Watermarking Public-Key Cryptographic Primitives

Rishab Goyal
UT Austin
goyal@utexas.edu

Sam Kim
Stanford University
skim13@cs.stanford.edu

Nathan Manohar
UCLA
nmanohar@cs.ucla.edu

Brent Waters
UT Austin and NTT Research
bwaters@cs.utexas.edu

David J. Wu
University of Virginia
dwu4@virginia.edu

Abstract

A software watermarking scheme enables users to embed a message or mark within a program while preserving its functionality. Moreover, it is difficult for an adversary to remove a watermark from a marked program without corrupting its behavior. Existing constructions of software watermarking from standard assumptions have focused exclusively on watermarking pseudorandom functions (PRFs).

In this work, we study watermarking public-key primitives such as the signing key of a digital signature scheme or the decryption key of a public-key (predicate) encryption scheme. While watermarking public-key primitives might seem more challenging than watermarking PRFs, we show how to construct watermarkable variants of these notions while only relying on standard, and oftentimes, minimal, assumptions. Our watermarkable signature scheme relies only on the minimal assumption that one-way functions exist and satisfies ideal properties such as public marking, public mark-extraction, and full collusion resistance. Our watermarkable public-key encryption schemes are built using techniques developed for the closely-related problem of traitor tracing. Notably, we obtain fully collusion resistant watermarkable attribute-based encryption in the private-key setting from the standard learning with errors assumption and a bounded collusion resistant watermarkable predicate encryption scheme with public mark-extraction and public marking from the minimal assumption that public-key encryption exists.

1 Introduction

Watermarking is a way to embed special information called a "mark" into digital objects such as images, videos, audio, or software so that the marked object has the same appearance or behavior as the original object. Moreover, it should be difficult for an adversary to remove the mark without damaging the object itself. Watermarking is a useful tool both for protecting ownership and for preventing unauthorized distribution of digital media.

Software watermarking. In this work, we focus on software watermarking for cryptographic functionalities. Barak et al. [BGI+01, BGI+12] and Hopper et al. [HMW07] provided the first rigorous mathematical framework for software watermarking. Very briefly, a software watermarking scheme consists of two main algorithms. First, there is a *marking* algorithm that takes as input a program, modeled as a Boolean circuit C, and outputs a new marked circuit C' with the property that C and C' agree almost everywhere. Second, there is an *extraction* algorithm that takes as

input a circuit C and outputs a bit indicating whether the program is marked or not. In the case of message-embedding watermarking, the marking algorithm additionally takes a message τ as input, and the extraction algorithm will either output the mark τ or a special symbol $\perp$ to indicate an unmarked program. The primary security requirement is unremovability, which says that given a marked circuit C' with an embedded message τ, no efficient adversary can construct a new circuit $\tilde{C}'$ that has roughly the same behavior as C', and yet, the extraction algorithm on $\tilde{C}'$ fails to output τ. Notably, there are no restrictions on the circuit the adversary can output (other than the requirement that the adversary be efficient). This notion of security is often referred to as security against *arbitrary removal strategies* and captures the intuitive notion of watermarking where an adversary cannot replicate a program's functionality without also preserving the watermark.

Realizing the strong security requirements put forth in the early works on cryptographic watermarking [BGI+01, HMW07, BGI+12] has proven challenging. In fact, Barak et al. showed an impossibility result (under indistinguishability obfuscation) on the existence of watermarking schemes that are *perfectly functionality-preserving* (i.e., schemes where the input/output behavior of the marked function is identical to that of the original function). In light of this lower bound, early works [NSS99, YF11, Nis13] provided partial results for watermarking specific classes of cryptographic functionalities by imposing limitations on the adversary's ability to modify the program and remove the watermark.

The first positive result on constructing watermarking schemes with security against *arbitrary* adversarial strategies was due to Cohen et al. [CHN+16] who showed that if we relax the perfect functionality-preserving requirement to only require *statistical* functionality-preserving (i.e., the marked function only has to implement the original function almost everywhere), then watermarking is possible. Moreover, Cohen et al. showed how to watermark several classes of cryptographic primitives, including pseudorandom functions (PRFs) and public-key encryption, with strong security from indistinguishability obfuscation. Since the work of Cohen et al., a number of works have studied how to build watermarkable families of PRFs from weaker assumptions such as lattice-based assumptions [KW17, KW19] or CCA-secure encryption [QWZ18].

Watermarking public-key primitives. Existing constructions of software watermarking from standard cryptographic assumptions all focus on watermarking symmetric-key primitives, notably, PRFs [KW17, QWZ18, KW19]. The one exception is the work of Baldimtsi et al. [BKS17], who showed how to watermark public-key cryptographic primitives, but in a stateful setting, and under a modified security model where a trusted watermarking authority generates *both* unmarked and marked keys.[1] Our focus in this work is to study watermarking for two broad classes of public-key cryptographic primitives: digital signatures and public-key encryption.

1.1 Our Results and Approach

In this work, we consider watermarking for public-key primitives such as digital signatures and public-key encryption. We make contributions along two dimensions. On the definitional side, we introduce a more refined set of definitions for watermarking public-key primitives that better capture our intuitive notions of functionality-preserving and watermarking security than the previous definitions of Cohen et al. [CHN+16]. On the constructions side, we provide new watermarkable

[1] In the standard watermarking model, anyone can generate keys (without going through or trusting the watermarking authority), and at a later time, decide if they want to mark the keys or not.

signature and public-key encryption schemes that rely only on standard assumptions. We provide an overview of each of our contributions below.

Limitations of the Cohen et al. framework. To motivate our new definitions for watermarking public-key primitives, we begin by identifying several limitations in the existing definitions put forth by Cohen et al. [CHN+16]. In existing definitions for watermarkable signatures, a central watermarking authority runs a setup algorithm that takes as input a security parameter λ and generates a pair of marking and extraction keys $(\mathsf{mk}, \mathsf{xk})$. The system has a key-generation algorithm which uses the marking key mk in conjunction with a mark τ to sample a signing/verification key pair $(\mathsf{sk}, \mathsf{vk})$. Here the signing key sk contains the embedded mark τ. The holder of the signing key sk can sign any message m, and the corresponding signature σ can be verified using the verification key vk. Finally, the watermarking scheme provides an extraction algorithm that takes a circuit C and the extraction key xk as input and either outputs a mark τ or declares the circuit to be "unmarked." The main security requirement is unremovability which states that the extraction algorithm outputs the correct mark τ as long as the circuit C and the marked signing circuit $\mathsf{Sign}(\mathsf{sk}, \cdot)$ behave *identically* on a sufficiently large fraction of input messages.

While this definition appears to capture our intuitive notion of a watermarkable signature scheme, there are several definitional issues that allow *intuitively insecure* constructions to be proven secure. We illustrate this with a simple example below.

A simple watermarkable signature scheme. Take any digital signature scheme, which consists of three algorithms: Setup (for sampling signing/verification keys), Sign (for signing messages), and Verify (for verifying signatures), and consider the following watermarkable signature scheme. An unmarked signing/verification key pair $(\mathsf{sk}, \mathsf{vk})$ is sampled by running the Setup algorithm for the underlying signature scheme. The signature on a message m consists of a signature σ on m under sk and a special symbol $\perp$. Concretely, we define $\mathsf{Sign}'(\mathsf{sk}, m) := (\mathsf{Sign}(\mathsf{sk}, m), \perp)$. To verify a signature $\sigma' = (\sigma, \mathsf{tag})$, the verification algorithm simply ignores the second element of the tuple (the tag) and accepts the signature as long as σ is a valid signature on the message. To *mark* a signing key with mark τ, one simply appends the mark to the unmarked key $\mathsf{sk}_\tau = (\mathsf{sk}, \tau)$. A signature under the marked key is computed identically as with the unmarked key except the symbol $\perp$ is replaced with the mark τ. Specifically, we define $\mathsf{Sign}'(\mathsf{sk}_\tau, m) := (\mathsf{Sign}(\mathsf{sk}, m), \tau)$. Note that such signatures are still accepted by the verifier.

To extract a mark from a signing circuit C, the extraction algorithm simply runs the circuit C on a polynomial number of randomly-chosen messages to obtain a sequence of signatures $(\sigma_1, \mathsf{tag}_1), \ldots, (\sigma_T, \mathsf{tag}_T)$ for some polynomial T. Then, if there exists a mark τ that appears in a majority of these signatures, it outputs τ and otherwise, it declares the circuit unmarked.

Intuitively, this basic construction does not seem secure because an adversary can trivially remove the watermark from a marked key by just not outputting the tag with each signature. In fact, the marked key even contains a description of the original unmarked key! However, this basic construction does satisfy the unremovability definition described above. Namely, if the claimed circuit C and the honestly-marked circuit $\mathsf{Sign}'(\mathsf{sk}_\tau, \cdot)$ behave *identically* on a sufficiently large fraction of the input messages, then the extraction algorithm will always returns the correct mark τ. The simple attack where the adversary simply strips out the tag from the output is *not allowed* by the existing unremovability definition because it changes the input/output functionality of the marked circuit nearly everywhere. At the same time, the "modified" circuit the adversary constructed is still a perfectly valid signing circuit (and in fact, it is as good as the original signing key). Thus, the existing definitions allow for constructions of watermarking schemes that, while provably secure

under the current definitional framework, fall short of capturing our intuitive notion of a secure watermarking scheme.

Towards a better definition. A similar issue arises in the existing definitions for watermarkable public-key encryption. In both settings, the issues arise because the existing unremovability definitions do not allow for adversaries that preserve the desired "functionality" of the underlying object and, yet, change the exact input/output behavior of the circuit. This gap in existing watermarking definitions for public-key primitives leads to simple but problematic constructions. We provide a more detailed discussion of these definitional issues in Appendix B.

The simple strawman construction described above highlights the need for a much stronger notion of unremovability in the setting of watermarking public-key primitives. From a conceptual standpoint, unremovability should hold against any adversary that manages to output a "useful" circuit, even if that circuit does not replicate the same input/output behavior as the marked circuit. In the case of signatures, a "useful" circuit would be one that outputs valid signatures (with respect to the verification key), while in the case of encryption, a "useful" circuit would be one that can decrypt valid ciphertexts. In this work, we address the limitations in the existing definitional framework for watermarking public-key primitives by introducing a more refined framework that strengthens the existing notions in the following ways:

- **Stronger unremovability property:** As discussed previously, existing unremovability definitions in the public-key setting are too restrictive and preclude adversaries that would be reasonable according to our conceptual notion of watermarking. In this work, we strengthen the unremovability definition to better capture the full range of realistic adversarial strategies.
- **Independent marking and key generation:** As mentioned before, existing watermarking definitions in the public-key setting also consider a joint key-generation and marking algorithm. In this work, we decouple the two algorithms (similar to the setting in the case of watermarking PRFs). This better models our conceptual view of watermarking where (unmarked) keys can be independently generated and, subsequently, marked.
- **Collusion resistance:** Lastly, existing watermarking constructions from standard assumptions provide security only against adversaries that receive a *single* marked circuit. This limits the applicability of such systems as the security of these system could be completely compromised if the adversary gets to see multiple circuits marked with distinct tags/identities. In this work, we consider watermarking schemes that remain secure even when the adversary gets to see *many* marked keys.

Redefining watermarkable public-key primitives. First, we note that separating the marking and key-generation algorithms, as well as strengthening the unremovability property to capture collusion resistance, are natural extensions of the existing framework. Expanding the unremovability definition to capture a broader range of adversarial strategies poses a more subtle challenge. At a high level, the question is what qualifies as a "useful" circuit in the case of watermarking public-key primitives. For secret-key primitives, such as PRFs, the notion of exact functionality preserving seems to capture usefulness. Recall that for watermarkable PRFs, this is simply captured by requiring that the adversarially-generated circuit and the honestly-marked PRF circuit compute the *same* function on a sufficiently large fraction of inputs.

With the goal of refining the notion of "useful" circuits in the public-key setting, we draw inspiration from the closely-related primitive of traitor tracing [CFN94]. Very briefly, a traditional traitor tracing scheme is an encryption system where there is a single master public key pk and N decryption keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$ with the property that ciphertexts encrypted under pk can be decrypted by any of the N decryption keys. Then, there is a special tracing algorithm, which given oracle access to any "useful" decoding device D, outputs a set of users $T \subseteq [N]$, where T contains the indices of the decryption keys used to build D. Here, we say a decoder D is "useful" if it can decrypt an honestly-generated ciphertext with non-negligible probability. This is a more natural definition of "usefulness" compared to the existing notions in watermarking, where usefulness is measured by the fraction of inputs on which decoder D and the decryption algorithm (with the decryption key sk_i hardwired) agree. Drawing inspiration from the traitor tracing definitions, we propose the following definitions of what it means for a signing (or decryption) circuit to be "useful," and we require that mark-unremovability hold against any adversary that manages to produce a "useful" circuit:

Signing. We say a signing circuit C is *useful* with respect to a verification key vk if, on a non-negligible fraction of input messages, it outputs a signature σ that is accepted under key vk. In particular, C is useful as long as it produces valid signatures, even if those signatures are not exactly the same as the ones output by an unmarked signing circuit.

Decryption. We say a decryption circuit C is *useful* with respect to a public key pk if, on a non-negligible fraction of honestly-sampled ciphertexts, it correctly decrypts the ciphertext.[2] We do not enforce any requirements on the behavior of the circuit C on ciphertexts that may not be well-formed (i.e., not in the support of the honest encryption algorithm). This seems reasonable because the behavior of the honest decryption algorithm on malformed ciphertexts is usually unspecified. It thus seems very restrictive to require that a *useful* decryption circuit preserve the input/output behavior of the unmarked decryption circuit on such ciphertexts.

Our watermarking definitions. We now give an overview of our notion of a watermarkable signature scheme. In our framework, the watermarking authority is independent of the key-generator for the signature scheme. The watermarking authority runs a setup algorithm to generate a tuple $(\mathsf{wpp}, \mathsf{mk}, \mathsf{xk})$ containing the watermarking public parameters wpp, the marking key mk, and the extraction key xk. The scheme supports a set of standard signature algorithms Setup, Sign, and Verify, where these algorithm are parameterized by the watermarking public parameters wpp. The signature-setup algorithm is used to sample *unmarked* signing/verification key-pairs $(\mathsf{sk}, \mathsf{vk})$, and the signing/verification algorithms are defined in the usual way. The marking algorithm takes the marking key mk, an (unmarked) signing key sk, and a mark τ and outputs a marked key sk_τ. The extraction algorithm takes a circuit C, the extraction key xk, and a verification key vk as inputs, and it outputs an embedded mark τ or declares the circuit to be unmarked. Here, the key vk represents the verification key associated with the claimed signing circuit C. Note that since the verification key is public, providing it to the extraction algorithm seems reasonable.

For security of the system, we first require that the tuple $(\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$ is itself a secure digital signature scheme, and we require this to be the case even if the adversary gets to choose

[2] Looking ahead, we define an even weaker notion of usefulness for decryption circuits, where the circuit C might only be useful for decrypting ciphertexts associated with two arbitrarily chosen (but fixed) messages m_0, m_1 instead of requiring decryption capability over entire message space. This is also inspired by recent developments and improvements in the traitor tracing literature [NWZ16, GKRW18, GKW18].

the watermarking parameters. Next, for unremovability, we require that the extraction algorithm succeeds in recovering the correct mark τ from any circuit C that is a "useful" signing circuit even if the adversary is able to see the same circuit marked with different (and adversarially-chosen) tags τ. The latter property models *collusion resistance*.

In addition to the above security requirements, the watermarking scheme must also satisfy an additional *functionality-preserving* guarantee. In previous watermarking works, this was defined in an *exact* sense where the marked circuit C' was required to implement the same input/output behavior as the original unmarked circuit C almost everywhere. Although we can define an analogous functionality-preserving property for public-key primitives, in this work, we consider a relaxation inspired by our strengthened unremovability definition. Namely, we only require the marked circuit to be a "useful" signing or decryption circuit. While it may be possible to achieve the stronger notion of *exact* functionality preserving in conjunction with the stronger notions of unremovability we introduce, we do not study this in our current work.

We define our notion of a watermarkable public-key encryption systems analogously, and we refer the reader to Section 4 for the full description. Lastly, in this work, we study watermarking schemes in both the public and private marking/extraction settings. Very briefly, in the public marking or extraction setting, the unremovability property must hold even when the adversary is given the marking or extraction keys, respectively.[3]

Our constructions. In addition to our new definitional framework for watermarking public-key primitives, we provide a number of constructions that satisfy our strengthened security definitions. All of our constructions rely only on standard assumptions and can be summarized as follows:

- **Digital signatures:** We build a fully-public and collusion resistant watermarkable signature scheme that satisfies our definitions above. Our construction relies only on the minimal assumption that digital signatures exist. We give this construction in Section 3.3.
- **Attribute-based encryption:** We construct a collusion resistant watermarkable attribute-based encryption in the secret-key setting (namely, the scheme assumes a secret marking key and a secret extraction key). Our construction relies on techniques from recent constructions of traitor tracing [GKW18, CVW+18]. Concretely, our construction relies on a mixed functional encryption scheme [GKW18] and a delegatable attribute-based encryption scheme [SW05, GPSW06]. Both these primitives [BGG+14, GKW18, CVW+18] can be instantiated from the learning with errors (LWE) assumption [Reg05]. We give this construction in Section 4.3.
- **Predicate encryption:** Finally, we construct a fully-public watermarkable public-key predicate encryption scheme with security against bounded collusions. This construction can be based on the existence of any standard public-key encryption scheme, which is again the minimal assumption. We give this construction in Appendix C.2.

We note that all existing constructions of watermarking [KW17, QWZ18, KW19] from standard assumptions are insecure in the collusion resistant setting, and an adversary that sees even *two* marked keys can easily remove the embedded marks. Therefore, our constructions provide the first constructions of watermarking that support either full or even bounded collusion resistance from standard assumptions.

[3] In the fully-public setting, the unremovability property must hold even if *both* the marking and the extraction keys are public.

Our work additionally provides the first watermarking constructions that achieve both public marking and security against the watermarking authority. A few recent works [QWZ18, KW19] provided watermarking constructions that support public marking from standard assumptions; however, this was at the expense of giving the watermarking authority a trapdoor that allowed it to break security of all of the keys in the system (including *unmarked* keys). In contrast, our schemes support both public marking and public extraction while retaining security even against a malicious watermarking authority. In fact, several of our constructions do not even require the existence of a watermarking authority!

At a philosophical level, our work shows that watermarking public-key cryptographic primitives is much less challenging than previously believed. It also highlights the need to further explore and identify the "right" set of definitions for software watermarking. Crucially, we believe that looking at watermarking through the lens of traitor tracing [CFN94] will yield many valuable insights.

Next, we describe our construction of a watermarkable signature scheme, and, subsequently, we provide a high level overview of the techniques used in constructing watermarkable attribute-based encryption.

Watermarking digital signatures. Our watermarkable digital signature scheme relies on constrained signatures (also known as policy-based signatures) [BF14, Tsa17]. In a constrained signature scheme over a message space $\mathcal{M}$, the signing key sk can be used to derive a constrained signing key sk_f for a particular predicate $f\colon \mathcal{M} \to \{0,1\}$ with the property that the constrained key sk_f can be used to sign all messages m where $f(m) = 1$. The security property is that an adversary who is given constrained keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$ for functions $f_1, \ldots, f_n$ cannot produce a valid signature on any message m where $f_i(m) = 0$ for all $i \in [n]$. It is straightforward to construct constrained signatures from any standard signature scheme using certificates [BF14], and we briefly recall this basic construction in Section 3.4.

A constrained signature scheme that supports the class of "prefix-based" constraints immediately yields a watermarkable signature scheme. In more detail, if we want to construct a watermarkable signature with message space $\mathcal{M}$ and mark space $\mathcal{T}$, we use a prefix-constrained signature scheme with message space $\mathcal{T} \times \mathcal{M}$. Signing and verification keys for the watermarkable signature directly correspond to signing and verification keys for the underlying prefix-constrained signature scheme. A signature on a message m consists of a tuple $\sigma_m = (\bot, \sigma')$ where σ' is a signature on $(\bot, m)$. To verify a signature $\sigma = (\tau, \sigma')$ on a message m, the verification algorithm checks that σ' is a valid signature on the pair (τ, m). Now, to mark a signing key with mark $\tau^* \in \mathcal{T}$, the user constrains the signing key sk to the prefix-based constraint $f_{\tau^*}\colon \mathcal{T} \times \mathcal{M} \to \{0,1\}$ where $f_{\tau^*}(\tau, x) = 1$ if $\tau^* = \tau$ and 0 otherwise. The marked circuit C_{τ^*} is a circuit that takes as input a message m and outputs (τ^*, σ'), where σ' is a signature on (τ^*, m) using the constrained key sk_{τ^*}. To extract a watermark from a candidate circuit C', simply sample a random message $m \leftarrow \mathcal{M}$,[4] compute $(\tau, \sigma') \leftarrow C'(m)$, and output τ if σ' is a valid signature on (τ, m). Note that if C' only succeeds in producing valid signatures with ε probability (for non-negligible ε), then this procedure can be repeated λ/ε times. If no marks are extracted after λ/ε iterations, then declare the circuit unmarked.

By correctness of the underlying constrained signature scheme, the marked circuit C_{τ^*} outputs valid signatures on all messages $m \in \mathcal{M}$, so the marked circuit is functionality-preserving (even

[4] More generally, we can consider a stronger notion of unremovability where we replace the uniform distribution over $\mathcal{M}$ with *any* (adversarially-chosen) efficiently-sampleable distribution over $\mathcal{M}$ where the circuit succeeds in generating valid signatures with non-negligible probability. Notably, the support of this distribution may have negligible density in $\mathcal{M}$.

though the signatures output by C are noticeably different than the signatures output by the original signing algorithm). Unremovability follows from security of the underlying constrained signature. Namely, an adversary who only has signing circuits marked with $\tau_1, \ldots, \tau_n$ should only be able to compute signatures on tuples of the form (τ_i, m) for $i \in [n]$. Thus, if the extraction algorithm outputs some $\tau' \neq \tau_i$ for all $i \in [n]$, then the adversary's circuit must have forged a valid signature on (τ', m) for some message $m \in \mathcal{M}$, which breaks security of the underlying constrained signature scheme. In addition, if the underlying constrained signature scheme is collusion resistant (i.e., security holds against adversaries that obtain an a priori unbounded polynomial number of constrained keys), then the resulting watermarkable signature scheme is also collusion resistant. We describe this construction and its security analysis in greater detail in Section 3.

Watermarking public-key encryption schemes. Our watermarkable public-key encryption scheme is heavily inspired by the recent traitor tracing construction of Goyal et al. [GKW18]. An important component of their approach was the introduction of a special type of functional encryption (FE) called mixed functional encryption (mixed FE). Goyal et al. showed that mixed FE in conjunction with an attribute-based encryption (ABE) scheme for a sufficiently expressive predicate class was sufficient for traitor tracing. In this work, we show that similar techniques are also useful for watermarking various forms of public-key encryption schemes. We provide a brief overview here and defer the full technical details to Section 4.

