C^*(m; r)) \in R_{sk}] = \leq \tau$. By conditioning on the event $m \in \mathcal{M}'$, we can show that $\Pr \{ (vk_i, (m, C^*(m; r)) \in R_{sk}] = \leq \tau + 1 \cdot (1 - \gamma)$. By Equation (18), $\Pr [X_{ij} = 1] = \leq \tau + (1 - \gamma) + \text{negl}(\lambda)$. By defining the random variable $X_i = \sum_{\geq 1} X_{ij}$ and similarly to Lemma 4, we can show that Detect for the corresponding key $\text{Verify}_{vk}$ returns 1 for more than $\frac{\gamma}{2}$ outputs/signatures of $C^*$ with negligible probability. (End of Claim 9.1)
Given Claim 9.1, $C^*$ verifies correctly $\frac{1}{2^p}$ out of $\frac{1}{2^p}$ message-signature pairs under a verification key which belongs to a tuple in Marked \Corrupted set. If this verification key is the one given to $B$ by the EUF-CMA Challenger, then $B$ wins by just returning one message-signature pair among the above, assuming it has guessed correctly.

6 | WATERMARKING PUBLIC KEY CRYPTOGRAPHIC FUNCTIONALITIES AND DIGITAL SIGNATURES FUNCTIONALITIES

In Section 5, we presented watermarking constructions with secret key detection for both for public-key encryption and signature schemes. On the side of watermarking public-key cryptographic functionalities, in a similar spirit with the constructions of Section 5, watermarking constructions for public-key encryption and digital signature functionalities can be obtained from IBE [30] and Identity-based signatures (IBS) [30] respectively, that consider public detection. As shown in [1], a watermarking scheme for public-key encryption can be obtained from any IBE scheme. In particular, the master secret key of the IBE scheme corresponds to the marking key while the detection key is just the public parameters of the IBE scheme. Then, upon a marking request, the watermarking authority increments a counter by $1$ (which is initialized as $0$) and returns a public-secret key pair for the identity which equals the current value of the counter. This is accomplished by simply running the key-extraction algorithm of the IBE scheme. It is easy to see that any entity given the last value of the counter, which is stored as the public state, can re-generate all the public keys for the identities with values at most the counter, only based on the public parameters. Thus, given a circuit $C$, any entity can perform the same procedure as the detect algorithm of Figure 6 and decide whether a circuit is marked or not. In a similar way, a watermarking scheme for the digital signature functionality can be obtained from any IBS scheme, if we additionally assume that the IBS scheme satisfies verification soundness property, considering the results of Section 5.2.

7 | CONCLUSIONS AND FUTURE WORK

We proposed a watermarking model for watermarking public-key cryptographic functionalities and public-key schemes, providing a meaningful, unified model for both public-key encryption and digital signatures. Based on our model, we present watermarking constructions for public-key encryption and digital signature schemes that are based on standard assumptions and only rely on simple primitives such as PRFs.

An open question is to construct watermarked public-key encryption and digital signatures where the Detection/Extraction algorithm is sub-linear or even independent of the total number of marked circuits, without requiring shared state and while only relying standard assumptions and being practically efficient.

ACKNOWLEDGMENTS

The author was supported by the project CODAMODA, funded by ERC as well as supplementary funding by the General Secretariat for Research and Innovation, Greece. Research partly supported by EU Project No. 780477, PRIVILEGE.