First, a mixed FE scheme is identical to a regular public-key FE scheme in that it consists of setup, encryption, key-generation, and decryption algorithms. In addition, it also has a special *secret-key encryption* algorithm. The setup algorithm samples the master public/secret key-pair $(\mathsf{mpk}, \mathsf{msk})$. The normal (public-key) encryption algorithm only takes as input the public key mpk, and outputs a normal (public-key) ciphertext ct. The secret-key encryption algorithm takes as input the master secret key msk and the description of a binary-valued function f, and outputs a (secret-key) ciphertext ct_f. Given the master key msk, the key-generation algorithm allows one to generate a key sk_m for any input message m. Now, a user can use a secret key sk_m to decrypt a ciphertext. The correctness requirement says that decrypting a secret-key ciphertext ct_f with the secret-key sk_m should output $f(m) \in \{0, 1\}$, while decrypting a public-key ciphertext ct should always output 1, irrespective of the value of m. Security of a mixed FE scheme consists of two properties. The first is the standard FE indistinguishability property which requires that secret-key encryptions of two functions f_0 and f_1 are indistinguishable as long as $f_0(m) = f_1(m)$ for every key sk_m the adversary possesses. The second property states that it should be hard to distinguish between a public-key ciphertext ct and a secret-key encryption ct_f of a function f as long as $f(m) = 1$ for every key sk_m the adversary possesses.

At a high level, our watermarkable public-key encryption scheme is very similar to the traitor tracing scheme by Goyal et al. In particular, the public watermarking parameters are set to be a mixed FE public key. The underlying (watermarkable) public-key encryption system is instantiated using an ABE scheme. An encryption of a message is then an ABE encryption of the corresponding message with the attribute set to a freshly sampled mixed FE public-key ciphertext. Given the master secret key of the ABE scheme, a decryptor can always recover the encrypted message. To mark the decryption key, the marking algorithm first generates a mixed FE secret key for the corresponding mark. The marked key is then an ABE decryption key, where the function is mixed FE decryption (with the mixed FE secret key hardwired within). The intuition is that when the attribute is a normal mixed FE ciphertext (i.e., when the ciphertexts are honestly-generated), then all of the mixed FE keys decrypt to 1, in which case, ABE decryption successfully recovers the

encrypted message. However, if the attribute is set to be a mixed FE secret-key ciphertext, then we can control the marked keys for which decryption still works. This can be leveraged to obtain a mark-extraction algorithm using classic techniques for traitor tracing [BSW06, NWZ16, GKW18] (via private linear broadcast encryption). We provide the full description in Section 4.

Watermarking advanced public-key functionalities. Next, we show that if the underlying ABE scheme supports "key-delegation", the above template naturally extends to give a fully collusion resistant watermarkable ABE scheme. In this case, the watermarking scheme supports marking ABE decryption keys sk_f (for a specific predicate f). Using constructions of mixed FE [GKW18, CVW+18] and delegatable ABE [BGG+14] from the standard LWE assumption, we obtain a watermarkable ABE scheme from LWE.

Then, in Appendix C, we show how to adapt similar traitor tracing techniques to build a watermarkable predicate encryption scheme. This scheme supports both public marking and public mark-extraction and can be based on the minimal assumption that public-key encryption exists. However, it is only collusion resistant against an a priori bounded number of users.

1.2 Additional Related Work

In this section, we survey some additional related work and compare our new watermarking notions to related notions studied in prior work.

Constrained signatures. Numerous works [MPR11, BGI14, BF14, Tsa17] have studied constructing constrained signatures (and variants thereof) together with properties like privacy, anonymity, succinct keys, or succinct signatures.

Traitor tracing. Since the work of Chor et al. [CFN94], there have been a vast number of constructions of fully collusion resistant traitor tracing from combinatorial constructions [BN08, Sir06], pairing-based assumptions [BSW06, BW06, GKSW10, Fre10, LCW13, LW15, GKW19], lattice-based assumptions [GKW18, CVW+18, GKW19], and indistinguishability obfuscation [BZ14, NWZ16]. With the exception of [NWZ16, GKW19], the existing constructions only support efficient tracing over a polynomial-size identity space. There are also numerous constructions that provide security in the bounded collusion setting [CFN94, SW98, KD98, BF99, CFNP00, SSW01, PST06, ADM+07, FNP07, BP08, LPSS14, ABP+17].

Attribute-based traitor tracing. Directly relevant to our notion of watermarkable predicate encryption is the notion of attribute-based traitor tracing [ADM+07, LCW13, LW15, CVW+18], which is a hybrid of attribute-based encryption and traitor tracing. The main difference between these two notions is that in the traitor-tracing setting, the marking and key-generation algorithms are combined (namely, the key-generation algorithm takes as input the function together with the mark). In watermarking, we have the additional flexibility that we can embed the watermark *after* issuing the key as well as support watermarking *adversarially-chosen* keys. When considering the simpler notion of watermarkable public-key encryption and traitor tracing, we can equate these two notions with a suitable redefinition of the traitor tracing schema (assuming that the traitor tracing scheme supports a *public* tracing algorithm). However, this equivalence does not seem to extend to the setting of attribute-based encryption or predicate encryption. Another key difference is that existing constructions of attribute-based traitor tracing from standard assumptions only support tracing over a polynomial-size identity space, while in the standard notions of message-embedding watermarking, the identity space is exponential. Thus, our results give a collusion resistant attribute-based traitor tracing scheme that supports an exponential number of possible identities.

2 Preliminaries

Notation. For an integer n, we write $[n] := \{1, \ldots, n\}$ to denote the set of all positive integers up to n. For integers $n \geq m \geq 1$, we write $[m, n]$ to denote the set of integers $\{m, m+1, \ldots, n\}$, and $[m, n]_{\mathbb{R}}$ to denote the closed interval between m and n (inclusive) over the real numbers. Unless specified otherwise, all polynomials that we consider in the paper are positive polynomials. Also, we represent each finite set of integers $S \subset \mathbb{N}$ as an *ordered* set $S = \{i_1, i_2, \ldots, i_n\}$; namely, $i_j < i_k$ for every $1 \leq j < k \leq n$. For sets $\mathcal{X}$ and $\mathcal{Y}$, we write $\mathsf{Funs}[\mathcal{X}, \mathcal{Y}]$ to denote the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$. For any finite set S, we write $x \leftarrow S$ to denote a uniformly random element x drawn from the set S. Similarly, for any distribution $\mathcal{D}$, $x \leftarrow \mathcal{D}$ denotes an element x drawn from distribution $\mathcal{D}$. The distribution $\mathcal{D}^n$ is used to represent a distribution over vectors of n components, where each component is drawn independently from the distribution $\mathcal{D}$.

We use λ (often implicitly) to denote the security parameter. We write $\mathsf{poly}(\lambda)$ to denote a quantity that is bounded by a fixed polynomial in λ and $\mathsf{negl}(\lambda)$ to denote a function that is $o(1/\lambda^c)$ for all $c \in \mathbb{N}$. We say that an event occurs with overwhelming probability if its complement occurs with negligible probability. We say an algorithm is efficient if it runs in probabilistic polynomial time (PPT) in the length of its input.

Statistical distance and tail bounds. For two families of distributions $\mathcal{D}_1 = \{\mathcal{D}_{1,\lambda}\}_{\lambda \in \mathbb{N}}$ and $\mathcal{D}_2 = \{\mathcal{D}_{2,\lambda}\}_{\lambda \in \mathbb{N}}$, we write $\mathcal{D}_1 \overset{c}{\approx} \mathcal{D}_2$ if the two distributions are computationally indistinguishable (i.e., no efficient algorithm can distinguish distribution $\mathcal{D}_1$ from $\mathcal{D}_2$ except with negligible probability), and $\mathcal{D}_1 \overset{s}{\approx} \mathcal{D}_2$ if the two distributions are statistically indistinguishable (i.e., the statistical distance between $\mathcal{D}_1$ and $\mathcal{D}_2$ is $\mathsf{negl}(\lambda)$). We recall the Hoeffding's inequality that we use in our analysis:

Fact 2.1 (Hoeffding's Inequality [Hoe63])**.** Let $X_1, \ldots, X_n$ be independent random variables where $0 \leq X_i \leq 1$ for all $i \in [n]$. Let $S = \sum_{i \in [n]} X_i$ and let $\mathbb{E}[S]$ denote the expected value of S. Then, for any $t \geq 0$,

$$\Pr\left[\, |S - \mathbb{E}[S]| \geq nt \right] \leq 2^{-\Omega(nt^2)}.$$

Randomized circuits. A randomized circuit with domain $\mathcal{X}$, randomness space $\mathcal{R}$, and range $\mathcal{Y}$ is a circuit $C \colon \mathcal{X} \times \mathcal{R} \to \mathcal{Y}$. The evaluation of a randomized circuit C on an input $x \in \mathcal{X}$ is defined to be $y = C(x; r)$ for $r \leftarrow \mathcal{R}$. For all randomized circuits that we consider in this work, the number of random coins that they require will always be polynomially related to the input size of the circuit (i.e., $\log |\mathcal{R}| = \mathsf{poly}(\log |\mathcal{X}|)$). To simplify the notation, we will omit specifying the exact randomness space of a randomized circuit C and simply identify it with respect to its domain and range $C \colon \mathcal{X} \to \mathcal{Y}$ (as opposed to $C \colon \mathcal{X} \times \mathcal{R} \to \mathcal{Y}$).

Basic cryptographic primitives. We recall the definitions of several basic cryptographic primitives (e.g., public-key encryption, digital signatures, etc.) in Appendix A.

3 Watermarkable Signature Schemes

In this section, we show how to watermark a digital signature scheme. We begin by defining our notion of a watermarkable signature scheme. Our definition extends the existing definitions of Cohen et al. [CHN+16] for watermarking signatures, and it also incorporates different aspects of the definitions of watermarkable PRFs [CHN+16, BLW17, KW17, YAL+17, QWZ18, KW19] and traitor

tracing [NWZ16, GKRW18]. We present our construction in the fully public-key setting (namely, both marking and extraction are public operations, and there is no watermarking secret key).

3.1 Defining Watermarkable Signatures

Cohen et al. [CHN+16] provided the first formal definition of a watermarkable signature scheme. Although their definition captures the main intuition for a watermarkable digital signature scheme, it is limited in that the signature key generation algorithm and the watermarking key generation algorithm are combined into a *single* algorithm, and moreover, their unremovability definition falls short of capturing the full range of reasonable adversarial strategies. These issues limit the usefulness of the resulting watermarking scheme. We provide a more detailed comparison of our definitions with the previous one by Cohen et al. in Appendix B.

In this work, we modify the watermarking definition so that the signature keys and the marked keys are generated by two *independent* algorithms (as well as consider a stronger notion of unremovability). Having independent key-generation and marking algorithms is the standard syntax in the case of watermarking symmetric primitives like PRFs [CHN+16, BLW17, KW17, YAL+17, QWZ18, KW19]; namely, the marking algorithm is independent of the key-generation algorithm. By introducing a stronger and more refined definition, we are additionally able to model security against a *malicious* watermarking authority that attempts to forge signatures for honest users in the system. Furthermore, as the signature scheme and the watermarking scheme are decoupled, we can also model security against a *colluding set of malicious users* that hold copies of the same signing key, each marked with a different message. Existing watermarking schemes for PRFs from standard assumptions are not collusion resistant. In fact, in all of the current constructions from standard assumptions, an adversary that sees even *two* marked keys can easily remove the watermark.

Syntax. A watermarkable signature scheme with mark space $\mathcal{T}$ is a tuple of algorithms $\Pi_{\mathsf{WM}} = (\mathsf{WMSetup}, \mathsf{SigSetup}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Mark}, \mathsf{Extract})$ with the following syntax:

$\mathsf{WMSetup}(1^\lambda) \to (\mathsf{wpp}, \mathsf{mk}, \mathsf{xk})$. On input the security parameter λ, the watermarking setup algorithm outputs a set of public parameters wpp, a marking key mk, and an extraction key xk.

$\mathsf{SigSetup}(1^\lambda, \mathsf{wpp}) \to (\mathsf{vk}, \mathsf{sk})$. On input the security parameter λ and the watermarking public parameters wpp, the signature scheme setup algorithm outputs an (unmarked) signing-verification key pair $(\mathsf{vk}, \mathsf{sk})$.

The message space $\mathcal{M}$, verification key space $\mathcal{VK}$, signing key space $\mathcal{SK}$, and signature space $\mathcal{SIG}$ associated with the signature scheme are parameterized by the watermarking public parameters wpp. When the public parameters wpp are empty, then these components are functions of only the security parameter.

$\mathsf{Mark}(\mathsf{mk}, \mathsf{sk}, \tau) \to C$. On input the marking key mk, a signing key $\mathsf{sk} \in \mathcal{SK}$, and a mark $\tau \in \mathcal{T}$, the marking algorithm outputs a marked circuit $C \colon \mathcal{M} \to \mathcal{SIG}$.

$\mathsf{Sign}(\mathsf{sk}, m) \to \sigma$. On input an (unmarked) signing key sk and a message $m \in \mathcal{M}$, the signing algorithm outputs a signature σ.

$\mathsf{Verify}(\mathsf{vk}, m, \sigma) \to 0/1$. On input a verification key vk, a message $m \in \mathcal{M}$, and a signature σ, the verification algorithm outputs a bit to signify whether the signature is valid or not.

$\mathsf{Extract}(\mathsf{xk}, \mathsf{vk}, C) \to \tau/\perp$. On input the extraction key xk, a verification key vk, and a circuit $C\colon \mathcal{M} \to \mathcal{SIG}$, the extraction algorithm either outputs a mark $\tau \in \mathcal{T}$ or a special symbol $\perp$.

Remark 3.1 (Public vs. Private Marking/Extraction)**.** We say that a watermarkable signature scheme is "publicly markable" if $\mathsf{mk} = \mathsf{wpp}$ (i.e., the marking key is just the public parameters). Otherwise, we say that it is "privately markable". Similarly, say that a watermarkable signature scheme is "publicly extractable" if $\mathsf{xk} = \mathsf{wpp}$. If a watermarkable signature scheme is both publicly markable and extractable, then we say it is a *fully public* scheme.

Remark 3.2 (Extraction Semantics)**.** In our watermarking abstraction, the extraction algorithm $\mathsf{Extract}$ takes the verification key vk associated with the signing circuit C as an additional input. Since the verification key associated with a signature scheme is assumed to always be publicly available, this does not seem like a substantial limitation.

Correctness. A watermarkable signature scheme is said to be correct if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $(\mathsf{wpp}, \mathsf{mk}, \mathsf{xk}) \leftarrow \mathsf{WMSetup}(1^\lambda)$, $m \in \mathcal{M}$, $\tau \in \mathcal{T}$, the following holds

$$\Pr\left[\begin{array}{l} \mathsf{Verify}(\mathsf{vk}, m, \sigma) \neq 1 \ \vee \\ \mathsf{Extract}(\mathsf{xk}, \mathsf{vk}, C) \neq \tau \end{array} : \begin{array}{c} (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SigSetup}(1^\lambda, \mathsf{wpp}), \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m), \\ C \leftarrow \mathsf{Mark}(\mathsf{mk}, \mathsf{sk}, \tau) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Meaningfulness. Intuitively, the meaningfulness property captures the following two properties: (1) for any *fixed* circuit C, the probability that C is considered to be marked (with respect to honestly-generated watermarking and signature parameters) is negligible; and (2) the extraction algorithm outputs $\perp$ (with all but negligible probability) when given as input an unmarked signing circuit $\mathsf{Sign}(\mathsf{sk}, \cdot)$, where sk is sampled from $\mathsf{SigSetup}$. Formally, a watermarkable signature scheme satisfies the meaningfulness property if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $C\colon \mathcal{M} \to \mathcal{SIG}$,

$$\Pr\left[\mathsf{Extract}(\mathsf{xk}, \mathsf{vk}, C) \neq \perp : \begin{array}{c} (\mathsf{wpp}, \mathsf{mk}, \mathsf{xk}) \leftarrow \mathsf{WMSetup}(1^\lambda), \\ (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SigSetup}(1^\lambda, \mathsf{wpp}) \end{array}\right] \leq \mathsf{negl}(\lambda),$$

and for all $\lambda \in \mathbb{N}$, $(\mathsf{wpp}, \mathsf{mk}, \mathsf{xk}) \leftarrow \mathsf{WMSetup}(1^\lambda)$,

$$\Pr\left[\mathsf{Extract}(\mathsf{xk}, \mathsf{vk}, \mathsf{Sign}(\mathsf{sk}, \cdot)) \neq \perp : \ (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SigSetup}(1^\lambda, \mathsf{wpp})\right] \leq \mathsf{negl}(\lambda).$$

Functionality-preserving. The final property we define is functionality-preserving, which requires that a marked signing key still produces valid signatures (that verify with respect to the original verification key). However, we do *not* require that the marked signing key produces signatures from the same distribution as the unmarked key. A watermarkable signature scheme satisfies the functionality-preserving property if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $(\mathsf{wpp}, \mathsf{mk}, \mathsf{xk}) \leftarrow \mathsf{WMSetup}(1^\lambda)$, $m \in \mathcal{M}$, $\tau \in \mathcal{T}$, the following holds

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, m, C(m)) \neq 1 : \begin{array}{c} (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SigSetup}(1^\lambda, \mathsf{wpp}), \\ C \leftarrow \mathsf{Mark}(\mathsf{mk}, \mathsf{sk}, \tau) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Remark 3.3 (Exact Functionality-Preserving)**.** In the context of watermarking PRFs [CHN+16, BLW17, KW17, YAL+17, QWZ18, KW19], functionality-preserving states that the marked PRF key preserves the *exact* input/output behavior as that of the unmarked key on nearly all elements in the domain. While this notion makes sense for deterministic functions like PRFs, when considering possibly-randomized algorithms like the signing algorithm for a digital signature scheme, preserving the functionality is not tantamount to preserving the exact input/output behavior. This added flexibility enables new and simple constructions of watermarkable signatures and, at the same time, remains sufficient for realizing the existing applications of watermarking.

Remark 3.4 (Unique Signature Schemes and Functionality-Preserving)**.** We note that if we have a *unique* signature scheme (i.e., a signature scheme where for every message $m \in \mathcal{M}$, there is a unique signature σ that verifies with respect to the verification key), then our notion of functionality-preserving precisely requires that the marked circuit preserves the exact input/output behavior of the original signing circuit. We do not know how to watermark a unique signature scheme and leave this as an intriguing open problem.

Security. For security, we consider the standard notions of unforgeability and unremovability. However, our formulation of these notions are stronger than those of previous works. For unforgeability, we require that the signature scheme remains unforgeable even when the watermarking public parameters wpp are chosen in a malicious manner. This means that if the watermarkable signature scheme supports both public marking and public verification (as our construction in 3.3), then this stronger notion of unforgeability completely removes trust in any authority. This is an appealing property that is not satisfied by *any* existing watermarking scheme.[5]

For unremovability, we define a collusion resistant variant of the original definition where we say that the scheme is secure as long as an adversary that obtains polynomially-many marked keys (for a fixed initial signing key) cannot produce a new key that *preserves the signing functionality* and, yet, either does not contain the watermark or contains a totally different watermark. Our definition is a direct generalization of the unremovability notions considered in the setting of watermarking PRFs [CHN+16, BLW17], with the following differences:

- First, we use the same relaxation of functionality-preserving discussed above: namely, the adversary is allowed to construct any circuit that outputs valid signatures with noticeable probability (that verify under the signature scheme's verification key); it does not have to preserve the input/output behavior of the marked circuits it is given. This gives the adversary *more* power and is important for ruling out potential attacks on the scheme (see the discussion in Appendix B.1 and Remark B.3).

- Second, we allow the adversary to make multiple marking queries; namely, the adversary can see the same signing key marked with different and adversarially-chosen identities, and, even then, we require that the adversary cannot produce a new circuit whose watermark is not one of those corresponding to a signing circuit already given to the adversary. In particular, if an adversary sees a signing key marked with multiple identities $\{\tau_i\}_i$, it cannot create a new

[5] Recent constructions of watermarking for PRFs [QWZ18, KW19] have the drawback that even a semi-honest watermarking authority is able to break security of the *unmarked* keys in the system. Previous constructions from standard assumptions [KW17] become insecure if the watermarking authority generates the parameters maliciously. Our unforgeability notion ensures that a malicious party cannot generate the parameters in such a way as to embed a "trapdoor" into the signature scheme.

signing circuit where the watermark is not one of $\{\tau_i\}_i$. We discuss this notion of *collusion resistance* in greater detail in Remark 3.5.

- Finally, because of our relaxed notion of functionality-preserving, signatures output by the unmarked key can look different from signatures output by the marked key, so we additionally give the adversary access to the signing oracle with the unmarked key.[6]

Formally, we say that a watermarkable signature scheme $\Pi_{\mathsf{WM}} = (\mathsf{WMSetup}, \mathsf{SigSetup}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Mark}, \mathsf{Extract})$ is secure if it satisfies the following properties:

Unforgeability with malicious authority. For every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*) = 1 \wedge m \notin \mathcal{Q} : \begin{array}{c} \mathsf{wpp} \leftarrow \mathcal{A}(1^\lambda), \\ (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SigSetup}(1^\lambda, \mathsf{wpp}), \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{vk}) \end{array}\right] \leq \mathsf{negl}(\lambda),$$

where $\mathcal{Q} \subseteq \mathcal{M}$ is the set of messages $\mathcal{A}$ submitted to the Sign oracle.

ε-Unremovability. For every stateful *ε-unremovable admissible* PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\Pr\left[\mathsf{Extract}(\mathsf{xk}, \mathsf{vk}, C^*) \notin \mathcal{Q} : \begin{array}{c} (\mathsf{wpp}, \mathsf{mk}, \mathsf{xk}) \leftarrow \mathsf{WMSetup}(1^\lambda), \\ (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SigSetup}(1^\lambda, \mathsf{wpp}), \\ C^* \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot),\mathsf{Mark}(\mathsf{mk},\mathsf{sk},\cdot)}(1^\lambda, \mathsf{wpp}, \mathsf{vk}) \end{array}\right] \leq \mathsf{negl}(\lambda),$$

where $\mathcal{Q} \subseteq \mathcal{T}$ denotes the set of marks queried by $\mathcal{A}$ to the marking oracle,[7] and $\mathcal{A}$ is said to be ε-unremovable admissible if the circuit C^* it outputs is an ε-good signer for key vk. Here we say that C^* is an ε-good signer circuit for key vk if:

$$\Pr[\mathsf{Verify}(\mathsf{vk}, m, C^*(m)) = 1 : m \leftarrow \mathcal{M}] \geq \varepsilon.$$

Remark 3.5 (Bounded Collusion Resistance)**.** In our unremovability definition we allow the adversary to make an a priori unbounded polynomial number of marking queries. One could also consider bounded collusion variants, where we say a watermarkable signature scheme is q-bounded collusion secure if the unremovability property holds only against admissible adversaries that make at most q marking queries. Existing schemes for cryptographic functionalities from standard assumptions are only 1-bounded collusion secure [KW17, QWZ18, KW19].

[6] We would like to point out that disallowing signing queries answered using the unmarked key does not lead to a weaker definition. This is due to the fact there is a simple transformation that could be used to handle such queries. Briefly, the idea is to first watermark an unmarked signing key using a default (fixed) mark, and then use the marked key to sign the message. This way, a reduction algorithm could query the watermarking challenger (corresponding to the underlying scheme that does not permit signing queries under unmarked key) to obtain a marked key for the default mark, and then use that particular marked key to answer the adversary's signing queries. However, for ease of exposition, in our definition, we allow the adversary to make signing queries under the unmarked key.

[7] Here we only allow the adversary to make marking queries. A stronger definition would allow the adversary to make extraction queries as well. However, note that if the scheme has public marking and extraction procedures (which is what we build here), then all such oracle queries are already redundant.

Remark 3.6 (Small Unremovability Thresholds)**.** Previously, Cohen et al. [CHN+16] showed that message-embedding watermarking schemes satisfying ε-unremovability are possible only when $\varepsilon \geq 1/2 + 1/\mathsf{poly}(\lambda)$. This lower bound does not apply to our notion of ε-unremovability. In fact, our constructions satisfy ε-unremovability for *any* inverse polynomial $\varepsilon = 1/\mathsf{poly}(\lambda)$. The reason is that our mark-extraction algorithm takes in the verification key as input, while the Cohen et al. definition does not (i.e., their extraction algorithm only takes the circuit as input).

To provide some additional detail, we first recall the attack from Cohen et al. when $\varepsilon = 1/2$. Let C be the challenge circuit marked with a message m in the unremovability security game, and let C' be an arbitrary circuit (for a different function) marked with a message $m' \neq m$. In both the secret and public marking setting, the adversary can generate C' by either using the marking oracle (secret-marking setting) or using the public marking algorithm (public-marking setting). To carry out the attack, the adversary constructs a challenge circuit C^* that agrees with C on half of the points (chosen randomly) and agrees with C' on the other half. By symmetry, the extraction algorithm on C^* outputs m and m' with equal probability, where the probability is taken over the coins of the $\mathsf{Extract}$ algorithm and the adversary's randomness used to determine m, m', and C^*. This means that the probability that the extraction algorithm outputs m is at most $1/2$, and so the adversary succeeds with probability at least $1/2$.

The above attack critically relies on the fact that the adversary is able to obtain a marked circuit C' where the extraction algorithm on C' outputs $m' \neq m$. In order to mount the same attack in our setting, the adversary needs to be able to obtain a circuit C' such that $\mathsf{Extract}(\mathsf{xk}, \mathsf{vk}, C') = m'$, where vk is the verification key chosen by the challenger. In the security game, there is no mechanism for the adversary to obtain a marked circuit with respect to vk other than by making a marking query on m'. If the adversary makes a marking query on m', then as long as the extraction algorithm recovers *either* m or m', the adversary does not break unremovability. Observe that if, on the contrary, vk is not provided as input to $\mathsf{Extract}$, then the adversary can easily construct a circuit with an embedded mark m' (by marking an arbitrary key of its choosing) and mount the Cohen et al. attack. This distinction, where the extraction algorithm is defined with respect to a *specific* verification key, enables us to circumvent the lower bound for ε-unremovability when $\varepsilon \leq 1/2$.

3.2 Building Blocks: Constrained Signatures

In this section, we review the main building block of constrained signatures that we use to construct our scheme.

Constrained signatures. A constrained signature scheme [BF14] with message space $\mathcal{M}$ and constraint family $\mathcal{F} \subseteq \mathsf{Funs}[\mathcal{M}, \{0,1\}]$ is a tuple of algorithm $\Pi_{\mathsf{CSig}} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Constrain}, \mathsf{ConstrainSign})$ with the following syntax:

$\mathsf{Setup}(1^\lambda) \to (\mathsf{vk}, \mathsf{msk})$. On input the security parameter λ, the setup algorithm outputs the verification key and the master secret key msk.

$\mathsf{Sign}(\mathsf{msk}, m) \to \sigma$. On input the master signing key msk and a message $m \in \mathcal{M}$, the signing algorithm outputs a signature σ.

$\mathsf{Verify}(\mathsf{vk}, m, \sigma) \to b$. On input the verification key vk, a message $m \in \mathcal{M}$, and a signature σ, the verification algorithm outputs a bit $b \in \{0,1\}$.

$\mathsf{Constrain}(\mathsf{msk}, f) \to \mathsf{sk}_f$. On input the master signing key msk and a function $f \in \mathcal{F}$, the constrain algorithm outputs a constrained key sk_f.

$\mathsf{ConstrainSign}(\mathsf{sk}_f, m) \to \sigma$. On input a constrained key sk_f and a message $m \in \mathcal{M}$, the signing algorithm outputs a signature σ.

Correctness. A constrained signature scheme is correct if for all messages $m \in \mathcal{M}$ and key pair $(\mathsf{vk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$,

$$\Pr[\mathsf{Verify}(\mathsf{vk}, m, \mathsf{Sign}(\mathsf{msk}, m)) = 1] = 1.$$

In addition, for all constraints $f \in \mathcal{F}$ where $f(m) = 1$, if we compute $\mathsf{sk}_f \leftarrow \mathsf{Constrain}(\mathsf{msk}, f)$,

$$\Pr[\mathsf{Verify}(\mathsf{vk}, m, \mathsf{ConstrainSign}(\mathsf{sk}_f, m)) = 1] = 1.$$

Definition 3.7 (Constrained Unforgeability)**.** A constrained signature scheme is secure if for every stateful admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*) = 1 : \begin{array}{c}(\mathsf{vk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{msk},\cdot),\mathsf{Constrain}(\mathsf{msk},\cdot)}(1^\lambda, \mathsf{vk})\end{array}\right] \leq \mathsf{negl}(\lambda),$$

where $\mathcal{A}$ is an admissible adversary if (1) it does not make a signing query on message m^*; and (2) it does not make a constrained key query for any function $f \in \mathcal{F}$ such that $f(m^*) = 1$.

3.3 Watermarking Signatures from Constrained Signatures

In this section, we show how to construct a fully collusion resistant watermarking scheme for digital signatures from prefix-constrained signatures.

Construction 3.8 (Watermarkable Signatures from Prefix-Constrained Signatures)**.** Fix a message space $\mathcal{M}$ and a mark space $\mathcal{T}$. Let $\varepsilon = 1/\mathsf{poly}(\lambda)$ be an unremovability parameter. We define the following primitives:

- Let $\mathcal{T}' = \mathcal{T} \cup \{\perp\}$. For a mark $\tau^* \in \mathcal{T}$, let $f_{\tau^*} \colon \mathcal{T}' \times \mathcal{M} \to \{0,1\}$ be the function where $f_{\tau^*}(\tau, m) = 1$ if $\tau = \tau^*$ and 0 otherwise.
- Let $\Pi_{\mathsf{CSig}} = (\mathsf{CSig.Setup}, \mathsf{CSig.Sign}, \mathsf{CSig.Verify}, \mathsf{CSig.Constrain}, \mathsf{CSig.ConstrainSign})$ be a constrained signature scheme with message space $\mathcal{M}' = \mathcal{T}' \times \mathcal{M}$ and function class $\mathcal{F} = \{\tau^* \in \mathcal{T} \colon f_{\tau^*}\}$. Let $\mathcal{SIG}'$ be the signature space of Π_{CSig}.

We construct a watermarkable signature scheme $\Pi_{\mathsf{WM}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Mark}, \mathsf{Extract})$ with signature space $\mathcal{SIG} = \mathcal{T}' \times \mathcal{SIG}'$ as follows:

$\mathsf{WMSetup}(1^\lambda) \to (\mathsf{wpp}, \mathsf{mk}, \mathsf{xk})$**.** On input the security parameter λ, the setup algorithm outputs $\mathsf{wpp}, \mathsf{mk}, \mathsf{xk} = \perp$. Namely, the scheme does not require any watermarking parameters.

$\mathsf{SigSetup}(1^\lambda, \mathsf{wpp}) \to (\mathsf{vk}, \mathsf{sk})$. On input the security parameter λ and public parameters $\mathsf{wpp} = \perp$, the key-generation algorithm outputs a signing/verification key-pair $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{CSig.Setup}(1^\lambda)$.

$\mathsf{Sign}(\mathsf{sk}, m) \to \sigma$**.** On input a signing key sk, and a message $m \in \mathcal{M}$, the signing algorithm signs $\sigma' \leftarrow \mathsf{CSig.Sign}(\mathsf{sk}, (\perp, m))$, and outputs the signature $\sigma = (\perp, \sigma')$.

$\mathsf{Verify}(\mathsf{vk}, m, \sigma) \to b$**.** On input a verification key vk, a message $m \in \mathcal{M}$, and a signature $\sigma = (\tau', \sigma')$, the verification algorithm outputs $b \leftarrow \mathsf{CSig.Verify}(\mathsf{vk}, (\tau', m), \sigma')$.

$\mathsf{Mark}(\mathsf{mk}, \mathsf{sk}, \tau) \to C$**.** On input a marking key $\mathsf{mk} = \bot$, a signing key sk, and a mark $\tau \in \mathcal{T}$, the marking algorithm computes $\mathsf{sk}_\tau \leftarrow \mathsf{CSig.Constrain}(\mathsf{sk}, f_\tau)$ and outputs a circuit $C_\tau \colon \mathcal{M} \to \mathcal{SIG}$ where $C_\tau(\cdot) := (\tau, \mathsf{CSig.ConstrainSign}(\mathsf{sk}_\tau, (\tau, \cdot)))$.

$\mathsf{Extract}(\mathsf{xk}, \mathsf{vk}, C) \to \tau/\bot$**.** On input an extraction key $\mathsf{xk} = \bot$, a verification key vk, and a circuit $C \colon \mathcal{M} \to \mathcal{SIG}$, the extraction algorithm performs the following procedure $T = \lambda/\varepsilon = \mathsf{poly}(\lambda)$ times:

- For $i \in [T]$, sample $m_i \leftarrow \mathcal{M}$ and compute $(\tau_i', \sigma_i') \leftarrow C(m)$. If $\mathsf{CSig.Verify}(\mathsf{vk}, (\tau_i', m_i), \sigma_i') = 1$, abort and output τ_i'.

If the above procedure does not abort with some output τ, then output $\bot$.

Correctness and security analysis. We now state our correctness and security theorems, but defer their formal analysis to Section 3.3.1.

Theorem 3.9 (Correctness)**.** *Suppose Π_{CSig} is correct and satisfies constrained unforgeability. Then, the watermarkable signature scheme Π_{WM} from Construction 3.8 satisfies correctness, meaningfulness, and functionality-preserving.*

Theorem 3.10 (Signature Unforgeability)**.** *If Π_{CSig} satisfies constrained unforgeability, then the watermarkable signature scheme Π_{WM} from Construction 3.8 satisfies signature unforgeability in the presence of a malicious watermarking authority.*

Theorem 3.11 (Unremovability)**.** *Take any $\varepsilon = 1/\mathsf{poly}(\lambda)$. If $1/|\mathcal{M}| = \mathsf{negl}(\lambda)$ and Π_{CSig} satisfies constrained unforgeability, then the watermarkable signature scheme Π_{WM} from Construction 3.8 is ε-unremovable.*

3.3.1 Analysis of Watermarkable Signatures (Construction 3.8)

In this section, we give the correctness and security analysis of the watermarkable digital signature scheme from Section 3.3 (Construction 3.8).

Proof of Theorem 3.9 (Correctness). We check each of the properties below:

- **Correctness:** Take any message $m \in \mathcal{M}$ and tag $\tau \in \mathcal{T}$ and let $(\mathsf{wpp}, \mathsf{mk}, \mathsf{xk}) \leftarrow \mathsf{WMSetup}(1^\lambda)$. Take $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SigSetup}(1^\lambda, \mathsf{wpp})$, $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$, and $C \leftarrow \mathsf{Mark}(\mathsf{mk}, \mathsf{sk}, \tau)$. We show the two requirements separately.
  - **Correctness of decryption:** By construction, σ is a signature on the pair $(\bot, m)$ under sk, so by correctness of Π_{CSig}, $\Pr[\mathsf{CSig.Verify}(\mathsf{vk}, (\bot, m), \sigma) = 1] = 1$, and correspondingly, $\Pr[\mathsf{Verify}(\mathsf{vk}, m, \sigma) = 1] = 1$.
  - **Correctness of extraction:** By construction, $C(\cdot) := (\tau, \mathsf{CSig.ConstrainSign}(\mathsf{sk}_\tau, (\tau, \cdot)))$ where $\mathsf{sk}_\tau \leftarrow \mathsf{CSig.Constrain}(\mathsf{sk}, f_\tau)$. By definition, $f_\tau(\tau, m) = 1$ for all $m \in \mathcal{M}$, so by correctness of Π_{CSig}, for all $m \in \mathcal{M}$, if we define $(\tau', \sigma') \leftarrow C(m)$, we have that $\tau' = \tau$, and
  $$\Pr[\mathsf{CSig.Verify}(\mathsf{vk}, (\tau', m), \sigma') = 1] = 1.$$
  Correspondingly, $\Pr[\mathsf{Extract}(\mathsf{xk}, \mathsf{vk}, C) = \tau] = 1$.

- **Meaningfulness:** We show the two requirements separately:
  - *Most circuits unmarked.* Take any circuit $C\colon \mathcal{M} \to \mathcal{SIG}$. Let $(\mathsf{wpp}, \mathsf{mk}, \mathsf{xk}) \leftarrow \mathsf{WMSetup}(1^\lambda)$ and $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SigSetup}(1^\lambda, \mathsf{wpp})$. By construction, $(\mathsf{vk}, \mathsf{sk})$ are sampled from $\mathsf{CSig.Setup}(1^\lambda)$, independently of the circuit C. By constrained unforgeability of Π_{CSig}, if we sample $m \leftarrow \mathcal{M}$, and compute $(\tau, \sigma) \leftarrow C(m)$

    $$\Pr[\mathsf{CSig.Verify}(\mathsf{vk}, (\tau, m), \sigma) = 1] = \mathsf{negl}(\lambda).$$

    Otherwise, an adversary with a hardwired circuit C can be used to forge signatures with respect to a freshly-sampled verification key vk, which breaks unforgeability of Π_{CSig}. Since $T = \mathsf{poly}(\lambda)$ in the $\mathsf{Extract}$ algorithm, the claim now follows by a union bound.
  - *Extraction fails on unmarked keys.* By construction, $\mathsf{Sign}(\mathsf{sk}, \cdot)$ always outputs tuples of the form $(\bot, \sigma)$, and by the construction of $\mathsf{Extract}$, the output will be $\bot$ with probability 1.
- **Functionality-preserving:** Follows by the same argument as that used to show correctness of extraction.

Proof of Theorem 3.10 (Signature Unforgeability). Unforgeability follows directly from constrained unforgeability of Π_{CSig}. Namely, suppose there is an adversary $\mathcal{A}$ that breaks signature unforgeability of Π_{WM} in the presence of a malicious watermarking authority. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ for the constrained unforgeability game for Π_{CSig}:

1. Algorithm $\mathcal{B}$ receives a verification key vk from the challenger for Π_{CSig} and gives vk to $\mathcal{A}$. Note that since there are no public parameters in Construction 3.8, the adversary $\mathcal{A}$ does not get to choose the public parameters.
2. When $\mathcal{A}$ makes a signing query on a message $m \in \mathcal{M}$, algorithm $\mathcal{B}$ forwards the message $(\bot, m)$ to its challenger and receives back a signature σ which it forwards to $\mathcal{A}$.
3. At the end, $\mathcal{A}$ outputs a message $m^* \in \mathcal{M}$ and a signature σ^*. Algorithm $\mathcal{B}$ outputs $(\bot, m^*)$ as its message and σ^* as its signature.

By construction, algorithm $\mathcal{B}$ perfectly simulates the constrained unforgeability game for $\mathcal{A}$, and so with non-negligible probability, $\mathcal{A}$ will output a valid signature σ^* on a message m^* (that did not appear in any signing query). Also by construction, σ^* is a valid signature on the tuple $(\bot, m^*)$, and since $\mathcal{B}$ does not make any constrain queries and only makes signing queries on $(\bot, m)$ where $m \neq m^*$, $\mathcal{B}$ succeeds in constructing a forgery with the same advantage as $\mathcal{A}$. □

Proof of Theorem 3.11 (Unremovability). Suppose there exists an efficient and admissible adversary $\mathcal{A}$ that breaks ε-unremovability of Π_{WM} with probability $\varepsilon = 1/\mathsf{poly}(\lambda)$. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that breaks security of the constrained signature scheme Π_{CSig}:

1. At the beginning of the constrained signature security game, algorithm $\mathcal{B}$ receives a verification key vk from the constrained signature challenger. It gives vk to $\mathcal{A}$ (the other components wpp, mk, xk are all empty).

2. Whenever $\mathcal{A}$ makes a marking query on a mark $\tau \in \mathcal{T}$, algorithm $\mathcal{B}$ issues a constrain query on the function f_τ and receives a constrained key sk_τ. Algorithm $\mathcal{B}$ defines the circuit $C_\tau(\cdot) := (\tau, \mathsf{CSig.ConstrainSign}(\mathsf{sk}_\tau, (\tau, \cdot)))$, and gives C_τ to $\mathcal{A}$. Let $\mathcal{Q}_{\mathsf{mark}} \subseteq \mathcal{T}$ be the set of marks $\mathcal{A}$ submitted to the marking oracle.

3. Whenever $\mathcal{A}$ makes a signing query on a message $m \in \mathcal{M}$, algorithm $\mathcal{B}$ makes a signing query on $(\bot, m)$ and receives a signature σ'. It gives $\sigma = (\bot, \sigma')$ to $\mathcal{A}$. Let $\mathcal{Q}_{\mathsf{sign}} \subseteq \mathcal{M}$ be the set of messages $\mathcal{A}$ submitted to the signing oracle.

4. At the end of the game, algorithm $\mathcal{A}$ outputs a circuit $C^*\colon \mathcal{M} \to \mathcal{SIG}$. Algorithm $\mathcal{B}$ repeats the following procedure $\lambda/\varepsilon = \mathsf{poly}(\lambda)$ times:

   - Sample a random message $m^* \leftarrow \mathcal{M}$ and compute $(\tau^*, \sigma^*) \leftarrow C^*(m^*)$. If $\tau^* \notin \mathcal{Q}_{\mathsf{mark}}$ and $\mathsf{CSig.Verify}(\mathsf{vk}, (\tau^*, m^*), \sigma^*) = 1$, then algorithm $\mathcal{B}$ completes the simulation and outputs $(\tau^*, m^*), \sigma^*$ as its forgery.

   If $\mathcal{B}$ did not abort, then it outputs $\bot$.

By construction, $\mathcal{B}$ perfectly simulates the ε-unremovability game for $\mathcal{A}$. This means with non-negligible probability ε, $\mathsf{Extract}(\mathsf{xk}, \mathsf{vk}, C^*)$ outputs some $\hat{\tau} \notin \mathcal{Q}_{\mathsf{mark}}$. We argue that with probability at least $\varepsilon - \mathsf{negl}(\lambda)$, algorithm $\mathcal{B}$ breaks constrained unforgeability of Π_{CSig}. Consider the behavior of $\mathsf{Extract}(\mathsf{xk}, \mathsf{vk}, C^*)$. Let m_i be the i^{th} message sampled by the $\mathsf{Extract}$ algorithm for $i \in [T]$. Similarly, let $\sigma_i = (\tau_i', \sigma_i') \leftarrow C(m_i)$. We consider two possibilities:

- Suppose $\hat{\tau} = \tau_i'$ for some $i \in [T]$. This means that $\mathsf{CSig.Verify}(\mathsf{vk}, (\tau_i', m_i), \sigma_i') = 1$. Moreover, in this case, since $\tau_i' = \hat{\tau} \notin \mathcal{Q}_{\mathsf{mark}}$, algorithm $\mathcal{B}$ will set $(\tau^*, m^*) = (\tau_i', m_i)$ and $\sigma^* = \sigma_i'$.

- Suppose $\hat{\tau} = \bot$ and that $\mathsf{CSig.Verify}(\mathsf{vk}, (\tau_i', m_i), \sigma_i') = 0$ for all $i \in [T]$. In this case, $\mathcal{B}$ fails to produce a forgery and outputs $\bot$. First, if $\mathsf{CSig.Verify}(\mathsf{vk}, (\tau_i', m_i), \sigma_i') = 0$, then $\mathsf{Verify}(\mathsf{vk}, m_i, \sigma_i) = 0$. Moreover, since $\mathcal{A}$ is ε-admissible, we conclude that for $m_i \leftarrow \mathcal{M}$,

$$\Pr[\mathsf{CSig.Verify}(\mathsf{vk}, (\tau_i', m_i), \sigma_i') = 0] \leq \Pr[\mathsf{Verify}(\mathsf{vk}, m_i, \sigma_i) = 0] \leq 1 - \varepsilon.$$

  Thus,

$$\Pr[\forall i \in [T] : \mathsf{CSig.Verify}(\mathsf{vk}, (\tau_i', m_i), \sigma_i') = 0] \leq (1-\varepsilon)^T = (1-\varepsilon)^{\lambda/\varepsilon} \leq e^{-\lambda} = \mathsf{negl}(\lambda).$$

Thus, if $\mathsf{Extract}(\mathsf{xk}, \mathsf{vk}, C^*)$ outputs $\hat{\tau} \notin \mathcal{Q}_{\mathsf{mark}}$ with probability ε, algorithm $\mathcal{B}$ outputs a valid signature σ^* on the message (τ^*, m^*) where $\tau^* = \hat{\tau}$. We now check the remaining properties:

- Since $\mathcal{B}$ samples the messages $m^* \leftarrow \mathcal{M}$, $\Pr[m^* \in \mathcal{Q}_{\mathsf{sign}}] \leq \mathcal{Q}_{\mathsf{sign}}/\left|\mathcal{M}\right| = \mathsf{negl}(\lambda)$. Thus, with overwhelming probability, algorithm $\mathcal{B}$ never made a signing query on the message (τ^*, m^*).

- By assumption, $\tau^* = \hat{\tau} \notin \mathcal{Q}_{\mathsf{mark}}$ and correspondingly, $f_\tau(\tau^*, m^*) = 0$ for all functions f_τ that $\mathcal{B}$ submitted to the constrain oracle.

We conclude that $\mathcal{B}$ succeeds in breaking constrained unforgeability of Π_{CSig} with probability $\varepsilon - \mathsf{negl}(\lambda)$. □

3.4 Instantiations and Extensions

As noted by Bellare and Fuchsbauer [BF14], fully collusion resistant constrained signatures (for arbitrary circuit constraints) satisfying unforgeability follow immediately from any standard signature scheme (which can in turn be based on one-way functions [Gol04]). We briefly recall the "certificate-based" construction here. The public parameters for the constrained signature scheme is a verification key vk for a standard signature scheme, and the master secret key is the associated signing key sk. To issue a constrained key for a function f, the authority generates a fresh pair of signing and verification keys $(\mathsf{vk}', \mathsf{sk}')$, and constructs a signature ("certificate") σ on (vk', f) with the master signing key sk. The constrained key is the tuple $(\mathsf{vk}', \mathsf{sk}', f, \sigma)$. A signature on m using the constrained key consists of a signature σ' on m using sk' together with the tuple $(\mathsf{vk}', f, \sigma)$. To verify, one checks that σ is a valid signature on (vk', f) with respect to the master verification key vk, that $f(m) = 1$, and that σ' is a valid signature on m with respect to vk'. From this construction, we obtain the following corollary:

Corollary 3.12 (Watermarkable Signatures from One-Way Functions)**.** *Take any $\varepsilon = 1/\mathsf{poly}(\lambda)$ and mark space $\mathcal{T} = \{0,1\}^\ell$, where $\ell = \mathsf{poly}(\lambda)$. Assuming one-way functions exist, there exists a fully collusion resistant watermarkable family of signatures with mark space $\mathcal{T}$ that satisfies ε-unremovability and where the underlying signature scheme is unforgeable even against a malicious watermarking authority.*

Watermarking unforgeability in the secret-key setting. A dual property that is often studied in the context of watermarking is unforgeability [CHN+16, BLW17, KW17, KW19], which intuitively says that no efficient adversary should be able to construct a new circuit (i.e., a circuit that is sufficiently different from marked circuits it received from the watermarking authority) and yet, is still consider to be marked. Like other watermarking schemes that support *public* marking [QWZ18, KW19], unforgeability is an incompatible property since the adversary can simply run the marking algorithm on its own to produce new marked circuits. However, we can consider a variant of our scheme in the secret marking setting.