REFERENCES

1. Baldimtsi, F., Kiyias, A., Samari, K.: Watermarking public-key cryptographic functionalities and implementations. ISC 2017, 173–191 (November 2017)
2. Baldimtsi, F., Kiyias, A., Samari, K.: Watermarking probabilistic circuits: the case of digital signatures, to appear in. Proceedings of Balkan-CryptSec (2018)
3. Adelsbach, A., Katzenbeisser, S., Veith, H.: Watermarking schemes provably secure against copy and ambiguity attacks, pp. 111–119. ACM Workshop on Digital Rights Management (October 2003)
4. Cox, I.J., et al.: Digital Watermarking, vol. 1558607145. Springer (2002)
5. Podlichuk, C.I., Delp, E.J.: Digital watermarking: algorithms and applications. IEEE Signal Process. Mag. 18(4), 33–46 (2001)
6. Potdar, VM., Han, S., Chang, E.: A survey of digital image watermarking techniques. INDIN 2005, 709–716 (September 2005)
7. Hopper, N., Molnar, D., Wagner, D.: From weak to strong watermarking, pp. 362–382. TCC 2007 (February 2007)
8. Barak, B., et al.: On the (im)possibility of obfuscating programs. CRYPTO 2001, 1–18 (August 2001)
9. Barak, B., et al.: On the (im)possibility of obfuscating programs. J. ACM. 59(2), 6 (2012)
10. Garg, S., et al.: Candidate indistinguishability obfuscation and functional encryption for all circuits, pp. 40–49. FOCS 2013 (October 2013)
11. Nishimaki, R.: How to watermark cryptographic functions. In: EUROCRYPT 2013, pp. 111–125. Springer, Berlin (2013)
12. Nishimaki, R.: ‘How to watermark cryptographic functions’, (2014). IACR Cryptology ePrint Archive Report 2014/472. http://eprint.iacr.org/2014/472
13. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. STOC 2008, 187–196 (May 2008)
14. Naccache, D., Shamir, A., Stern, J.P.: How to copyright a function? pp. 188–196. PKC 1999 (March 1999)
15. Cohen, A., et al.: Watermarking cryptographic capabilities, pp. 1115–1126. STOC 2016 (June 2016)
16. Cohen, A., et al.: Watermarking cryptographic capabilities. SIAM J. Comput. 47(6), 2157–2202 (2018)
17. Cohen, A., Holmgren, J., Vaikuntanathan, V.: ‘Publicly verifiable software watermarking’, IACR Cryptology ePrint Archive Report 2015/373 (2015). http://eprint.iacr.org/2015/373
18. Nishimaki, R., Wichs, D.: ‘Watermarking cryptographic programs against arbitrary removal strategies’, IACR Cryptology ePrint Archive Report 2015/344 (2015). http://eprint.iacr.org/2015/344
19. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption and more, pp. 475–484. STOC 2014 (June 2014)
20. Kiyias, A., et al.: Delegatable pseudorandom functions and applications, pp. 669–684. CCS 2013 (November 2013)
21. Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. ASIACRYPT 2013, 280–300 (December 2013)
22. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. PKC 2014, 501–519 (March 2014)
23. Yang, R., et al.: Unforgeable watermarking schemes with public extraction. SCN 2018, 63–80 (September 2018)
24. Boneh, D., Lewi, K., Wu, D.J.: Constraining pseudorandom functions privately. PKC 2017, 494–524 (March 2017)
25. Kim, S., Wu, D.J.: Watermarking cryptographic functionalities from standard lattice assumptions. CRYPTO 2017, 503–536 (August 2017)
APPENDIX A

SigGen: On input \( \lambda \), it chooses public parameters \( \langle p, q, g, H \rangle \), where \( g \) is a generator of a group \( G \) of order \( q \) and \( H : \{0, 1\}^* \rightarrow \mathbb{Z}_q \) is a hash function. Then, it chooses \( x \in \mathbb{Z}_q \) and computes \( y = g^x \bmod p \). It sets \( vk = y \) and \( sk = x \).

Sign: On input \( sk, M \), it chooses \( k \in \mathbb{Z}_q \), computes \( r = g^k \bmod p \), \( e = H(r\|M) \) and \( s = k - xe \). It outputs \( \sigma = (s, e) \).

Verify: On input \( vk, \sigma = (s, e) \), it computes \( r' = g^s \bmod p \) and \( e' = H(r'\|M) \). Then, \( e' = e \) it returns 1, otherwise it returns 0.

Proposition A 2 The Schnorr scheme for digital signatures satisfies verification soundness as defined in Definition 2.

Proof A Let \( (vk, sk) \) be a pair of keys generated by the Gen algorithm, that is \( sk = x \) and \( vk = y^k \bmod p \), and \( (M, (s, e)) \) a message-signature pair such that \( \text{Verify}_{sk}(M, (s, e)) = 1 \). Notice that the randomness space in the Schnorr signature scheme is \( \mathbb{Z}_q \). We will show that if there is pair \( (M, (s, e)) \) such that \( \text{Verify}_{vk}(M, (s, e)) = 1 \), then there is \( k \in \mathbb{Z}_q \) such that \( \text{Sign}_{sk}(M; k) = (s, e) \).

Since \( \text{Verify}_{vk}(M, (s, e)) = 1 \), it holds that \( r' = g^{r+xe} \bmod p \) and \( e = H(r'\|M) \). We fix the value \( k = s + xe \bmod q \) and we run the algorithm Sign on input the secret key \( x \) and the message \( M \) by using the value \( k = s + xe \) as a random value. Then, the algorithm Sign will compute \( r^* = g^k \bmod p \), \( e^* = H(r^*\|M) \) and \( s^* = k - xe \) and will output \( (s^*, e^*) \). Since \( r^* = r' \) we have that \( e^* = e \) and also. \( s^* = s \). Therefore there is a value \( k = s + xe \) such that \( \text{Sign}_{sk}(M; k) = (s, e) \). □