It is straightforward to modify our construction to obtain a watermarking scheme satisfying mark unforgeability in the secret-key setting. The (secret) marking key would consist of a signing key sk for a standard signature scheme and its associated verification key vk would be included as part of the watermarking public parameters wpp. To embed a mark $\tau \in \mathcal{T}$ in a signing key sk' (with associated verification key vk'), the marking authority would instead embed the mark (τ, σ_τ) where σ_τ is a signature under the authority's signing key on the pair (vk', τ). Extraction of the watermark operates exactly as before, except the $\mathsf{Extract}$ algorithm additionally checks that σ_τ is a valid signature on (vk', τ). To forge a new program that is considered to be marked (and which is noticeably different in behavior from one of the marked programs that the adversary already received), the adversary needs to produce a signature on a new (vk', τ) pair under the watermarking authority's secret signing key.

Remark 3.13 (Watermarking Constrained Signature Schemes)**.** We note that the watermarkable signature scheme in Construction 3.8 generalizes naturally to yield a watermarking scheme for *constrained signatures*, where we can watermark not only the master signing key, but also a constrained signing key. To do so, we require that the underlying constrained signature scheme Π_{CSig} in Construction 3.8 support *key delegation*. Very briefly, a constrained signature scheme supports key delegation if a constrained key for a policy $f\colon \mathcal{M} \to \{0,1\}$ can be further constrained to obtain

a key for a policy $g \wedge f$, where $(g \wedge f)(m) = 1$ if and only if $f(m) = 1 = g(m)$. Marking the master secret key can be handled in the same manner as in Construction 3.8, while marking a constrained key can be handled using key delegation and giving out a delegated prefix-constrained key (where the prefix is again the mark). The certificate-based constrained signature construction of [BF14] naturally supports key delegation, and thus, can be used to obtain a watermarkable constrained signature scheme.

4 Watermarkable Encryption Schemes

In this section, we show how to watermark a public-key attribute-based encryption scheme. As in Section 3, we start by introducing a stronger definition for a watermarkable public-key encryption scheme. Then, in Section 4.2, we recall the notions of mixed functional encryption (mixed FE) [GKW18] and delegatable attribute-based encryption [GPSW06], which we use to construct our fully collusion resistant watermarkable encryption scheme. We then provide our main construction in Section 4.3. In Appendix C, we provide an alternate construction that relies on hierarchical FE [ABG+13, BCG+17] and achieves *bounded collusion* in the fully public setting.

4.1 Defining Watermarkable Encryption

As in Section 3, our definition of a watermarkable public-key encryption scheme is an extension of the framework introduced by Cohen et al. [CHN+16]. We refer to Appendix B.2 for a more detailed comparison of our definitions with those of Cohen et al. In addition, we extend the notion of watermarking to the decryption circuits of public-key predicate encryption schemes [BW07, SBC+07, KSW08]. Our definition captures watermarking for simpler classes of public-key primitives like public-key encryption, identity-based encryption, and attribute-based encryption as a special case. Later in this section, we show how to watermark a public-key attribute-based encryption scheme (with full collusion resistance), and in Appendix C, we show how to watermark a predicate encryption scheme (but only providing bounded collusion resistance in exchange).

Syntax. A watermarkable public-key predicate encryption scheme with message space $\mathcal{M}$, attribute space $\mathcal{X}$, function space $\mathcal{F} \subseteq \mathsf{Funs}[\mathcal{X}, \{0,1\}]$, and mark space $\mathcal{T}$ is a tuple of algorithms $(\mathsf{WMSetup}, \mathsf{PESetup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Mark}, \mathsf{Extract})$ with the following syntax:

$\mathsf{WMSetup}(1^\lambda) \to (\mathsf{wpp}, \mathsf{mk}, \mathsf{xk})$. On input the security parameter λ, the watermarking setup algorithm outputs a set of public parameters wpp, a marking key mk, and an extraction key xk.

$\mathsf{PESetup}(1^\lambda, \mathsf{wpp}) \to (\mathsf{mpk}, \mathsf{msk})$. On input the security parameter λ and watermarking public parameters wpp, the key-generation algorithm outputs an (unmarked) predicate encryption master public-secret key pair $(\mathsf{mpk}, \mathsf{msk})$.

Here the message space $\mathcal{M}$, attribute space $\mathcal{X}$, master public key space $\mathcal{MPK}$, secret key space $\mathcal{SK}$, and ciphertext space $\mathcal{CT}$ are parameterized by the watermarking public parameters wpp.

$\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$. On input the master secret key msk, a predicate $f \in \mathcal{F}$, the key-generation algorithm outputs a predicate key sk_f.

$\mathsf{Enc}(\mathsf{mpk}, x, m) \to \mathsf{ct}$. On input a master public key mpk, an attribute $x \in \mathcal{X}$, a message $m \in \mathcal{M}$, the encryption algorithm outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) \to m/\perp$. On input a secret key sk_f and a ciphertext ct, the decryption algorithm outputs either a message $m \in \mathcal{M}$ or a special symbol $\perp$.

$\mathsf{Mark}(\mathsf{mk}, \mathsf{sk}_f, \tau) \to C$. On input a marking key mk, a secret key $\mathsf{sk}_f \in \mathcal{SK}$, and a mark $\tau \in \mathcal{T}$, the marking algorithm outputs a marked circuit $C \colon \mathcal{CT} \to \mathcal{M} \cup \{\perp\}$.

$\mathsf{Extract}(\mathsf{xk}, \mathsf{mpk}, x, (m_0, m_1), C) \to \tau/\perp$. On input the extraction key xk, a master public key mpk, an attribute x, a pair of messages (m_0, m_1), and a circuit $C \colon \mathcal{CT} \to \mathcal{M} \cup \{\perp\}$, the extraction algorithm either outputs a mark $\tau \in \mathcal{T}$ or a special symbol $\perp$.

Remark 4.1 (Public vs. Private Marking/Extraction)**.** As in the case of a watermarkable signature scheme, we say that a watermarkable PE scheme is "publicly markable" if $\mathsf{mk} = \mathsf{wpp}$ (i.e., the marking key is simply the public watermarking parameters). Otherwise, we say that it is "privately markable". Similarly, say that a watermarkable PE scheme is "publicly extractable" if $\mathsf{xk} = \mathsf{wpp}$. If a watermarkable PE scheme is both publicly markable and extractable, then we say it is a *fully public* scheme.

Remark 4.2 (Extraction Semantics)**.** Note that in our watermarking abstraction, the extraction algorithm $\mathsf{Extract}$ takes the master public key mpk, an attribute x, and a pair of messages (m_0, m_1) as an additional inputs. Taking the public key mpk as an additional input does not seem like a substantial limitation since mpk is always public. The intuition behind giving the attribute x and message pair (m_0, m_1) as inputs is to make the extraction procedure *more general* while disallowing a much wider range of watermarking attack strategies. Looking ahead, the intuition is that for security of watermarkable PE schemes, we would require that an adversary cannot even create an unmarked (or differently-marked) decryption circuit which can only decrypt a very small fraction of the ciphertext space, notably, ciphertexts encrypting either m_0 or m_1 under attribute x. Our notion of such strong extraction is inspired by the recent developments in the area of traitor tracing [GKRW18, GKW18], and it provides much more functionality than previously-studied extraction notions in the setting of watermarkable public-key encryption.

Correctness. A watermarkable predicate encryption scheme is said to be correct if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $(\mathsf{wpp}, \mathsf{mk}, \mathsf{xk}) \leftarrow \mathsf{WMSetup}(1^\lambda)$, $m^*, m_0 \neq m_1 \in \mathcal{M}$, $\tau \in \mathcal{T}$, $x \in \mathcal{X}$, $f \in \mathcal{F}$, the following holds

$$\Pr\left[\begin{array}{c} (f(x) = 0 \wedge \mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) \neq \perp) \vee \\ (f(x) = 1 \wedge \mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) \neq m^*) \vee \\ \left(\begin{array}{c} f(x) = 1 \ \wedge \\ \mathsf{Extract}(\mathsf{xk}, \mathsf{mpk}, x, (m_0, m_1), C) \neq \tau \end{array}\right) \end{array} : \begin{array}{c} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{PESetup}(1^\lambda, \mathsf{wpp}), \\ \mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f), \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, m^*), \\ C \leftarrow \mathsf{Mark}(\mathsf{mk}, \mathsf{sk}_f, \tau) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Next, we define the notion of meaningfulness and functionality-preserving similar to that in the case of watermarkable signatures. (Recall, we need these extra properties due to our independent setup abstraction.) Here for functionality-preserving, we again consider a relaxed notion, where we only require that a marked decryption key can decrypt honestly encrypted ciphertexts. That is, we allow the output of decryption to differ on ciphertexts that do not lie in the set of valid ciphertexts.

Meaningfulness. A watermarkable predicate encryption scheme satisfies the meaningfulness property if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $C \colon \mathcal{CT} \to \mathcal{M} \cup \{\perp\}$,

$x \in \mathcal{X}$, $m_0, m_1 \in \mathcal{M}$,

$$\Pr\left[\mathsf{Extract}(\mathsf{xk}, \mathsf{mpk}, x, (m_0, m_1), C) \neq \bot : \begin{array}{c}(\mathsf{wpp}, \mathsf{mk}, \mathsf{xk}) \leftarrow \mathsf{Setup}(1^\lambda), \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{PESetup}(1^\lambda, \mathsf{wpp})\end{array}\right] \leq \mathsf{negl}(\lambda),$$

and for all $\lambda \in \mathbb{N}$, $(\mathsf{wpp}, \mathsf{mk}, \mathsf{xk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $x \in \mathcal{X}$, $m_0, m_1 \in \mathcal{M}$, $f \in \mathcal{F}$,

$$\Pr\left[\begin{array}{c}f(x) = 1 \wedge \\ \mathsf{Extract}(\mathsf{xk}, \mathsf{mpk}, x, (m_0, m_1), \mathsf{Dec}(\mathsf{sk}_f, \cdot)) \neq \bot\end{array} : \begin{array}{c}(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{PESetup}(1^\lambda, \mathsf{wpp}), \\ \mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)\end{array}\right] \leq \mathsf{negl}(\lambda).$$

Functionality-preserving. A watermarkable predicate encryption scheme is functionality-preserving if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $(\mathsf{wpp}, \mathsf{mk}, \mathsf{xk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $x \in \mathcal{X}$, $m \in \mathcal{M}$, $f \in \mathcal{F}$, $\tau \in \mathcal{T}$, the following holds

$$\Pr\left[f(x) = 1 \wedge C(\mathsf{ct}) \neq m : \begin{array}{c}(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{PESetup}(1^\lambda, \mathsf{wpp}), \\ \mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f), \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, m), \\ C \leftarrow \mathsf{Mark}(\mathsf{mk}, \mathsf{sk}_f, \tau)\end{array}\right] \leq \mathsf{negl}(\lambda).$$

Security. For security, we consider the notions of semantic security and unremovability. As in the case of signatures, we strengthen the traditional variants of these notions that were considered in previous works. For semantic security, we strengthen the property by requiring that IND-CPA (and 2-sided attribute hiding) must hold even if the watermarking parameters wpp are chosen in a malicious manner. For unremovability, we define a collusion resistant variant as in the case of watermarkable signatures, where we say that even given polynomially-many marked versions of a fixed secret key sk_f, the adversary cannot generate a circuit C^* such that it contains a different (or no) watermark, and C^* preserves functionality as per our above definition. We give the formal requirements below:

Encryption security with malicious authority. For every stateful PPT attacker $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk}, \cdot)}(\mathsf{ct}) = b : \begin{array}{c}\mathsf{wpp} \leftarrow \mathcal{A}(1^\lambda), (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{PESetup}(1^\lambda, \mathsf{wpp}), \\ ((m_0, m_1), (x_0, x_1)) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk}, \cdot)}(\mathsf{mpk}), \\ b \leftarrow \{0, 1\},\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, x_b, m_b)\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where every predicate query f made by adversary $\mathcal{A}$ must satisfy one of the following conditions:

- $f(x_0) = f(x_1) = 0$ for all queried predicates f; or
- $m_0 = m_1$ and $f(x_0) = f(x_1) = 1$ for all queried predicates f.[8]

ε-Unremovability. For every stateful *ε-unremovable admissible* PPT attacker $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\mathsf{Extract}(\mathsf{xk}, \mathsf{mpk}, x, (m_0, m_1), C^*) \notin \mathcal{Q} : \begin{array}{c}(\mathsf{wpp}, \mathsf{mk}, \mathsf{xk}) \leftarrow \mathsf{WMSetup}(1^\lambda), \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{PESetup}(1^\lambda, \mathsf{wpp}), \\ f \leftarrow \mathcal{A}(1^\lambda, \mathsf{wpp}, \mathsf{mpk}), \\ \mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f), \\ (x, (m_0, m_1), C^*) \leftarrow \mathcal{A}^{\mathsf{Mark}(\mathsf{mk}, \mathsf{sk}_f, \cdot)}\end{array}\right] \leq \mathsf{negl}(\lambda),$$

[8] For watermarkable ABE schemes, the adversary is restricted to output a single challenge attribute (i.e., it must be that $x_0 = x_1$).

where $\mathcal{Q} \subseteq \mathcal{T}$ denotes the set of marks queried by $\mathcal{A}$ to the marking oracle[9], and $\mathcal{A}$ is said to be ε-unremovable admissible if the circuit C^* it outputs is an ε-good decoder with respect to key mpk, attribute x, and message pair (m_0, m_1). Here we say that C^* is an ε-good decoder circuit with respect to key mpk, attribute x, and message pair (m_0, m_1) if:

$$\Pr\left[C^*(\mathsf{ct}) = m_b \ : \ b \leftarrow \{0,1\}, \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, m_b)\right] \geq \frac{1}{2} + \varepsilon.$$

We can also consider a selective variant of the unremovability game, where the adversary is required to commit to its challenge attribute x *before* seeing the public parameters of the scheme. We note that selective security implies adaptive security if we use complexity leveraging [BB04] and rely on a *sub-exponential* hardness assumption.

Remark 4.3 (Bounded Collusion Resistance)**.** We say that a watermarkable predicate encryption scheme Π_{WM} is $(q_{\mathsf{key}}, q_{\mathsf{mark}})$-bounded collusion secure if the induced predicate encryption scheme is q_{key}-bounded collusion secure and the watermarking adversary in the unremovability game can make at most q_{mark} marking queries

Remark 4.4 (Stronger Notions of Unremovability)**.** In our unremovability definition, the adversary is allowed to request (multiple) marked versions of a *single* predicate encryption key sk_f. A stronger notion would be to allow the adversary to specify both a decryption function f as well as a mark τ on each marking oracle query. Such a scheme would then be secure even if an adversary could obtain marked versions of different decryption keys. Our construction does not achieve this stronger notion and we leave this as an open problem.

Remark 4.5 (Watermarking Functional Encryption)**.** A further generalization of watermarking predicate encryption is to watermark the decryption keys in a *functional encryption* scheme. One challenge here is characterizing the set of decryption keys that can be marked. For example, it is not possible to watermark a decryption key for a constant-valued function, since the adversary can implement the decryption functionality with a circuit that just computes the constant function (which, of course, removes the watermark). It seems plausible that we can watermark decryption keys corresponding to functions with "high min-entropy:" namely, functions $f\colon \mathcal{X} \to \mathcal{Y}$ where for any $y \in \mathcal{Y}$, $\Pr[x \leftarrow \mathcal{X} : f(x) = y] = \mathsf{negl}(\lambda)$. While it is straightforward to modify our construction of watermarkable predicate encryption to support marking function keys of this form, in the resulting construction, we would additionally have to provide the $\mathsf{Extract}$ algorithm a description of the function f associated with a particular decryption circuit. Whether this a reasonable modeling assumption will depend on the particular application. It is an interesting question to both develop a better understanding of the types of function families that can be watermarked as well as identify a suitable schema for watermarking general functional encryption schemes.

4.2 Building Blocks: Mixed FE, Delegatable ABE, and Jump Finding

In this section, we review the main building blocks and techniques used in our construction of a watermarkable ABE scheme.

[9]Here, we only allow the adversary to make marking queries. A stronger definition would also allow extraction queries. However, note that if the scheme has public marking and extraction procedures, then all such oracle queries are already redundant. In this work, we study watermarking encryption schemes both in the public as well as private marking/extraction setting.

4.2.1 Mixed Functional Encryption

Here we recall the notion of mixed functional encryption as introduced by Goyal, Koppula, and Waters [GKW18]. A mixed functional encryption (mixed FE) scheme with message space $\mathcal{M}$ and function class $\mathcal{F} \subseteq \mathsf{Funs}[\mathcal{M}, \{0,1\}]$, consists of five PPT algorithms $\Pi_{\mathsf{MFE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{SK\text{-}Enc}, \mathsf{Dec})$ with the following syntax:

$\mathsf{Setup}(1^\lambda) \to (\mathsf{pp}, \mathsf{msk})$. On input the security parameter λ, the setup algorithm outputs the public parameters pp and the master secret key msk.

$\mathsf{Enc}(\mathsf{pp}) \to \mathsf{ct}$. On input the public parameters pp, the normal encryption algorithm outputs a ciphertext ct.

$\mathsf{SK\text{-}Enc}(\mathsf{msk}, f) \to \mathsf{ct}$. On input the master secret key msk and a function $f \in \mathcal{F}$, the The secret-key encryption algorithm outputs a ciphertext ct.

$\mathsf{KeyGen}(\mathsf{msk}, m) \to \mathsf{sk}_m$. On input the master secret key msk and a message/input $m \in \mathcal{M}$, the The key-generation algorithm outputs a secret key sk_m.

$\mathsf{Dec}(\mathsf{sk}_m, \mathsf{ct}) \to \{0,1\}$. On input a secret key sk_m and a ciphertext ct, the decryption algorithm outputs a single bit.

Correctness. A mixed functional encryption scheme is said to be correct if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, for every $f \in \mathcal{F}$, $m \in \mathcal{M}$, the following holds:

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_m, \mathsf{ct}) = 1 : \begin{array}{c}(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda), \\ \mathsf{sk}_m \leftarrow \mathsf{KeyGen}(\mathsf{msk}, m), \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp})\end{array}\right] \geq 1 - \mathsf{negl}(\lambda),$$

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_m, \mathsf{ct}) = f(m) : \begin{array}{c}(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda), \\ \mathsf{sk}_m \leftarrow \mathsf{KeyGen}(\mathsf{msk}, m), \mathsf{ct} \leftarrow \mathsf{SK\text{-}Enc}(\mathsf{msk}, f)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Security. For security, we require that a mixed functional encryption scheme to satisfy function indistinguishability and accept indistinguishability. Formally, they are defined as follows:

q-Bounded function indistinguishability. Let $q = q(\lambda)$ be any fixed polynomial. A mixed functional encryption scheme $\Pi_{\mathsf{MFE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{SK\text{-}Enc}, \mathsf{Dec})$ is said to satisfy q-bounded function indistinguishability security if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$, such that for every $\lambda \in \mathbb{N}$ the following holds:

$$\Pr\left[\mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot),\mathsf{SK\text{-}Enc}(\mathsf{msk},\cdot)}(\mathsf{ct}) = b : \begin{array}{c}(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (f^{(0)}, f^{(1)}) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot),\mathsf{SK\text{-}Enc}(\mathsf{msk},\cdot)}(1^\lambda, \mathsf{pp}) \\ b \leftarrow \{0,1\},\ \mathsf{ct} \leftarrow \mathsf{SK\text{-}Enc}(\mathsf{msk}, f^{(b)})\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where

- $\mathcal{A}$ can make at most $q(\lambda)$ queries to $\mathsf{SK\text{-}Enc}(\mathsf{msk}, \cdot)$ oracle, and
- Every secret key query m made by adversary $\mathcal{A}$ to the $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ oracle must satisfy the condition that $f^{(0)}(m) = f^{(1)}(m)$.

q-Bounded accept indistinguishability. Let $q(\cdot)$ be any fixed polynomial. A mixed functional encryption scheme $\Pi_{\mathsf{MFE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{SK\text{-}Enc}, \mathsf{Dec})$ is said to satisfy q-bounded accept indistinguishability security if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$, such that for every $\lambda \in \mathbb{N}$ the following holds:

$$\Pr\left[\mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot),\mathsf{SK\text{-}Enc}(\mathsf{msk},\cdot)}(\mathsf{ct}_b) = b : \begin{array}{c}(\mathsf{pp},\mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ f^* \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot),\mathsf{SK\text{-}Enc}(\mathsf{msk},\cdot)}(1^\lambda,\mathsf{pp}) \\ b \leftarrow \{0,1\},\ \mathsf{ct}_1 \leftarrow \mathsf{SK\text{-}Enc}(\mathsf{msk}, f^*) \\ \mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{pp})\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where

- $\mathcal{A}$ can make at most $q(\lambda)$ queries to $\mathsf{SK\text{-}Enc}(\mathsf{msk},\cdot)$ oracle, and
- Every secret key query m made by adversary $\mathcal{A}$ to the $\mathsf{KeyGen}(\mathsf{msk},\cdot)$ oracle must satisfy the condition that $f^*(m) = 1$.

4.2.2 Delegatable Attribute-Based and Predicate Encryption

Here, we recall the notion of attribute-based encryption [SW05, GPSW06] and describe its extensions to the notion of delegatable attribute-based encryption and predicate encryption.

Attribute-based encryption. A key-policy attribute-based encryption (KP-ABE) scheme for attribute space $\mathcal{X}$, predicate class $\mathcal{F} \subseteq \mathsf{Funs}[\mathcal{X}, \{0,1\}]$, and message space $\mathcal{M}$, is a tuple of PPT algorithms $\Pi_{\mathsf{ABE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with the following syntax:

$\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$. On input the security parameter λ, the setup algorithm and outputs a master public/secret key pair $\mathsf{mpk}, \mathsf{msk}$.

$\mathsf{Enc}(\mathsf{pp}, x, m) \to \mathsf{ct}$. On input the master public key mpk, an attribute $x \in \mathcal{X}$ and a message $m \in \mathcal{M}$, the encryption algorithm outputs a ciphertext ct.

$\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$. On input the master secret key msk and a predicate $f \in \mathcal{F}$, the key-generation algorithm outputs a secret key sk_f.

$\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) \to m/\perp$. On input a secret key sk_f and a ciphertext ct, the decryption algorithm outputs either a message $m \in \mathcal{M}$ or a special symbol $\perp$.

Correctness. A key-policy attribute-based encryption scheme is said to be correct if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, for all $x \in \mathcal{X}$, $f \in \mathcal{F}$, $m \in \mathcal{M}$, the following holds

$$\Pr\left[\begin{array}{c}(f(x) = 0 \wedge \mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) \neq \perp) \vee \\ (f(x) = 1 \wedge \mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) \neq m)\end{array} : \begin{array}{c}(\mathsf{mpk},\mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda), \\ \mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f), \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, m)\end{array}\right] \leq \mathsf{negl}(\lambda).$$

ABE security. The standard notion of security for a KP-ABE scheme is that of full or adaptive security. Specifically, a key-policy attribute-based encryption scheme $\Pi_{\mathsf{ABE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be fully secure if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$, such that for every $\lambda \in \mathbb{N}$ the following holds:

$$\Pr\left[\mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{ct}) = b : \begin{array}{c}(\mathsf{mpk},\mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ ((m_0,m_1),x) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(1^\lambda,\mathsf{mpk}) \\ b \leftarrow \{0,1\},\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, m_b)\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where every predicate query $f \in \mathcal{F}$ that $\mathcal{A}$ makes to the $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ oracle must satisfy the condition that $f(x) = 0$.

Predicate encryption. A predicate encryption (PE) scheme Π_{PE} with attribute space $\mathcal{X}$, predicate class $\mathcal{F} \subseteq \mathsf{Funs}[\mathcal{X}, \{0,1\}]$, and message space $\mathcal{M}$, is syntactically identical to an ABE scheme, except we additionally require that the ciphertexts also hide the associated attribute.

Predicate encryption security. A predicate encryption scheme $\Pi_{\mathsf{PE}} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ is said to be fully secure if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$, such that for every $\lambda \in \mathbb{N}$ the following holds:

$$\Pr\left[\mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{ct}) = b : \begin{array}{c} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ ((m_0, m_1), (x_0, x_1)) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(1^\lambda, \mathsf{mpk}) \\ b \leftarrow \{0,1\},\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x_b, m_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where the adversary's key-generation queries f must satisfy one of the following conditions:

- $f(x_0) = f(x_1) = 0$ for all queried predicates f, or
- $m_0 = m_1$ and $f(x_0) = f(x_1) = 1$ for all queried predicates f.

Delegatable ABE. A delegatable ABE (DABE) scheme Π_{DABE} with attribute space $\mathcal{X}$, predicate class $\mathcal{F} \subseteq \mathsf{Funs}[\mathcal{X}, \{0,1\}]$, and message space $\mathcal{M}$, is syntactically identical to an ABE scheme, except it also has a special key-delegation algorithm defined as follows:

$\mathsf{Delegate}(\mathsf{sk}_f, g) \to \mathsf{sk}_{f,g}$. The key-delegation algorithm takes as input a predicate key sk_f and a predicate $g \in \mathcal{F}$. It outputs a delegated predicate key $\mathsf{sk}_{f,g}$.[10]

Correctness of delegation. For correctness, a delegatable ABE scheme must also satisfy the following condition: there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, for all $x \in \mathcal{X}$, $f, g \in \mathcal{F}$, $m \in \mathcal{M}$, whenever $f(x) = g(x) = 1$, the following holds

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_{f,g}, \mathsf{ct}) = m : \begin{array}{c} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda), \mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f), \\ \mathsf{sk}_{f,g} \leftarrow \mathsf{Delegate}(\mathsf{sk}_f, g), \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, m) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Delegatable ABE security. The standard notion of security for a DABE scheme is that of full or adaptive security. A delegatable key-policy ABE scheme $\Pi_{\mathsf{DABE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Delegate})$ is said to be secure if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$, such that for every $\lambda \in \mathbb{N}$ the following holds:

$$\Pr\left[\mathcal{A}^{O(\mathsf{msk},\cdot)}(\mathsf{ct}) = b : \begin{array}{c} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ ((m_0, m_1), x) \leftarrow \mathcal{A}^{O(\mathsf{msk},\cdot)}(1^\lambda, \mathsf{mpk}) \\ b \leftarrow \{0,1\},\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, m_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where the oracle $O(\mathsf{msk}, \cdot)$ is a stateful oracle initialized with parameter $n := 1$ that takes as input a tuple $(f, \mathsf{ind}, \mathsf{mode}) \in \mathcal{F} \times \mathbb{N} \times \{\mathsf{StoreKey}, \mathsf{OutputKey}, \mathsf{DelegateKey}\}$ and answers each query as follows:

[10] Here, we only consider single-hop delegation since it is already sufficient for our construction; however, it can be extended to multi-hop delegation as well.

- If $\mathsf{mode} = \mathsf{StoreKey}$, then the challenger generates $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$, stores (n, f, sk_f), and replies with $(n, \perp)$. It also updates $n := n + 1$.
- If $\mathsf{mode} = \mathsf{OutputKey}$, then the challenger first checks if there exists a key tuple of the form $(\mathsf{ind}, g, \mathsf{sk}_g)$. If no such tuple exists or if $g(x) = 1$, it outputs $\perp$. Otherwise, it replies with $(\mathsf{ind}, \mathsf{sk}_g)$.
- If $\mathsf{mode} = \mathsf{DelegateKey}$, then the challenger first checks if there exists a key tuple of the form $(\mathsf{ind}, g, \mathsf{sk}_g)$. If no such tuple exists or if $g(x) = f(x) = 1$, it outputs $\perp$. Otherwise, it generates $\mathsf{sk}_{g,f} \leftarrow \mathsf{Delegate}(\mathsf{sk}_g, f)$ and replies with $(\mathsf{ind}, \mathsf{sk}_{g,f})$.

Remark 4.6 (Collusion Resistance)**.** For any fixed polynomial $q = q(\lambda)$, we say that an ABE, PE, or DABE scheme is q-bounded collusion secure if the security property holds against all efficient adversaries that make at most $q(\lambda)$ predicate key queries.

Remark 4.7 (Selective/Adaptive Security)**.** We say that an ABE, PE, or DABE scheme is selectively secure if the security property holds against all efficient adversaries that commits to the challenge attribute x (or challenge attributes x_0, x_1 in the case of predicate encryption) *before* it sees the master public key mpk and makes key-generation queries. The scheme is said to be adaptively (or fully) secure if no such restriction is applied.

4.2.3 Jump-Finding Problem

We recall the jump-finding problem introduced in the work of Nishimaki et al. [NWZ16] for constructing flexible traitor tracing schemes (i.e., traitor tracing schemes where the space of identities that can be traced is exponential). We rely on similar techniques in our proof later.

Definition 4.8 (Noisy Jump Finding Problem [NWZ16, Definition 3.6])**.** The $(N, q, \delta, \varepsilon)$-jump-finding problem is defined as follows. An adversary chooses a set $C \subseteq [N]$ of q unknown points. Then, the adversary provides an oracle $P\colon [0, N] \to [0, 1]_{\mathbb{R}}$ with the following properties:

- $|P(N) - P(0)| \geq \varepsilon$.
- For any $x, y \in [0, N]$ where $x < y$ and $[x + 1, y] \cap C = \varnothing$, then $|P(y) - P(x)| < \delta$.

The $(N, q, \delta, \varepsilon)$-jump finding problem is to interact with the oracle P and output an element in C. In the $(N, q, \delta, \varepsilon)$-*noisy jump finding problem*, the oracle P is replaced with a randomized oracle $Q\colon [0, N] \to \{0, 1\}$ where on input $x \in [0, N]$, $Q(x)$ outputs 1 with probability $P(x)$. A fresh independent draw is chosen for each query to $Q(x)$.

Theorem 4.9 (Noisy Jump Finding Algorithm [NWZ16, Theorem 3.7])**.** *There is an efficient algorithm* $\mathsf{QTrace}^Q(\lambda, N, q, \delta, \varepsilon)$ *that runs in time* $t = \mathsf{poly}(\lambda, \log N, q, 1/\delta)$ *and makes at most* t *queries to* Q *that solves the* $(N, q, \delta, \varepsilon)$*-noisy-jump-finding problem whenever* $\varepsilon > \delta(5 + 2(\lceil \log N - 1 \rceil)q)$*. In particular,* $\mathsf{QTrace}^Q(\lambda, N, q, \delta, \varepsilon)$ *will output at least one element in* C *with probability* $1 - \mathsf{negl}(\lambda)$ *and will never output an element outside* C*. Moreover, any element* x *output by* $\mathsf{QTrace}^Q(\lambda, N, q, \delta, \varepsilon)$ *has the property that* $P(x) - P(x - 1) > \delta$*, where* $P(x) = \Pr[Q(x) = 1]$.

Remark 4.10 (Relaxed Non-Intersection Property [NWZ16, Remark 3.8])**.** The algorithm QTrace^Q in Theorem 4.9 succeeds in solving the $(N, q, \delta, \varepsilon)$-noisy-jump-finding problem even if the associated oracle P does not satisfy the second property in Definition 4.8: namely, there may exist x, y where $[x + 1, y] \cap C = \varnothing$ and $|P(y) - P(x)| \geq \delta$. As long as the property holds for all pairs x, y queried by QTrace^Q, Theorem 4.9 applies.

4.3 Watermarkable ABE from Mixed FE and Delegatable ABE

Here, we show how to construct a fully collusion resistant watermarkable ABE scheme for general predicates from any fully collusion resistant delegatable ABE scheme for general circuits and a bounded collusion secure mixed FE scheme. Then, in Appendix C, we show how to leverage traitor tracing techniques to watermark a predicate encryption scheme (with security against bounded collusions).

Construction 4.11 (Watermarkable ABE from Mixed FE and Delegatable ABE)**.** Let $\mathcal{T} = \{0,1\}^{\ell_1}$ be the mark space, $\mathcal{X} = \{0,1\}^{\ell_2}$ be the attribute space, and $\mathcal{C} \subseteq \mathsf{Funs}[\mathcal{X}, \{0,1\}]$ be the predicate class for our watermarkable ABE scheme for some parameters ℓ_1, ℓ_2. Our construction relies on the following primitives:

- Let $\Pi_{\mathsf{MFE}} = (\mathsf{MFE.Setup}, \mathsf{MFE.KeyGen}, \mathsf{MFE.Enc}, \mathsf{MFE.SK\text{-}Enc}, \mathsf{MFE.Dec})$ be a mixed functional encryption scheme for the class of comparison functions over message space $\{0,1\}^{\ell_1}$ with ciphertexts of length $\kappa(\lambda, \ell_1)$.
- Let $\mathcal{Y} = \{0,1\}^{\ell_2+\kappa}$ and $\mathcal{D} = \mathcal{C} \cup \{\mathsf{MFE.Dec}\} \cup \{\mathsf{MFE.Dec} \wedge C\}_{C \in \mathcal{C}}$.
- Let $\Pi_{\mathsf{DABE}} = (\mathsf{DABE.Setup}, \mathsf{DABE.KeyGen}, \mathsf{DABE.Enc}, \mathsf{DABE.Dec}, \mathsf{DABE.Delegate})$ be a key-policy delegatable attribute-based encryption scheme for attribute space $\mathcal{Y}$, predicate class $\mathcal{D}$, and message space $\mathcal{M}$. In particular, elements in $\mathcal{Y}$ encode an attribute $x \in \mathcal{X}$ together with a mixed FE ciphertext, and the predicate class $\mathcal{D}$ contains all circuits in class $\mathcal{C}$, the $\mathsf{MFE.Dec}$ circuit, and the conjunction of every circuit in $\mathcal{C}$ with the $\mathsf{MFE.Dec}$ circuit.
- Let ε be the unremovability parameter.

We give our construction $\Pi_{\mathsf{WM}} = (\mathsf{WMSetup}, \mathsf{PESetup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Mark}, \mathsf{Extract})$ below:

$\mathsf{WMSetup}(1^\lambda) \to (\mathsf{wpp}, \mathsf{mk}, \mathsf{xk})$. The watermarking setup algorithm runs $\mathsf{MFE.Setup}$ to generate the mixed FE public parameters and master secret key as $(\mathsf{mfe.mpk}, \mathsf{mfe.msk}) \leftarrow \mathsf{MFE.Setup}(1^\lambda)$. It sets the watermarking parameters as $\mathsf{wpp} = \mathsf{mfe.mpk}$ and $\mathsf{mk} = \mathsf{xk} = (\mathsf{wpp}, \mathsf{mfe.msk})$.

$\mathsf{ABESetup}(1^\lambda, \mathsf{wpp}) \to (\mathsf{mpk}, \mathsf{msk})$. The ABE setup algorithm runs $\mathsf{DABE.Setup}$ to generate the delegatable ABE public parameters and master secret key as $(\mathsf{dabe.mpk}, \mathsf{dabe.msk}) \leftarrow \mathsf{DABE.Setup}(1^\lambda)$. It sets the ABE master key pair as $\mathsf{mpk} = (\mathsf{dabe.mpk}, \mathsf{wpp})$ and $\mathsf{msk} = \mathsf{dabe.msk}$.

$\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$. Let $\widetilde{f}\colon \{0,1\}^{\ell_2+\kappa} \to \{0,1\}$ denote the predicate $\widetilde{f}(x,c) = f(x)$ for every $x \in \{0,1\}^{\ell_2}$ and $c \in \{0,1\}^\kappa$. The key-generation algorithm runs the DABE key generation algorithm to generate the secret key sk_f as $\mathsf{sk}_f \leftarrow \mathsf{DABE.KeyGen}(\mathsf{msk}, \widetilde{f})$.

$\mathsf{Enc}(\mathsf{mpk}, x, m) \to \mathsf{ct}$. Let $\mathsf{mpk} = (\mathsf{dabe.mpk}, \mathsf{mfe.mpk})$. The encryption algorithm first computes $\mathsf{ct}_{\mathsf{attr}} \leftarrow \mathsf{MFE.Enc}(\mathsf{mfe.mpk})$. Next, it encrypts message m under attribute $y = (x, \mathsf{ct}_{\mathsf{attr}})$ as $\mathsf{ct} \leftarrow \mathsf{DABE.Enc}(\mathsf{dabe.mpk}, y, m)$ and outputs ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to m/\perp$. The decryption algorithm computes and outputs $m \leftarrow \mathsf{DABE.Dec}(\mathsf{sk}, \mathsf{ct})$.

$\mathsf{Mark}(\mathsf{mk}, \mathsf{sk}_f, \tau) \to C$. Let $\mathsf{mk} = (\mathsf{wpp}, \mathsf{mfe.msk})$. The marking algorithm first computes $\mathsf{mfe.sk}_\tau \leftarrow \mathsf{MFE.KeyGen}(\mathsf{mfe.msk}, \tau)$. Let $g_{\mathsf{mfe.sk}_\tau}$ denote the mixed FE decryption circuit with key $\mathsf{mfe.sk}_\tau$ hardwired; that is, $g_{\mathsf{mfe.sk}_\tau} := \mathsf{MFE.Dec}(\mathsf{mfe.sk}_\tau, \cdot)$. Next, it computes a delegated

key $\mathsf{sk}_{f,\tau} \leftarrow \mathsf{DABE.Delegate}(\mathsf{sk}_f, \widetilde{g_{\mathsf{mfe.sk}_\tau}})$, where $\widetilde{g_{\mathsf{mfe.sk}_\tau}}$ denotes the following predicate: for every $x \in \{0,1\}^{\ell_2}$ and $c \in \{0,1\}^\kappa$, $\widetilde{g_{\mathsf{mfe.sk}_\tau}}(x,c) = g_{\mathsf{mfe.sk}_\tau}(c)$. It outputs the marked circuit $C \leftarrow \mathsf{DABE.Dec}(\mathsf{sk}_{f,\tau}, \cdot)$.

$\mathsf{Extract}(\mathsf{xk}, \mathsf{mpk}, x, (m_0, m_1), C, q) \to \tau/\perp$.[11] Let $\mathsf{xk} = (\mathsf{wpp}, \mathsf{mfe.msk})$ and $\mathsf{mpk} = (\mathsf{dabe.mpk}, \mathsf{wpp})$. The extraction algorithm runs the QTrace algorithm as $\tau \leftarrow \mathsf{QTrace}^{Q_C}(\lambda, 2^{\ell_1}, q, \delta, \varepsilon)$, where $\delta = \varepsilon/(5 + 2\ell_1 q)$ and oracle Q_C is simulated as follows:

> On input $\tau \in [0, 2^{\ell_1}]$:
>
> - Compute a mixed FE ciphertext as $\mathsf{ct}_{\mathsf{attr}} \leftarrow \mathsf{MFE.SK\text{-}Enc}(\mathsf{mfe.msk}, \mathsf{comp}_\tau)$, where comp_τ is the comparison function that on input z, outputs 1 if and only if $z \geq \tau$.
> - Sample a random bit $b \leftarrow \{0,1\}$ and compute a DABE ciphertext ct as $\mathsf{ct} \leftarrow \mathsf{DABE.Enc}(\mathsf{dabe.mpk}, y, m_b)$, where $y = (x, \mathsf{ct}_{\mathsf{attr}})$.
> - Compute $m' \leftarrow C(\mathsf{ct})$ and output 1 if $m' = m_b$ and 0 otherwise.

Figure 1: The extraction oracle Q_C

If $\tau = \varnothing$, then it outputs $\perp$. Otherwise, it outputs the mark as τ.

Remark 4.12 (Removing the Dependence on the Parameter q)**.** In our construction above, the extraction algorithm takes an additional parameter q. This is meant to represent the number of marked keys provided to the adversary. However, our extraction definition does not allow such an explicit parameter, and it should work for all polynomially-bounded values of q. In particular, we want to support an extraction capability even if the number of marked keys queried exceeds any specific (a priori fixed) bound q. This could be generically achieved as follows: the (unbounded) extraction algorithm simply runs the bounded version of extraction with parameter q as increasing powers of two until the extraction algorithm outputs at least one mark value.

Correctness and security analysis. We now state our correctness and security theorems, but defer their formal analysis to Section 4.3.1.

Theorem 4.13 (Correctness)**.** *If* Π_{MFE} *and* Π_{DABE} *are correct and* Π_{DABE} *is secure, then the watermarkable predicate encryption scheme* Π_{WM} *from Construction 4.11 satisfies correctness, meaningfulness, and functionality-preserving.*

Theorem 4.14 (Predicate Encryption Security)**.** *If* Π_{DABE} *is a selectively-secure attribute-based encryption, then* Π_{WM} *from Construction 4.11 is selectively secure in the presence of a malicious watermarking authority.*

Theorem 4.15 (ε-Unremovability)**.** *For any unremovability parameter* $\varepsilon = \varepsilon(\lambda) = 1/\mathsf{poly}(\lambda)$*. If* Π_{DABE} *is selectively-secure and* Π_{MFE} *is secure, then* Π_{WM} *from Construction 4.11 is selectively* ε*-unremovability.*

[11] For simplicity of exposition, we describe the $\mathsf{Extract}$ algorithm as taking an additional parameter q. We describe how to remove the dependence on q in Remark 4.12.

4.3.1 Analysis of Watermarkable ABE (Construction 4.11)

In this section, we analyze the correctness and other security properties of the watermarkable ABE scheme described above.

Proof of Theorem 4.13 (Correctness). Fix any $\lambda \in \mathbb{N}$, message $m \in \mathcal{M}$, attribute $x \in \mathcal{X}$, mark $\tau \in \mathcal{T}$, predicate $f \in \mathcal{C}$, mixed FE parameters $(\mathsf{mfe.mpk}, \mathsf{mfe.msk}) \leftarrow \mathsf{MFE.Setup}(1^\lambda)$, and DABE parameters $(\mathsf{dabe.mpk}, \mathsf{dabe.msk}) \leftarrow \mathsf{DABE.Setup}(1^\lambda)$. The unmarked predicate key sk_f for predicate f is simply the DABE predicate key $\mathsf{sk}_f \leftarrow \mathsf{DABE.KeyGen}(\mathsf{dabe.msk}, \widetilde{f})$, and a marked predicate key $\mathsf{sk}_{f,\tau}$ with mark τ is a delegated key $\mathsf{sk}_{f,\tau} \leftarrow \mathsf{DABE.Delegate}(\mathsf{sk}_f, \widetilde{g_{\mathsf{mfe.sk}_\tau}})$, where $\mathsf{mfe.sk}_\tau$ is a mixed FE secret key computed as $\mathsf{mfe.sk}_\tau \leftarrow \mathsf{MFE.KeyGen}(\mathsf{mfe.msk}, \tau)$. We show each of the properties below:

- **Correctness of decryption:** For any ciphertext ct computed as $\mathsf{ct} \leftarrow \mathsf{DABE.Enc}(\mathsf{dabe.mpk}, (x, \mathsf{ct_{attr}}), m)$, where $\mathsf{ct_{attr}} \leftarrow \mathsf{MFE.Enc}(\mathsf{mfe.mpk})$, we know that $\widetilde{f}(x, \mathsf{ct_{attr}}) = f(x)$ by the definition of $\widetilde{f}$. Therefore, by correctness of Π_{DABE}, we have that with overwhelming probability $\mathsf{DABE.Dec}(\mathsf{sk}_f, \mathsf{ct}) = m$ if $f(x) = 1$; otherwise $\mathsf{DABE.Dec}(\mathsf{sk}_f, \mathsf{ct}) = \bot$.

- **Correctness of extraction:** Fix any two messages $m_0, m_1 \in \mathcal{M}$, and let q be some polynomial in λ. Here we argue that the probability the extraction algorithm does not output τ on input $(\mathsf{wpp} = \mathsf{mfe.mpk}, \mathsf{xk} = \mathsf{mfe.msk}, \mathsf{mpk} = (\mathsf{dabe.mpk}, \mathsf{wpp}), x, m_0, m_1, C)$ is a negligible function in λ, where C is the marked circuit as described above. Note that the extraction procedure simply runs the QTrace procedure with circuit C dependent oracle Q_C as described in Fig. 1. Below, we argue that, on every input $\tau^* \in \mathcal{T}$, the oracle Q_C outputs 1 with overwhelming probability if $\tau^* \geq \tau$ and 0 otherwise.

  For any message $m \in \mathcal{M}$ and mark $\tau^* \in \mathcal{T}$, consider a ciphertext ct computed as $\mathsf{ct} \leftarrow \mathsf{DABE.Enc}(\mathsf{dabe.mpk}, (x, \mathsf{ct_{attr}}), m)$, where $\mathsf{ct_{attr}}$ is computed as $\mathsf{ct_{attr}} \leftarrow \mathsf{MFE.SK\text{-}Enc}(\mathsf{mfe.msk}, \mathsf{comp}_{\tau^*})$. Now observe that for the marked circuit $C = \mathsf{DABE.Dec}(\mathsf{sk}_{f,\tau}, \cdot)$, we have that for every ciphertext ct computed as described above, $C(\mathsf{ct}) = m$ with overwhelming probability whenever $f(x) = 1$ and $\tau \geq \tau^*$. Otherwise, $C(\mathsf{ct}) = \bot$ with overwhelming probability. This is because (with overwhelming probability) $\mathsf{MFE.Dec}(\mathsf{mfe.sk}_\tau, \mathsf{ct_{attr}}) = \mathsf{comp}_\tau(\tau^*) = (\tau^* \geq \tau)$ by correctness of Π_{MFE}, which implies that $\widetilde{g_{\mathsf{mfe.sk}_\tau}}(x, \mathsf{ct_{attr}}) = \mathsf{comp}_\tau(\tau^*)$. Therefore, by correctness of Π_{DABE}, $C(\mathsf{ct}) = m$ whenever $f(x) = 1$ and $\tau \geq \tau^*$, and otherwise, $C(\mathsf{ct}) = \bot$. Thus, whenever $f(x) = 1$, we have that Q_C outputs 1 with overwhelming probability if $\tau^* \geq \tau$, and otherwise, it outputs 0. Appealing to Theorem 4.9 and Remark 4.10, we conclude that the extraction algorithm outputs τ with overwhelming probability whenever $f(x) = 1$.

- **Meaningfulness:** We show each of the two properties:

  - *Most circuits unmarked.* Take any circuit C and any pair of messages $m_0, m_1 \in \mathcal{M}$. Let q be some polynomial in λ. Let $(\mathsf{mfe.mpk}, \mathsf{mfe.msk}) \leftarrow \mathsf{MFE.Setup}(1^\lambda)$ be the mixed FE keys sampled during the watermarking setup. Let δ be the probability the extraction algorithm does not output $\bot$ on inputs $(\mathsf{wpp} = \mathsf{mfe.mpk}, \mathsf{xk} = \mathsf{mfe.msk}, \mathsf{mpk} = (\mathsf{dabe.mpk}, \mathsf{wpp}), x, m_0, m_1, C)$, where the probability is taken over the choice of the DABE public key $\mathsf{dabe.mpk}$ and the coins used by the extraction algorithm. Suppose δ is a non-negligible function in λ. Note that the extraction procedure simply runs the QTrace procedure with the circuit C dependent oracle as described in Fig. 1. Now, by Theorem 4.9

and Remark 4.10, we know that if the extraction algorithm outputs a mark $\tau \neq \bot$, then the (probabilistic) oracle Q_C on inputs τ and $\tau - 1$ outputs 1 with probability η_τ and $\eta_{\tau-1}$, respectively, such that $|\eta_\tau - \eta_{\tau-1}| > \delta$, where the probability is taken over the randomness of the mixed FE and DABE encryption algorithms.

Since δ is non-negligible, we have that at least one of $(\eta_\tau - 1/2)$ and $(\eta_{\tau-1} - 1/2)$ is non-negligible as well. Suppose $\eta_\tau - 1/2 > \varepsilon$ for some non-negligible function ε. From this, we can conclude that Q_C breaks semantic security of the underlying DABE scheme with at least ε advantage, where the challenge attribute vector is computed as in Fig. 1 with input τ and challenge messages (m_0, m_1). This contradicts the assumption that Π_{DABE} is a secure DABE scheme.

– *Extraction fails on unmarked keys.* Fix any $\lambda \in \mathbb{N}$, a predicate $f \in \mathcal{C}$, attribute $x \in \mathcal{X}$, and messages $m_0, m_1 \in \mathcal{M}$. Also, let q be some polynomial in λ. Let $(\mathsf{mfe.mpk}, \mathsf{mfe.msk}) \leftarrow \mathsf{MFE.Setup}(1^\lambda)$ be the mixed FE keys sampled during the watermarking setup. Here, we argue that the probability the extraction algorithm does not output $\bot$ on inputs $(\mathsf{wpp} = \mathsf{mfe.mpk},\ \mathsf{xk} = \mathsf{mfe.msk},\ \mathsf{mpk} = (\mathsf{dabe.mpk}, \mathsf{wpp}),\ x,\ m_0, m_1,\ \mathsf{DABE.Dec}(\mathsf{sk}_f, \cdot))$ is a negligible function in λ. Here, $\mathsf{sk}_f \leftarrow \mathsf{DABE.KeyGen}(\mathsf{msk}, \widetilde{f})$ and the probability is taken over the choice of DABE public key $\mathsf{dabe.mpk}$ and the extraction randomness. Note that the extraction procedure simply runs the QTrace procedure with the circuit $C = \mathsf{DABE.Dec}(\mathsf{sk}_f, \cdot)$ dependent oracle Q_C as described in Fig. 1. We now argue that on every input $\tau \in \mathcal{T}$, the oracle Q_C outputs 1 with overwhelming probability.

On input $\tau \in \mathcal{T}$, the oracle Q_C begins by sampling a mixed FE ciphertext $\mathsf{ct}_{\mathsf{attr}} \leftarrow \mathsf{MFE.SK\text{-}Enc}(\mathsf{mfe.msk}, \mathsf{comp}_\tau)$. Next, it computes $\mathsf{ct} \leftarrow \mathsf{DABE.Enc}(\mathsf{dabe.mpk}, (x, \mathsf{ct}_{\mathsf{attr}}), m_b)$, where $b \leftarrow \{0,1\}$. It runs the decryption circuit $C = \mathsf{DABE.Dec}(\mathsf{sk}_f, \cdot)$ on the ciphertext ct. We know that $\widetilde{f}(x, \mathsf{ct}_{\mathsf{attr}}) = f(x)$ by the definition of $\widetilde{f}$, and so by the correctness of Π_{DABE}, we have that with overwhelming probability $C(\mathsf{ct}) = \mathsf{DABE.Dec}(\mathsf{sk}_f, \mathsf{ct}) = m_b$ if $f(x) = 1$. Therefore, the oracle Q_C outputs 1 with overwhelming probability whenever $f(x) = 1$. In other words, the oracle Q_C on every input $\tau \in \mathcal{T}$ outputs 1 with overwhelming probability. Hence, combining this with Theorem 4.9 and Remark 4.10, we get that the extraction algorithm outputs $\bot$ whenever $f(x) = 1$.

- **Functionality-preserving:** For any ciphertext $\mathsf{ct} \leftarrow \mathsf{DABE.Enc}(\mathsf{dabe.mpk}, (x, \mathsf{ct}_{\mathsf{attr}}), m)$, where $\mathsf{ct}_{\mathsf{attr}} \leftarrow \mathsf{MFE.Enc}(\mathsf{mfe.mpk})$, we know that $\widetilde{f}(x, \mathsf{ct}_{\mathsf{attr}}) = f(x)$ by the definition of $\widetilde{f}$. We also have that $\widetilde{g_{\mathsf{mfe.sk}_\tau}}(x, \mathsf{ct}_{\mathsf{attr}}) = 1$ with overwhelming probability. This is because $\widetilde{g_{\mathsf{mfe.sk}_\tau}}(x, \mathsf{ct}_{\mathsf{attr}}) = g_{\mathsf{mfe.sk}_\tau}(\mathsf{ct}_{\mathsf{attr}}) = \mathsf{MFE.Dec}(\mathsf{mfe.sk}_\tau, \mathsf{ct}_{\mathsf{attr}}) = 1$, where the last equality holds with overwhelming probability by the correctness of Π_{MFE}. By the correctness of Π_{DABE}, $C(\mathsf{ct}) = m$ whenever $f(x) = 1$.

Proof of Theorem 4.14 (ABE Security). Suppose there exists a PPT adversary $\mathcal{A}$ that has non-negligible advantage in the ABE security game for Π_{WM}. We construct an algorithm $\mathcal{B}$ that breaks semantic security of the DABE scheme:

1. The reduction algorithm $\mathcal{B}$ receives the watermarking parameter wpp and challenge attribute x^* from $\mathcal{A}$. It then computes $\mathsf{ct}^*_{\mathsf{attr}} \leftarrow \mathsf{MFE.Enc}(\mathsf{wpp})$ and sends $(x^*, \mathsf{ct}^*_{\mathsf{attr}})$ to the DABE challenger as its challenge attribute. The DABE challenger generates a key pair $(\mathsf{dabe.mpk}, \mathsf{dabe.sk})$ and sends $\mathsf{dabe.mpk}$ to $\mathcal{B}$.

2. On each predicate key query f made by $\mathcal{A}$, the reduction algorithm $\mathcal{B}$ queries the DABE challenger for a secret key for predicate $\widetilde{f}$, where $\widetilde{f}$ is as defined in the construction. Algorithm $\mathcal{B}$ forwards the challenger's response sk_f to $\mathcal{A}$.

3. Adversary $\mathcal{A}$ then sends two challenge messages (m_0^*, m_1^*) to $\mathcal{B}$, and $\mathcal{B}$ forwards (m_0^*, m_1^*) as its challenge messages to DABE challenger. Next, $\mathcal{B}$ forwards the challenge ciphertext ct^* it receives from the DABE challenger to $\mathcal{A}$.

4. Algorithm $\mathcal{B}$ answers the post-challenge key queries made by $\mathcal{A}$ as in the pre-challenge phase.

5. At the end of the game, $\mathcal{A}$ outputs a bit b, which $\mathcal{B}$ echoes as its own guess.

First, note that if $\mathcal{A}$ is an admissible adversary (i.e., the attribute x^* satisfies $f(x^*) = 0$ for every queried predicate f), then the challenge attribute $(x^*, \mathsf{ct}^*_{\mathsf{attr}})$ on each predicate $\widetilde{f}$ queried by $\mathcal{B}$ also evaluates to 0. This follows from the definition of the predicate $\widetilde{f}$ as $\widetilde{f}(x^*, \mathsf{ct}^*_{\mathsf{attr}}) = f(x^*) = 0$. Thus, the reduction algorithm $\mathcal{B}$ is also an admissible adversary in the DABE security game. Thus, $\mathcal{B}$ perfectly simulates the selective encryption security game for $\mathcal{A}$. As a result, if $\mathcal{A}$'s advantage is non-negligible, then $\mathcal{B}$ breaks DABE security with the same non-negligible advantage. □

Proof of Theorem 4.15 (Unremovability). We begin by introducing some notation for our security proof. Consider a pair of mixed FE and DABE system parameters $(\mathsf{mfe.msk}, \mathsf{mfe.mpk})$, $(\mathsf{dabe.mpk}, \mathsf{dabe.msk})$. For any decryption circuit C, attribute $x \in \mathcal{X}$, messages m_0, m_1, and mark $\tau \in [0, 2^{\ell_1}]$, let

$$p^{\tau,C}_{m_0,m_1,x} = \Pr\left[C(\mathsf{ct}) = m_b \ : \ \begin{array}{c} b \leftarrow \{0,1\}, \mathsf{ct}_{\mathsf{attr}} \leftarrow \mathsf{MFE.SK\text{-}Enc}(\mathsf{mfe.msk}, \mathsf{comp}_\tau), \\ \mathsf{ct} \leftarrow \mathsf{DABE.Enc}(\mathsf{dabe.mpk}, (x, \mathsf{ct}_{\mathsf{attr}}), m_b) \end{array}\right],$$

where the probability is taken over the random coins of the (possibly randomized) circuit C as well as the randomness used during encryption. Similarly, let

$$p^{\mathsf{nrml},C}_{m_0,m_1,x} = \Pr\left[C(\mathsf{ct}) = m_b \ : \ \begin{array}{c} b \leftarrow \{0,1\}, \mathsf{ct}_{\mathsf{attr}} \leftarrow \mathsf{MFE.Enc}(\mathsf{mfe.mpk}), \\ \mathsf{ct} \leftarrow \mathsf{DABE.Enc}(\mathsf{dabe.mpk}, (x, \mathsf{ct}_{\mathsf{attr}}), m_b) \end{array}\right].$$

The above probabilities are also parameterized by the mixed FE and DABE keys, but for simplicity of notation, we do not include them as they are clear from context.

For any adversary $\mathcal{A}$, we define the experiment $\mathsf{GetCircuit}_{\mathcal{A}}$ (see Fig. 2). The experiment is identical to the selective unremovability game, except that the challenger does not run the extraction algorithm at the end, and the output of the experiment is set to be the decryption circuit C^*, two messages m_0, m_1, and the attribute x that $\mathcal{A}$ chooses.

Using the $\mathsf{GetCircuit}$ experiment, for any mark $\tau \in [0, 2^{\ell_1}]$, we define the following probabilities, parameterized by $\gamma \in [0, 1/2]$, as a function of λ:

$$\Pr\text{-}\mathsf{Good\text{-}Dec}^{(\tau)}_{\mathcal{A},\gamma}(\lambda) = \Pr\left[p^{\tau,C^*}_{m_0,m_1,x} \geq \frac{1}{2} + \gamma \ : \ (C^*, m_0, m_1, x) \leftarrow \mathsf{GetCircuit}_{\mathcal{A}}(\lambda)\right]$$

Similarly, we also define

$$\Pr\text{-}\mathsf{Good\text{-}Dec}^{(\mathsf{nrml})}_{\mathcal{A},\gamma}(\lambda) = \Pr\left[p^{\mathsf{nrml},C^*}_{m_0,m_1,x} \geq \frac{1}{2} + \gamma \ : \ (C^*, m_0, m_1, x) \leftarrow \mathsf{GetCircuit}_{\mathcal{A}}(\lambda)\right].$$

Experiment $\mathsf{GetCircuit}_{\mathcal{A}}(\lambda)$

1. On input the security parameter 1^λ, the adversary $\mathcal{A}$ chooses a challenge attribute x and sends it to the challenger.
2. The challenger samples a mixed FE key pair $(\mathsf{mfe.mpk}, \mathsf{mfe.msk}) \leftarrow \mathsf{MFE.Setup}(1^\lambda)$ and a DABE key pair $(\mathsf{dabe.mpk}, \mathsf{dabe.msk}) \leftarrow \mathsf{DABE.Setup}(1^\lambda)$. It sends the watermarking parameters and ABE public key as $\mathsf{wpp} = \mathsf{mfe.mpk}$, $\mathsf{mpk} = (\mathsf{dabe.mpk}, \mathsf{mfe.mpk})$ to $\mathcal{A}$.
3. $\mathcal{A}$ then sends the challenge predicate f, and the challenger samples and stores a predicate key $\mathsf{sk}_f \leftarrow \mathsf{DABE.KeyGen}(\mathsf{dabe.msk}, \widetilde{f})$. (Here $\widetilde{f}$ is as defined previously.)
4. Next, $\mathcal{A}$ makes polynomially many marking queries. For every marking query $\tau \in \mathcal{T}$, the challenger computes the delegated key $\mathsf{sk}_{f,\tau}$ as described in the construction and sends the circuit $\mathsf{DABE.Dec}(\mathsf{sk}_{f,\tau}, \cdot)$ as its response.
5. Finally, $\mathcal{A}$ outputs a pair of messages (m_0, m_1) and a circuit C^*. The output of the experiment is set as (C^*, m_0, m_1, x).

Figure 2: Experiment $\mathsf{GetCircuit}_{\mathcal{A}}(\lambda)$

All the above probabilities are defined over all the random coins chosen by the challenger and adversary $\mathcal{A}$ during the $\mathsf{GetCircuit}_{\mathcal{A}}(\lambda)$ experiment.

Now, the proof idea at a high level is as follows. Suppose there exists a successful ε-unremovable attacker $\mathcal{A}$. That is, $\mathcal{A}$ produces a circuit C^* along with attribute x and message pair (m_0, m_1), after making polynomially many queries to a marking oracle, such that $p^{\mathsf{nrml},C^*}_{m_0,m_1,x} \geq 1/2 + \varepsilon$, and the extraction algorithm outputs a mark τ^* outside the set of marks queried by $\mathcal{A}$. Let $\delta = \varepsilon/(5 + 2\ell_1 q)$ as used in the construction. We first argue that it must also be the case that $p^{0,C^*}_{m_0,m_1,x} > 1/2 + \varepsilon - \delta$, as otherwise we could use $\mathcal{A}$ to break the mixed FE accept indistinguishability property. Next, we argue that $p^{2^{\ell_1},C^*}_{m_0,m_1,x} < 1/2 + \delta$, as otherwise, we could break the security of the DABE scheme. Lastly, we also show that for any two marks $\tau_1 < \tau_2$, $p^{\tau_1,C^*}_{m_0,m_1,x} - p^{\tau_2,C^*}_{m_0,m_1,x} < \delta$ as long as $\mathcal{A}$ does not make any marking query for a mark in the range $[\tau_1, \tau_2]$. This argument relies on the mixed FE function indistinguishability property. Combining these statements with the guarantees provided by the noisy jump finding algorithm (Theorem 4.9), we conclude that the extraction algorithm does not output an incorrect mark. We give the full proof below.

Lemma 4.16. *Let $\delta = \varepsilon/(5 + 2\ell_1 q)$. If Π_{MFE} is a mixed functional encryption scheme satisfying 1-bounded accept indistinguishability property, then for every ε-unremovable admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,*

$$\Pr\text{-}\mathsf{Good\text{-}Dec}^{(\mathsf{nrml})}_{\mathcal{A},\varepsilon}(\lambda) \leq \Pr\text{-}\mathsf{Good\text{-}Dec}^{(0)}_{\mathcal{A},\varepsilon-\delta}(\lambda) + \mathsf{negl}(\lambda).$$

Proof. Suppose there exists an ε-unremovable admissible adversary $\mathcal{A}$ playing the selective unremovability game such that $\Pr\text{-}\mathsf{Good\text{-}Dec}^{(\mathsf{nrml})}_{\mathcal{A},\varepsilon}(\lambda) \geq \Pr\text{-}\mathsf{Good\text{-}Dec}^{(0)}_{\mathcal{A},\varepsilon-\delta}(\lambda) + \gamma(\lambda)$ for some non-negligible function $\gamma(\cdot)$. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that can distinguish normal encryptions from secret key encryptions, which breaks the 1-bounded accept indistinguishability security of the mixed FE scheme:

1. Adversary $\mathcal{A}$ begins by choosing a challenge attribute x and sends it over to $\mathcal{B}$.
2. The mixed FE challenger samples a mixed FE key pair $(\mathsf{mfe.mpk}, \mathsf{mfe.msk}) \leftarrow \mathsf{MFE.Setup}(1^\lambda)$, and sends $\mathsf{mfe.mpk}$ to $\mathcal{B}$. Algorithm $\mathcal{B}$ samples a DABE key pair $(\mathsf{dabe.mpk}, \mathsf{dabe.msk}) \leftarrow$

$\mathsf{DABE.Setup}(1^\lambda)$, and sends the watermarking parameters $\mathsf{wpp} = \mathsf{mfe.mpk}$ and ABE public key $\mathsf{mpk} = (\mathsf{dabe.mpk}, \mathsf{mfe.mpk})$ to $\mathcal{A}$.

3. Adversary $\mathcal{A}$ chooses a challenge predicate f and sends f to $\mathcal{B}$. Algorithm $\mathcal{B}$ samples and stores a predicate key $\mathsf{sk}_f \leftarrow \mathsf{DABE.KeyGen}(\mathsf{dabe.msk}, \widetilde{f})$. (Here $\widetilde{f}$ is as defined in Construction 4.11.)

4. Adversary $\mathcal{A}$ can then make (polynomially-many) marking queries. For each marking query $\tau_i \in \mathcal{T}$, $\mathcal{B}$ forwards τ_i as its secret key query to the mixed FE challenger and receives a mixed FE secret key $\mathsf{mfe.sk}_{\tau_i} \leftarrow \mathsf{MFE.KeyGen}(\mathsf{mfe.msk}, \tau_i)$ from the mixed FE challenger. Algorithm $\mathcal{B}$ then computes the delegated key as $\mathsf{sk}_{f,\tau_i} \leftarrow \mathsf{DABE.Delegate}(\mathsf{sk}_f, \widetilde{g_{\mathsf{mfe.sk}_{\tau_i}}})$, where the predicate $\widetilde{g_{\mathsf{mfe.sk}_{\tau_i}}}$ has the mixed FE secret key $\mathsf{mfe.sk}_{\tau_i}$ hardwired inside as defined in Construction 4.11. Algorithm $\mathcal{B}$ replies to $\mathcal{A}$ with the circuit $\mathsf{DABE.Dec}(\mathsf{sk}_{f,\tau_i}, \cdot)$.

5. At the end of the game, $\mathcal{A}$ outputs a pair of messages (m_0, m_1) and a circuit C^*.

6. Algorithm $\mathcal{B}$ samples two random bits $\alpha, \beta \leftarrow \{0,1\}$ and sends comp_0 (i.e., comparison with 0) as its challenge function to the mixed FE challenger. It also makes a secret-key encryption query on the function comp_0 to the challenger. Let $\mathsf{ct}^*_{\mathsf{attr}}$ and $\mathsf{ct}^{(1)}_{\mathsf{attr}}$ denote the challenge ciphertext and the queried ciphertext $\mathcal{B}$ receives from the mixed FE challenger, respectively. Then, $\mathcal{B}$ computes a fresh mixed FE public-key ciphertext $\mathsf{ct}^{(0)}_{\mathsf{attr}} \leftarrow \mathsf{MFE.Enc}(\mathsf{mfe.mpk})$ as well as two DABE ciphertexts $\mathsf{ct}^* \leftarrow \mathsf{DABE.Enc}(\mathsf{dabe.mpk}, (x, \mathsf{ct}^*_{\mathsf{attr}}), m_\alpha)$ and $\mathsf{ct} \leftarrow \mathsf{DABE.Enc}(\mathsf{dabe.mpk}, (x, \mathsf{ct}^{(\beta)}_{\mathsf{attr}}), m_\alpha)$. Finally, $\mathcal{B}$ runs the adversary's decryption circuit C^* on ct^* and ct. If $C(\mathsf{ct}^*) = C(\mathsf{ct})$, it outputs $b' = \beta$. Otherwise, it outputs $b' = 1 - \beta$.

First, note that $\mathcal{B}$ is an admissible adversary in the 1-bounded accept indistinguishability game as the challenge function comp_0 that $\mathcal{B}$ chooses always accepts on all key queries $\mathcal{B}$ makes. Also, note that if the mixed FE challenger computed $\mathsf{ct}^*_{\mathsf{attr}}$ as a normal FE ciphertext, then $\mathcal{B}$ computes ct^* as a standard watermarkable ABE ciphertext. Otherwise it computes ct^* as a special watermarkable ABE ciphertext (as defined in the description of $p^{0,C^*}_{m_0,m_1,x}$). Thus, $\mathcal{B}$ perfectly simulates the $\mathsf{GetCircuit}$ experiment for $\mathcal{A}$. As a result, if $\Pr\text{-}\mathsf{Good\text{-}Dec}^{(\mathsf{nrml})}_{\mathcal{A},\varepsilon}(\lambda) \geq \Pr\text{-}\mathsf{Good\text{-}Dec}^{(0)}_{\mathcal{A},\varepsilon-\delta}(\lambda) + \gamma$ where δ and γ are non-negligible functions, then $\mathcal{B}$ wins in the 1-bounded accept indistinguishability security with advantage $\geq \frac{1}{2} + \frac{\gamma}{2} \cdot \delta^2$. (Here, the analysis of $\mathcal{B}$'s advantage is almost identical to that provided in the proof of [GKRW18, Claim 5.2].) This completes the proof. □

Lemma 4.17. *Let $\delta = \varepsilon/(5 + 2\ell_1 q)$. If Π_{MFE} satisfies 1-bounded function indistinguishability, then for every ε-unremovable admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ and all marks $\tau_1 < \tau_2 \in [0, 2^{\ell_1}]$,*

$$\Pr\left[p^{\tau_1,C^*}_{m_0,m_1,x} - p^{\tau_2,C^*}_{m_0,m_1,x} \geq \delta \ : \ (C^*, m_0, m_1, x) \leftarrow \mathsf{GetCircuit}_{\mathcal{A}}(\lambda)\right] \leq \mathsf{negl}(\lambda)$$

provided that $\mathcal{A}$ did not query for a marked key for a mark $\tau \in [\tau_1, \tau_2 - 1]$ in the $\mathsf{GetCircuit}$ experiment.

Proof. Suppose there exist marks $\tau^*_1 < \tau^*_2$ and an ε-unremovable admissible adversary $\mathcal{A}$ playing the selective unremovability game such that the event $p^{\tau^*_1,C^*}_{m_0,m_1,x} - p^{\tau^*_2,C^*}_{m_0,m_1,x} \geq \delta$ occurs with some non-negligible probability $\gamma(\lambda)$. We construct an algorithm $\mathcal{B}$ that can distinguish between two different secret key encryptions and, therefore, break the 1-bounded function indistinguishability security of the mixed FE scheme:

1. Adversary $\mathcal{A}$ begins by choosing a challenge attribute x and sends it over to $\mathcal{B}$.

2. The mixed FE challenger samples a mixed FE key pair $(\mathsf{mfe.mpk}, \mathsf{mfe.msk}) \leftarrow \mathsf{MFE.Setup}(1^\lambda)$, and sends $\mathsf{mfe.mpk}$ to $\mathcal{B}$. Algorithm $\mathcal{B}$ samples a DABE key pair $(\mathsf{dabe.mpk}, \mathsf{dabe.msk}) \leftarrow \mathsf{DABE.Setup}(1^\lambda)$, and sends the watermarking parameters $\mathsf{wpp} = \mathsf{mfe.mpk}$ and ABE public key $\mathsf{mpk} = (\mathsf{dabe.mpk}, \mathsf{mfe.mpk})$ to $\mathcal{A}$.

3. Adversary $\mathcal{A}$ chooses a challenge predicate f and sends f to $\mathcal{B}$. Algorithm $\mathcal{B}$ samples and stores a predicate key $\mathsf{sk}_f \leftarrow \mathsf{DABE.KeyGen}(\mathsf{dabe.msk}, \widetilde{f})$. (Here $\widetilde{f}$ is as defined previously.)

4. Adversary $\mathcal{A}$ can then make (polynomially-many) marking queries. For every marking query $\tau_i \in \mathcal{T}$, the reduction $\mathcal{B}$ aborts (and outputs a random bit) if $\tau_i \in [\tau_1, \tau_2 - 1]$. Otherwise, it forwards τ_i as its secret key query to the mixed FE challenger to receive a key $\mathsf{mfe.sk}_{\tau_i} \leftarrow \mathsf{MFE.KeyGen}(\mathsf{mfe.msk}, \tau_i)$. Algorithm $\mathcal{B}$ then computes the delegated key $\mathsf{sk}_{f,\tau_i} \leftarrow \mathsf{DABE.Delegate}(\mathsf{sk}_f, \widetilde{g_{\mathsf{mfe.sk}_{\tau_i}}})$, where the predicate $\widetilde{g_{\mathsf{mfe.sk}_{\tau_i}}}$ has the mixed FE secret key $\mathsf{mfe.sk}_{\tau_i}$ hardwired as defined in Construction 4.11. Algorithm $\mathcal{B}$ sends $\mathsf{DABE.Dec}(\mathsf{sk}_{f,\tau_i}, \cdot)$ as the marked circuit.

5. At the end of the game, $\mathcal{A}$ outputs a pair of messages (m_0, m_1) and a circuit C^*.

6. Algorithm $\mathcal{B}$ samples two random bits $\alpha, \beta \leftarrow \{0,1\}$ and sends $(\mathsf{comp}_{\tau_0^*}, \mathsf{comp}_{\tau_1^*})$ (i.e., comparison with τ_0^* and τ_1^*) as its two challenge functions to the mixed FE challenger. In addition, it also makes a secret-key encryption query for function $\mathsf{comp}_{\tau_\beta^*}$. Let $\mathsf{ct}^*_{\mathsf{attr}}$ and $\mathsf{ct}_{\mathsf{attr}}$ denote the challenge ciphertext and queried ciphertext sent by mixed FE challenger, respectively. Then, $\mathcal{B}$ computes two DABE ciphertexts $\mathsf{ct}^* \leftarrow \mathsf{DABE.Enc}(\mathsf{dabe.mpk}, (x, \mathsf{ct}^*_{\mathsf{attr}}), m_\alpha)$ and $\mathsf{ct} \leftarrow \mathsf{DABE.Enc}(\mathsf{dabe.mpk}, (x, \mathsf{ct}_{\mathsf{attr}}), m_\alpha)$. Finally, $\mathcal{B}$ runs the decryption circuit C^* on ct^* and ct, and if $C(\mathsf{ct}^*) = C(\mathsf{ct})$, it outputs $b' = \beta$. Otherwise, it outputs $b' = 1 - \beta$.

First, note that as long as $\mathcal{B}$ does not abort, then it is an admissible adversary in the 1-bounded function indistinguishability game as the challenge functions $(\mathsf{comp}_{\tau_0^*}, \mathsf{comp}_{\tau_1^*})$ $\mathcal{B}$ chooses are functionally-identical on all the key queries made by $\mathcal{B}$. Also, note that if $\mathsf{ct}^*_{\mathsf{attr}}$ was a secret-key FE encryption of $\mathsf{comp}_{\tau_b^*}$, then $\mathcal{B}$ computes ct^* as a special watermarkable ABE ciphertext (as defined in the description of $p^{\tau_b^*, C^*}_{m_0, m_1, x}$). Thus, $\mathcal{B}$ perfectly simulates the $\mathsf{GetCircuit}$ experiment for $\mathcal{A}$. As a result, if $\mathcal{A}$ outputs a marked circuit C^* such that $p^{\tau_1^*, C^*}_{m_0, m_1, x} - p^{\tau_2^*, C^*}_{m_0, m_1, x} \geq \delta$ with probability γ, where δ and γ are non-negligible functions, then $\mathcal{B}$ wins in the 1-bounded function indistinguishability security with advantage $\geq \frac{1}{2} + \frac{\gamma}{2} \cdot \delta^2$. (Here, also, the analysis of $\mathcal{B}$'s advantage is almost identical to that provided in the proof of [GKRW18, Claim 5.2].) □

Lemma 4.18. *Let $\delta = \varepsilon/(5 + 2\ell_1 q)$. If Π_{DABE} is selectively-secure, then for every ε-unremovable admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,*

$$\Pr\text{-}\mathsf{Good\text{-}Dec}^{(2^{\ell_1})}_{\mathcal{A},\delta}(\lambda) \leq \mathsf{negl}(\lambda).$$

Proof. Suppose there exists an ε-unremovable admissible adversary $\mathcal{A}$ playing the selective unremovability game such that $\Pr\text{-}\mathsf{Good\text{-}Dec}^{(2^{\ell_1})}_{\mathcal{A},\delta}(\lambda) \geq \gamma(\lambda)$ for some non-negligible function $\gamma(\cdot)$. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that can break semantic security of Π_{DABE}:

1. The adversary $\mathcal{A}$ begins by choosing its challenge attribute x and sends it to $\mathcal{B}$.

2. Algorithm $\mathcal{B}$ samples a mixed FE key pair $(\mathsf{mfe.mpk}, \mathsf{mfe.msk}) \leftarrow \mathsf{MFE.Setup}(1^\lambda)$ and computes a mixed FE ciphertext $\mathsf{ct}^*_{\mathsf{attr}} \leftarrow \mathsf{MFE.SK\text{-}Enc}(\mathsf{mfe.msk}, \mathsf{comp}_{2^{\ell_1}})$. Next, $\mathcal{B}$ sends $(x, \mathsf{ct}^*_{\mathsf{attr}})$ as its challenge attribute to the DABE challenger.

3. The DABE challenger samples a key pair $(\mathsf{dabe.mpk}, \mathsf{dabe.sk}) \leftarrow \mathsf{DABE.Setup}(1^\lambda)$ and sends $\mathsf{dabe.mpk}$ to $\mathcal{B}$. Algorithm $\mathcal{B}$ then sends the watermarking parameters $\mathsf{wpp} = \mathsf{mfe.mpk}$ and ABE public key $\mathsf{mpk} = (\mathsf{dabe.mpk}, \mathsf{mfe.mpk})$ to $\mathcal{A}$.

4. Adversary $\mathcal{A}$ then chooses a challenge predicate f, and the reduction $\mathcal{B}$ sends $(\widetilde{f}, \bot, \mathsf{StoreKey})$ as its query to the DABE challenger. (Here $\widetilde{f}$ is as defined in Construction 4.11.) The DABE challenger samples the secret key $\mathsf{sk}_f \leftarrow \mathsf{DABE.KeyGen}(\mathsf{dabe.sk}, \widetilde{f})$ but does not send sk_f to $\mathcal{B}$ since $\mathcal{B}$ queries the secret key for storing.

5. Adversary $\mathcal{A}$ can then make (polynomially-many) marking queries. For every marking query $\tau_i \in \mathcal{T}$, the reduction $\mathcal{B}$ computes a mixed FE secret key $\mathsf{mfe.sk}_{\tau_i} \leftarrow \mathsf{MFE.KeyGen}(\mathsf{mfe.msk}, \tau_i)$, and sends $(\widetilde{g_{\mathsf{mfe.sk}_{\tau_i}}}, 1, \mathsf{DelegateKey})$ to the DABE challenger as its key query. Here, the predicate $\widetilde{g_{\mathsf{mfe.sk}_{\tau_i}}}$ has the mixed FE secret key $\mathsf{mfe.sk}_{\tau_i}$ hardwired inside as defined in Construction 4.11. The challenger replies with a secret key sk_{f,τ_i}. Algorithm $\mathcal{B}$ replies to the marking query with the marked circuit $\mathsf{DABE.Dec}(\mathsf{sk}_{f,\tau_i}, \cdot)$.

6. At the end of the game, $\mathcal{A}$ outputs a pair of messages (m_0, m_1) and a circuit C^*.

7. Algorithm $\mathcal{B}$ forwards (m_0, m_1) as its challenge messages to the DABE challenger to receive a challenge ciphertext ct^*. Algorithm $\mathcal{B}$ then samples a random bit $\beta \leftarrow \{0,1\}$ and computes a fresh ciphertext $\mathsf{ct} \leftarrow \mathsf{DABE.Enc}(\mathsf{dabe.mpk}, (x, \mathsf{ct}_{\mathsf{attr}}), m_\beta)$, where $\mathsf{ct}_{\mathsf{attr}} \leftarrow \mathsf{MFE.SK\text{-}Enc}(\mathsf{mfe.msk}, \mathsf{comp}_{2^{\ell_1}})$. Finally, $\mathcal{B}$ runs the decryption circuit C^* on ct^* and ct, and if $C(\mathsf{ct}^*) = C(\mathsf{ct})$, it outputs $b' = \beta$. Otherwise, it outputs $b' = 1 - \beta$.

First, note that the challenge attribute $(x, \mathsf{ct}^*_{\mathsf{attr}})$ on each predicate query $\widetilde{g_{\mathsf{mfe.sk}_{\tau_i}}}$ made by $\mathcal{B}$ evaluates to 0, with overwhelming probability. This follows by correctness of Π_{MFE} since $\mathsf{ct}^*_{\mathsf{attr}}$ encrypts the function $\mathsf{comp}_{2^{\ell_1}}$ and for all i, $\mathsf{comp}_{2^{\ell_1}}(\tau_i) = 0$. Thus, decrypting $\mathsf{ct}^*_{\mathsf{attr}}$ using $\mathsf{mfe.sk}_{\tau_i}$ outputs 0. Then, with overwhelming probability, algorithm $\mathcal{B}$ is an admissible adversary for the DABE security game. Moreover, $\mathcal{B}$ perfectly simulates the $\mathsf{GetCircuit}$ experiment for $\mathcal{A}$. As a result, if $\mathcal{A}$ outputs a marked circuit C^* such that $\Pr\text{-}\mathsf{Good\text{-}Dec}^{(2^{\ell_1})}_{\mathcal{A},\delta}(\lambda) \geq \gamma(\lambda)$, where δ and γ are non-negligible functions, then $\mathcal{B}$ breaks semantic security of Π_{DABE} with advantage that is at least $\frac{1}{2} + \frac{\gamma}{2} \cdot \delta^2$. (Here, the analysis of $\mathcal{B}$'s advantage is almost identical to that provided in the proof of [GKRW18, Lemma 5.6].) □

Finally, combining Lemmas 4.16 to 4.18 with Theorem 4.9 and Remark 4.10, we see that the extraction algorithm outputs an element in $\mathcal{Q}$ whenever C^* is a ε-good decoder circuit as per the unremovability definition. This is because the oracle Q_{C^*} satisfies the following properties:

- $\Pr[\Pr[Q_{C^*}(0) = 1] - Q_{C^*}(2^{\ell_1}) = 1] \geq \varepsilon - \mathsf{negl}(\lambda)$ (by Lemmas 4.16 and 4.18 and the fact that $\mathcal{A}$ is an ε-unremovable admissible adversary).
- For any $\tau_1 < \tau_2 \in [0, 2^{\ell_1}]$ queried by the adversary where $[\tau_1, \tau_2 - 1] \cap \mathcal{Q} = \varnothing$, $|Q_{C^*}(\tau_1) - Q_{C^*}(\tau_2)| \leq \delta$ (Lemma 4.17).

The theorem follows. □

Remark 4.19 (Handling Extraction Queries)**.** Note that the construction described above is a watermarkable ABE scheme with private-marking and private-extraction procedures. Also, recall that in our unremovability definition, the adversary can make (a priori unbounded) polynomially many marking queries, but we do not provide an extraction oracle to the adversary. A stronger notion of security in the private-extraction setting would be to also allow the adversary to make (a priori unbounded) polynomially many extraction queries. We currently do not know how to achieve such a notion of security. However, our existing construction can be proven secure under an intermediate notion, where the adversary is instead allowed to make an a priori polynomially bounded number of extraction queries instead. This follows from the observation that to answer one extraction query, the reduction algorithm would need to make a fixed (polynomially bounded) number of mixed FE secret key encryption queries. Let $T = \mathsf{poly}(\lambda, \ell_1, q, 1/\varepsilon))$ denote the number of mixed FE encryption queries required for answering one extraction query. Thus, if the underlying mixed FE is $(N \cdot T + 1)$-bounded secure, then the resulting watermarkable ABE scheme is also secure even when the adversary is allowed to make N extraction queries.

4.4 Instantiations

Both mixed functional encryption [GKW18, CVW+18] and (fully collusion resistant) delegatable attribute-based encryption [BGG+14] can be built from the LWE assumption. Together with Construction 4.11, we obtain the following corollary:

Corollary 4.20 (Watermarkable ABE from LWE)**.** *Take any $\varepsilon = 1/\mathsf{poly}(\lambda)$ and mark space $\mathcal{T} = \{0,1\}^\ell$, where $\ell = \mathsf{poly}(\lambda)$. Under the polynomial hardness of the LWE assumption (with a super-polynomial modulus-to-noise ratio), there exists a selectively secure, fully collusion resistant watermarkable ABE scheme in the secret-key setting with mark space $\mathcal{T}$ that satisfies ε-unremovability and security against a malicious watermarking authority. If we instead assume* sub-exponential *hardness of the LWE assumption (with a super-polynomial modulus-to-noise ratio), then the watermarkable ABE scheme is adaptively secure in the secret-key setting.*

Acknowledgments

We thank Prabhanjan Ananth and Aayush Jain for helpful discussions and pointers and the anonymous CRYPTO reviewers for useful feedback on the presentation. R. Goyal was supported by an IBM PhD fellowship. S. Kim was supported by NSF, DARPA, a grant from ONR, and the Simons Foundation. N. Manohar was supported in part by a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1619348, 1228984, 1136174, and 1065276, and BSF grant 2012378. B. Waters was supported by NSF CNS-1908611, CNS-1414082, a DARPA/ARL SAFEWARE award and a Packard Foundation Fellowship. Opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

References

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. *IACR Cryptology ePrint Archive*, 2013,

2013.

[ABP+17] Shweta Agrawal, Sanjay Bhattacherjee, Duong Hieu Phan, Damien Stehlé, and Shota Yamada. Efficient public trace and revoke from standard assumptions: Extended abstract. In *ACM CCS*, 2017.

[ADM+07] Michel Abdalla, Alexander W. Dent, John Malone-Lee, Gregory Neven, Duong Hieu Phan, and Nigel P. Smart. Identity-based traitor tracing. In *PKC*, 2007.

[AV19] Prabhanjan Ananth and Vinod Vaikuntanathan. Optimal bounded-collusion secure functional encryption. *IACR Cryptology ePrint Archive*, 2019, 2019.

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT*, 2004.

[BCG+17] Zvika Brakerski, Nishanth Chandran, Vipul Goyal, Aayush Jain, Amit Sahai, and Gil Segev. Hierarchical functional encryption. In *ITCS*, 2017.

[BF99] Dan Boneh and Matthew K. Franklin. An efficient public key traitor tracing scheme. In *CRYPTO*, 1999.

[BF14] Mihir Bellare and Georg Fuchsbauer. Policy-based signatures. In *PKC*, 2014.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *EUROCRYPT*, 2014.

[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. In *CRYPTO*, 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), 2012.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *PKC*, 2014.

[BKS17] Foteini Baldimtsi, Aggelos Kiayias, and Katerina Samari. Watermarking public-key cryptographic functionalities and implementations. In *ISC*, 2017.

[BLW17] Dan Boneh, Kevin Lewi, and David J. Wu. Constraining pseudorandom functions privately. In *PKC*, 2017.

[BN08] Dan Boneh and Moni Naor. Traitor tracing with constant size ciphertext. In *ACM CCS*, 2008.

[BP08] Olivier Billet and Duong Hieu Phan. Efficient traitor tracing from collusion secure codes. In *ICITS*, 2008.

[BSW06] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *EUROCRYPT*, 2006.

[BW06] Dan Boneh and Brent Waters. A fully collusion resistant broadcast, trace, and revoke system. In *ACM CCS*, 2006.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, 2007.

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *CRYPTO*, 2014.

[CFN94] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In *CRYPTO*, 1994.

[CFNP00] Benny Chor, Amos Fiat, Moni Naor, and Benny Pinkas. Tracing traitors. *IEEE Trans. Information Theory*, 46(3), 2000.

[CHN+16] Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. In *STOC*, 2016.

[CVW+18] Yilei Chen, Vinod Vaikuntanathan, Brent Waters, Hoeteck Wee, and Daniel Wichs. Traitor-tracing from LWE made simple and attribute-based. In *TCC*, 2018.

[FNP07] Nelly Fazio, Antonio Nicolosi, and Duong Hieu Phan. Traitor tracing with optimal transmission rate. In *ISC*, 2007.

[Fre10] David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In *EUROCRYPT*, 2010.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

[GGHZ16] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Functional encryption without obfuscation. In *TCC*, 2016.

[GKRW18] Rishab Goyal, Venkata Koppula, Andrew Russell, and Brent Waters. Risky traitor tracing and new differential privacy negative results. In *Crypto*, 2018.

[GKSW10] Sanjam Garg, Abishek Kumarasubramanian, Amit Sahai, and Brent Waters. Building efficient fully collusion-resilient traitor tracing and revocation schemes. In *ACM CCS*, 2010.

[GKW18] Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In *STOC*, 2018.

[GKW19] Rishab Goyal, Venkata Koppula, and Brent Waters. New approaches to traitor tracing with embedded identities. In *TCC*, 2019.

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS*, 2006.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *CRYPTO*, 2012.

[HMW07] Nicholas Hopper, David Molnar, and David A. Wagner. From weak to strong watermarking. In *TCC*, 2007.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301), 1963.

[KD98] Kaoru Kurosawa and Yvo Desmedt. Optimum traitor tracing and asymmetric schemes. In *EUROCRYPT*, 1998.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, 2008.

[KW17] Sam Kim and David J. Wu. Watermarking cryptographic functionalities from standard lattice assumptions. In *CRYPTO*, 2017.

[KW19] Sam Kim and David J. Wu. Watermarking PRFs from lattices: Stronger security via extractable PRFs. In *CRYPTO*, 2019.

[LCW13] Zhen Liu, Zhenfu Cao, and Duncan S. Wong. Blackbox traceable CP-ABE: how to catch people leaking their keys by selling decryption devices on eBay. In *ACM CCS*, 2013.

[LPSS14] San Ling, Duong Hieu Phan, Damien Stehlé, and Ron Steinfeld. Hardness of k-LWE and applications in traitor tracing. In *CRYPTO*, 2014.

[LW15] Zhen Liu and Duncan S. Wong. Practical ciphertext-policy attribute-based encryption: Traitor tracing, revocation, and large universe. In *ACNS*, 2015.

[MPR11] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-based signatures. In *CT-RSA*, 2011.

[Nis13] Ryo Nishimaki. How to watermark cryptographic functions. In *EUROCRYPT*, 2013.

[NSS99] David Naccache, Adi Shamir, and Julien P. Stern. How to copyright a function? In *PKC*, 1999.

[NWZ16] Ryo Nishimaki, Daniel Wichs, and Mark Zhandry. Anonymous traitor tracing: How to embed arbitrary information in a key. In *EUROCRYPT*, 2016.

[PST06] Duong Hieu Phan, Reihaneh Safavi-Naini, and Dongvu Tonien. Generic construction of hybrid public key traitor tracing with full-public-traceability. In *ICALP*, 2006.

[QWZ18] Willy Quach, Daniel Wichs, and Giorgos Zirdelis. Watermarking PRFs under standard assumptions: Public marking and security with extraction queries. In *TCC*, 2018.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, 2005.

[SBC+07] Elaine Shi, John Bethencourt, Hubert T.-H. Chan, Dawn Xiaodong Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *IEEE S&P*, 2007.

[Sir06] Thomas Sirvent. Traitor tracing scheme with constant ciphertext rate against powerful pirates. *IACR Cryptology ePrint Archive*, 2006, 2006.

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *ACM CCS*, 2010.

[SSW01] Jessica Staddon, Douglas R. Stinson, and Ruizhong Wei. Combinatorial properties of frameproof and traceability codes. *IEEE Trans. Information Theory*, 47(3), 2001.

[SW98] Douglas R. Stinson and Ruizhong Wei. Combinatorial properties and constructions of traceability schemes and frameproof codes. *SIAM J. Discrete Math.*, 11(1), 1998.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, 2005.

[Tsa17] Rotem Tsabary. An equivalence between attribute-based signatures and homomorphic signatures, and new constructions for both. In *TCC*, 2017.

[Wat15] Brent Waters. A punctured programming approach to adaptively secure functional encryption. In *CRYPTO*, 2015.

[YAL+17] Rupeng Yang, Man Ho Au, Junzuo Lai, Qiuliang Xu, and Zuoxia Yu. Collusion resistant watermarking schemes for cryptographic functionalities. *IACR Cryptology ePrint Archive*, 2017, 2017.

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

A Additional Preliminaries

In this section, we recall the definitions of several cryptographic primitives.

A.1 Public-Key Encryption

A public-key encryption (PKE) scheme Π_{PKE} with message space $\mathcal{M}$ is a tuple of PPT algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ with the following syntax:

$\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$. On input the security parameter λ, the setup algorithm outputs a public/secret key-pair $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{Enc}(\mathsf{pk}, m) \to \mathsf{ct}$. On input a public key pk and a message $m \in \mathcal{M}$, the encryption algorithm outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to m/\perp$. On input a secret key sk and a ciphertext ct, the decryption algorithm either outputs a message $m \in \mathcal{M}$ or a special symbol $\perp$.

Correctness. A PKE scheme is said to be correct if for all $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$, every key-pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$, every ciphertext $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$, we have that $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = m$.

Security. For security, we require that it satisfies IND-CPA security. A PKE scheme $\Pi_{\mathsf{PKE}} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ is IND-CPA secure if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\mathcal{A}(\mathsf{ct}) = b \ : \ \begin{array}{c}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda), b \leftarrow \{0,1\}, \\ (m_0, m_1) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pk}), \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

A.2 Digital Signatures

A digital signature scheme $\Pi_{\mathsf{Sig}} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$ with message space $\mathcal{M}$ consists of three PPT algorithms with the following syntax:

$\mathsf{Setup}(1^\lambda) \to (\mathsf{sk}, \mathsf{vk})$. On input the security parameter λ, the setup algorithm outputs a pair of keys $(\mathsf{sk}, \mathsf{vk})$, where sk is the signing key and vk is the verification key.

$\mathsf{Sign}(\mathsf{sk}, m) \to \sigma$. On input a signing key sk and a message m, the signing algorithm outputs a signature σ.

$\mathsf{Verify}(\mathsf{vk}, m, \sigma) \to 0/1$. On input a verification key vk, a message m, and a signature σ, the verification algorithm outputs 1 (accepts) if verification succeeds, and 0 (rejects) otherwise.

Correctness. A signature scheme Π_{Sig} must satisfy the following correctness requirement: for all $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$, signing/verification key-pairs $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$, and every signature $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$, we have that $\mathsf{Verify}(\mathsf{vk}, m, \sigma) = 1$.

Unforgeability. The security requirement on a digital signature scheme is unforgeability. Namely, a signature scheme $\Pi_{\mathsf{Sig}} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$ satisfies unforgeability if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*) = 1 \wedge m^* \notin \mathcal{Q} : \ \begin{array}{c}(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(1^\lambda, \mathsf{vk})\end{array}\right] \leq \mathsf{negl}(\lambda),$$

where $\mathcal{Q} \subseteq \mathcal{M}$ is the set of messages $\mathcal{A}$ submitted to the Sign oracle.

A.3 Traitor Tracing with Embedded Identities

We will now recall the syntax and definitions for general traitor tracing with embedded identities [GKW19]. An embedded-identity traitor tracing system for message space $\mathcal{M}$ and identity space $\mathcal{ID} = \{\{0,1\}^\kappa\}_{\kappa \in \mathbb{N}}$ consists of five efficient algorithms $\Pi_{\mathsf{TT}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ with the following syntax:

$\mathsf{Setup}(1^\lambda, 1^\kappa) \to (\mathsf{msk}, \mathsf{pk}, \mathsf{tk})$. On input the security parameter λ and an identity space index κ, the setup algorithm outputs a master secret key msk, a public key pk, and a tracing key tk.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}) \to \mathsf{sk}_{\mathsf{id}}$. On input the master secret key msk and an identity $\mathsf{id} \in \{0,1\}^\kappa$, the key-generation algorithm outputs a secret key $\mathsf{sk}_{\mathsf{id}}$.

$\mathsf{Enc}(\mathsf{pk}, m) \to \mathsf{ct}$. On input a public key pk and a message $m \in \mathcal{M}$, the encryption algorithm outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to z/\perp$. On input a secret key sk and a ciphertext ct, the decryption algorithm either outputs a message $z \in \mathcal{M}$ or a special symbol $\perp$.

$\mathsf{Trace}^D(\mathsf{tk}, 1^y, m_0, m_1) \to T$. On input a tracing key tk, a parameter y, and two messages m_0, m_1, and given oracle access to a decoder D, the tracing algorithm outputs a set $T \subseteq \{0,1\}^\kappa$ of identities.

Correctness. A traitor tracing scheme is correct if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda, \kappa \in \mathbb{N}$, $m \in \mathcal{M}$ and identity $\mathsf{id} \in \{0,1\}^\kappa$, the following holds

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = m : \begin{array}{c}(\mathsf{msk}, \mathsf{pk}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa) \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}), \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Security. There are two security requirements for a traitor tracing scheme. First, it should satisfy IND-CPA security. Second, the tracing algorithm must (almost always) correctly trace at least one key used to create a pirate decoding box (whenever the pirate box successfully decrypts with noticeable probability), and moreover, it should not falsely accuse any user of cheating. We provide the formal definitions below:

IND-CPA security. A embedded-identity traitor tracing scheme $\Pi_{\mathsf{TT}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ is IND-CPA secure if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\mathcal{A}(\mathsf{ct}) = b \ : \ \begin{array}{c}1^\kappa \leftarrow \mathcal{A}(1^\lambda), (\mathsf{msk}, \mathsf{pk}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa), \\ b \leftarrow \{0,1\}, (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pk}), \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Secure tracing. Let $\Pi_{\mathsf{TT}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ be an embedded-identity traitor tracing scheme. For any non-negligible function $\varepsilon = \varepsilon(\lambda)$, polynomial $p = p(\lambda)$ and PPT adversary $\mathcal{A}$, consider the experiment $\mathsf{Expt\text{-}TT}^{\Pi_{\mathsf{TT}}}_{\mathcal{A},\varepsilon}(\lambda)$ defined in Fig. 3 below:

Experiment $\mathsf{Expt\text{-}TT}^{\Pi_{\mathsf{TT}}}_{\mathcal{A},\varepsilon,p}(\lambda)$

For a security parameter λ, $\mathsf{Expt\text{-}TT}^{\Pi_{\mathsf{TT}}}_{\mathcal{A},\varepsilon,p}(\lambda)$ is defined as follows:

- $1^\kappa \leftarrow \mathcal{A}(1^\lambda)$.
- $(\mathsf{msk}, \mathsf{pk}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa)$.
- $(D, m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk})$.
- $T \leftarrow \mathsf{Trace}^D(\mathsf{tk}, 1^{1/\varepsilon(\lambda)}, m_0, m_1)$.

Let $S_{\mathcal{ID}}$ be the set of identities queried by $\mathcal{A}$. Here, $\mathcal{O}(\cdot)$ is an oracle that has msk hardwired, takes as input an identity $\mathsf{id} \in \{0,1\}^\kappa$ and outputs $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$.

Figure 3: Experiment Expt-TT

Based on the above experiment, we now define the following (probabilistic) events and the corresponding probabilities (which are functions of λ and parameterized by $\mathcal{A}, \varepsilon$):

- $\mathsf{Good\text{-}Decoder}$: $\Pr[D(\mathsf{ct}) = b \ : \ b \leftarrow \{0,1\}, \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)] \geq 1/2 + \varepsilon(\lambda)$
  $\Pr\text{-}\mathsf{G\text{-}D}_{\mathcal{A},\varepsilon,p}(\lambda) = \Pr[\mathsf{Good\text{-}Decoder}]$.
- $\mathsf{Cor\text{-}Tr}$: $T \neq \varnothing \wedge T \subseteq S_{\mathcal{ID}}$
  $\Pr\text{-}\mathsf{Cor\text{-}Tr}_{\mathcal{A},\varepsilon,p}(\lambda) = \Pr[\mathsf{Cor\text{-}Tr}]$.
- $\mathsf{Fal\text{-}Tr}$: $T \not\subseteq S_{\mathcal{ID}}$
  $\Pr\text{-}\mathsf{Fal\text{-}Tr}_{\mathcal{A},\varepsilon,p}(\lambda) = \Pr[\mathsf{Fal\text{-}Tr}]$.

A traitor tracing scheme Π_{TT} is said to be secure if for every PPT adversary $\mathcal{A}$, polynomial $q(\cdot)$ and non-negligible function $\varepsilon(\cdot)$, there exists a negligible functions $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ satisfying $\varepsilon(\lambda) > 1/q(\lambda)$, the following holds

$$\Pr\text{-}\mathsf{Fal\text{-}Tr}_{\mathcal{A},\varepsilon}(\lambda) \leq \mathsf{negl}(\lambda), \quad \Pr\text{-}\mathsf{Cor\text{-}Tr}_{\mathcal{A},\varepsilon,p}(\lambda) \geq \Pr\text{-}\mathsf{G\text{-}D}_{\mathcal{A},\varepsilon,p}(\lambda) - \mathsf{negl}(\lambda).$$

B Definitions: Watermarkable Public-Key Primitives

In this section, we recall the notions of watermarkable public-key encryption and watermarkable signature schemes as introduced and formalized in the work of Cohen et al. [CHN+16]. We then discuss some of the limitations of these definitions and describe simple constructions that satisfy the existing security notions and, yet, are vulnerable to simple unremovability attacks (which are not captured by the Cohen et al. unremovability definitions).

B.1 Watermarkable Signature Schemes

We start by recalling the notion of watermarkable signatures as introduced in [CHN+16]. After that, we describe a simple generic construction that satisfies the existing definition. We then discuss the limitations of the existing definition.

B.1.1 The [CHN+16] Definition for Watermarkable Signatures

Cohen et al. [CHN+16] formalize watermarkable signature as follows. A watermarkable signature scheme with message space $\mathcal{M}$, mark space $\mathcal{T}$, and signature space $\mathcal{SIG}$ is a tuple of algorithms $(\mathsf{WM.Setup}, \mathsf{Sig.Setup}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Extract})$ with the following syntax:

$\mathsf{WM.Setup}(1^\lambda) \to (\mathsf{mk}, \mathsf{xk})$. On input the security parameter λ, the watermarking setup algorithm outputs a marking key mk and an extraction key xk.

$\mathsf{Sig.Setup}(1^\lambda, \mathsf{mk}, \tau) \to (\mathsf{sk}, \mathsf{vk})$. On input the security parameter λ, marking key mk and a mark $\tau \in \mathcal{T}$, the signature scheme setup algorithm outputs a signing-verification key pair $(\mathsf{sk}, \mathsf{vk})$.

$\mathsf{Sign}(\mathsf{sk}, m) \to \sigma$. On input the signing key sk and a message $m \in \mathcal{M}$, the signing algorithm outputs a signature σ.

$\mathsf{Verify}(\mathsf{vk}, m, \sigma) \to 0/1$. On input a verification key vk, a message $m \in \mathcal{M}$, and a signature σ, the verification algorithm outputs a bit to signify whether the signature is valid or not.

$\mathsf{Extract}(\mathsf{xk}, C) \to \tau/\perp$. On input the watermarking extraction key xk and a circuit $C\colon \mathcal{M} \to \mathcal{SIG}$, the extraction algorithm either outputs a mark $\tau \in \mathcal{T}$ or a special symbol $\perp$.

Correctness. A watermarkable signature scheme is correct if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$, $\tau \in \mathcal{T}$, the following holds

$$\Pr\left[\begin{array}{c}\mathsf{Verify}(\mathsf{vk}, m, \sigma) \neq 1 \vee \\ \mathsf{Extract}(\mathsf{xk}, \mathsf{Sign}(\mathsf{sk}, \cdot)) \neq \tau\end{array} \;:\; \begin{array}{c}(\mathsf{mk}, \mathsf{xk}) \leftarrow \mathsf{WM.Setup}(1^\lambda), \\ (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Sig.Setup}(1^\lambda, \mathsf{mk}, \tau), \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)\end{array}\right] \leq \mathsf{negl}(\lambda).$$

Security. For security, Cohen et al. defined two requirements which were unforgeability and unremovability. Intuitively, unforgeability says that the signature scheme is unforgeable as long as the adversary does not get to see the marking key mk. Additionally, unremovability requires that given a marked signing key sk, an adversary can not produce a circuit C^* such that C^* has a different mark embedded but is still functionally close to the original signing circuit $\mathsf{Sign}_{\mathsf{sk}}$. We describe the properties below:

Existential unforgeability. For every stateful PPT attacker $\mathcal{A}$, mark $\tau \in \mathcal{T}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*) = 1 : \begin{array}{c}(\mathsf{mk}, \mathsf{xk}) \leftarrow \mathsf{WM.Setup}(1^\lambda), \\ (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Sig.Setup}(1^\lambda, \mathsf{mk}, \tau), \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(1^\lambda, \mathsf{vk}, \mathsf{xk})\end{array}\right] \leq \mathsf{negl}(\lambda),$$

where $\mathcal{A}$ should never have queried m^* to Sign oracle.

ε-Unremovability. For every stateful PPT attacker $\mathcal{A}$, mark $\tau \in \mathcal{T}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\begin{array}{c}C^* \cong_\varepsilon \mathsf{Sign}(\mathsf{sk}, \cdot) \wedge \\ \mathsf{Extract}(\mathsf{xk}, C^*) \neq \tau\end{array} \;:\; \begin{array}{c}(\mathsf{mk}, \mathsf{xk}) \leftarrow \mathsf{WM.Setup}(1^\lambda), \\ (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Sig.Setup}(1^\lambda, \mathsf{mk}, \tau), \\ C^* \leftarrow \mathcal{A}(1^\lambda, \mathsf{sk}, \mathsf{vk}, \mathsf{xk})\end{array}\right] \leq \mathsf{negl}(\lambda),$$

where $C^* \cong_\varepsilon \mathsf{Sign}(\mathsf{sk}, \cdot)$ denotes that circuits C^* and $\mathsf{Sign}(\mathsf{sk}, \cdot)$ agree on an ε fraction of their inputs.

B.1.2 Simple Construction for Achieving the [CHN+16] Definition

If we allow the verification key to depend on the mark (i.e., as in the Cohen et al. abstraction), then there is a simple way to satisfy the definition. In this setting, one could take any signature scheme and make it watermarkable by directly including the mark τ as part of the signing key and verification key. A signature on a message m is just the pair (τ, σ), where σ is a vanilla signature on m. Verification first affirms that the first component of the signature is the mark τ and then checks σ as usual. If the adversary constructs a circuit that outputs valid signatures with probability better than $\varepsilon > 1/2 + 1/\mathsf{poly}(\lambda)$, then the output of the circuit contains the mark τ on a majority of inputs[12]. In this case, the extraction algorithm can evaluate the circuit on $\mathsf{poly}(\lambda)$ random inputs and output the majority tag.

[12]The unremovability definition in Cohen et al. [CHN+16] (for message-embedding watermarking) is satisfiable only when the adversary is restricted to constructing circuits that agree with the marked circuit on strictly more than half of the inputs.

Another limitation of the Cohen et al. [CHN+16] definitions is that the unremovability security definition only considers adversaries that mostly preserve the *exact input/output behavior* of the marked circuit. While this seems like a natural notion of security, it can be too restrictive in many settings. We show below (Remark B.3) that our basic construction satisfies the Cohen et al. security definitions, and, yet, there is a simple adversary that can take a marked circuit and construct from it a new circuit that outputs valid signatures but contains *no information* about the embedded watermark. The circuit produced by this adversary does not match the exact input/output behavior of the original marked circuit, and, thus, the existence of such an adversary does not violate the unremovability security requirement. This demonstrates the need for more general definitions of watermarking unremovability that better capture the full range of potential adversarial strategies.

Construction B.1 (Simple Watermarkable Signature Scheme)**.** Let $\mathcal{T}$ be the mark space and $\mathcal{M}$ be the message space for our watermarkable signature scheme. Let $\Pi_{\mathsf{Sig}} = (\mathsf{S.Setup}, \mathsf{S.Sign}, \mathsf{S.Verify})$ be a signature scheme for message space $\mathcal{M}$. Below we describe our construction $\Pi_{\mathsf{WM}} = (\mathsf{WM.Setup}, \mathsf{Sig.Setup}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Extract})$, where ε be the unremovability parameter.

$\mathsf{WM.Setup}(1^\lambda) \to (\mathsf{mk}, \mathsf{xk})$. The watermarking setup algorithm simply sets the watermarking parameters as $\mathsf{mk} = \mathsf{xk} = \bot$. Namely, there are no watermarking parameters.

$\mathsf{Sig.Setup}(1^\lambda, \mathsf{mk}, \tau) \to (\mathsf{sk}, \mathsf{vk})$. The watermarkable signature setup algorithm runs the setup for the underlying signature scheme $\mathsf{S.Setup}$ to generate secret-verification key pair as $(\mathsf{sig.sk}, \mathsf{sig.vk}) \leftarrow \mathsf{S.Setup}(1^\lambda)$. It sets the watermarked key pair as $\mathsf{sk} = (\mathsf{sig.sk}, \tau)$ and $\mathsf{vk} = (\mathsf{sig.vk}, \tau)$.

$\mathsf{Sign}(\mathsf{sk}, m) \to \sigma$. Let $\mathsf{sk} = (\mathsf{sig.sk}, \tau)$. The signing algorithm first computes $\mathsf{sig}.\sigma \leftarrow \mathsf{Sign}(\mathsf{sig.sk}, m)$. Next, it outputs the signature as $\sigma = (\mathsf{sig}.\sigma, \tau)$.

$\mathsf{Verify}(\mathsf{vk}, m, \sigma) \to 0/1$. Let $\mathsf{vk} = (\mathsf{sig.vk}, \tau_1)$ and $\sigma = (\mathsf{sig}.\sigma, \tau_2)$. The verification algorithm verifies the underlying signature as $z = \mathsf{S.Verify}(\mathsf{sig.vk}, m, \mathsf{sig}.\sigma)$, and checks that marks are identical, i.e. $\tau_1 = \tau_2$. It outputs 1 if both checks succeed. Otherwise, it outputs 0.

$\mathsf{Extract}(\mathsf{xk}, C) \to \tau/\bot$. The extraction algorithm performs the following procedure $T = \lambda/(\varepsilon - 1/2)$ times:

— For $i \in [T]$, sample message $m_i \leftarrow \mathcal{M}$ and compute $(\tau_i, \sigma_i) \leftarrow C(m)$.

Let $S = (\tau_1, \tau_2, \ldots, \tau_T)$ denote the sequence of marks as computed above. Next, let $\tau^* \in \mathcal{T}$ be such that $\tau^* = \tau_i$ for at least $T/2$ marks $\tau_i \in S$. The algorithm outputs τ^* if such a mark exists. Otherwise, it outputs $\bot$.

The correctness of the scheme follows directly from the correctness of the underlying signature scheme. For security, we can show the following.

Theorem B.2 (Watermarkable Signature Security)**.** *Fix any unremovability parameter $\varepsilon > 1/2 + 1/\mathsf{poly}(\lambda)$. If $\Pi_{\mathsf{Sig}} = (\mathsf{S.Setup}, \mathsf{S.Sign}, \mathsf{S.Verify})$ is a secure signature scheme, then $\Pi_{\mathsf{WM}} = (\mathsf{WM.Setup}, \mathsf{Sig.Setup}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Extract})$ is a secure watermarkable signature scheme (under the Cohen et al. [CHN+16] definition from Appendix B.1.1) with unremovability parameter ε.*

Proof sketch. Here, we briefly sketch the proof of unforgeability and unremovability for the watermarkable signature scheme Π_{WM} described above. First, existential unforgeability follows

directly from the security of the underlying signature scheme Π_{Sig}. This is because the watermarking marking and extraction keys are empty strings, and a marked secret-verification key pair simply consists of a signing/verification key-pair for the underlying signature scheme together with the mark τ. Thus, a valid forgery under a marked key directly gives a valid forgery under the unmarked key.

Next, we observe that ε-unforgeability follows in a straightforward manner by our construction. Let $\varepsilon = 1/2 + \delta$ for some $\delta = 1/\mathsf{poly}(\lambda)$. The main idea is as follows. Let C^* be a (randomized) circuit that is ε-close to the signing circuit $\mathsf{Sign}(\mathsf{sk}, \cdot)$ where sk is marked with mark τ. We know that the output of the signing circuit $\mathsf{Sign}(\mathsf{sk}, \cdot)$ on every input is of the form (σ, τ) for some bit string σ. Now since C^* and $\mathsf{Sign}(\mathsf{sk}, \cdot)$ agree on $1/2 + \delta$ fraction of inputs, we can appeal to a standard Chernoff bound and conclude that if we sample $T = \lambda/(\varepsilon - 1/2) = \lambda/\delta$ random messages m, then with all but $\mathsf{negl}(\lambda)$ probability, the output on a majority of these messages will be identical to the output of $\mathsf{Sign}(\mathsf{sk}, \cdot)$. In this case, the extraction algorithm successfully recovers the mark τ and unremovability follows. □

Remark B.3 (Limitations of the Cohen et al. Definition)**.** From a conceptual perspective, Construction B.1 is unsatisfying since the scheme essentially concatenates the watermark to the signature, and moreover, the verification algorithm can *ignore* the mark altogether. Thus, given a watermarked circuit C, an adversary can construct a circuit C' that, on input m, computes $(\sigma, \tau) \leftarrow C(m)$ and outputs σ. That is, the adversary's circuit only outputs the signature and drops the mark τ entirely. Of course, the extraction algorithm cannot extract a mark from C', and, yet, C' still outputs signatures that would be accepted by the verification algorithm. While this may appear to be a direct attack on the watermarking security game, the Cohen et al. model does not allow for it: namely, the unremovability adversary is constrained to output a circuit whose input/output behavior is identical to that of the marked circuit; this constraint effectively forces the adversary to include the original mark in the outputs of its new circuit. In practice, this requirement seems too restrictive, as any circuit that manages to produce valid signatures (and, yet, does not contain the mark) should be considered a valid attack on the scheme.

The definitions we consider in this work (Section 3.1) capture these types of attacks. Specifically, we require unremovability against *any* efficient adversary that produces a "useful" signing circuit (i.e., a circuit that outputs valid signatures under the verification key), irrespective of whether the outputs of the signing circuit are identical or not to those of the original circuit. We refer to Section 3.1 for further discussion and comparison of our notions with those of Cohen et al.

B.2 Watermarkable Public-Key Encryption

We start by recalling the notion of watermarkable encryption systems as introduced in [CHN+16]. We then provide some discussion of the definitions and then describe a simple generic construction from any traitor tracing scheme that satisfies the existing definition.

B.2.1 Then [CHN+16] Definition for Watermarkable Public-Key Encryption

We recall the Cohen et al. [CHN+16] definition for watermarkable public-key encryption. A watermarkable public-key encryption scheme with message space $\mathcal{M}$, mark space $\mathcal{T}$, and ciphertext space $\mathcal{CT}$ is a tuple of algorithms $(\mathsf{WM.Setup}, \mathsf{PKE.Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Extract})$ with the following syntax:

$\mathsf{WM.Setup}(1^\lambda) \to (\mathsf{mk}, \mathsf{xk})$. On input the security parameter λ, the watermarking setup algorithm outputs a marking key mk and an extraction key xk.

$\mathsf{PKE.Setup}(1^\lambda, \mathsf{mk}, \tau) \to (\mathsf{pk}, \mathsf{sk})$. On input the security parameter λ, marking key mk and a mark $\tau \in \mathcal{T}$, the PKE setup algorithm outputs a public-secret key pair $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{Enc}(\mathsf{pk}, m) \to \mathsf{ct}$. On input a public key pk and a message $m \in \mathcal{M}$, the encryption algorithm outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to m$. On input a secret key sk and a ciphertext $\mathsf{ct} \in \mathcal{CT}$, the decryption algorithm outputs a message m.

$\mathsf{Extract}(\mathsf{xk}, C) \to \tau/\perp$. On input the watermarking extraction key xk and a circuit $C\colon \mathcal{CT} \to \mathcal{M}$, the extraction algorithm either outputs a mark $\tau \in \mathcal{T}$ or a special symbol $\perp$.

Correctness. A watermarkable public-key encryption scheme is said to be correct if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$, $\tau \in \mathcal{T}$, the following holds

$$\Pr\left[\begin{array}{c} \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \neq m \vee \\ \mathsf{Extract}(\mathsf{xk}, \mathsf{Dec}_{\mathsf{sk}}) \neq \tau \end{array} : \begin{array}{c} (\mathsf{mk}, \mathsf{xk}) \leftarrow \mathsf{WM.Setup}(1^\lambda), \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Setup}(1^\lambda, \mathsf{mk}, \tau), \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Security. For security, they defined two requirements which were semantic security and unremovability. Intuitively, for encryption security they required that IND-CPA should hold even when an adversary gets to see the marking key mk and the extraction key xk. Unremovability requires that given a marked decryption key sk, an adversary cannot produce a circuit C^* such that it has a different mark embedded but is still functionally close to the original decryption circuit $\mathsf{Dec}(\mathsf{sk}, \cdot)$. Formally we describe it below:

IND-CPA. For every stateful PPT attacker $\mathcal{A}$, mark $\tau \in \mathcal{T}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\mathcal{A}(\mathsf{ct}) = b : \begin{array}{c} (\mathsf{mk}, \mathsf{xk}) \leftarrow \mathsf{WM.Setup}(1^\lambda), \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Setup}(1^\lambda, \mathsf{mk}, \tau), \\ (m_0, m_1) \leftarrow \mathcal{A}(1^\lambda, \mathsf{mk}, \mathsf{xk}, \mathsf{pk}), \\ b \leftarrow \{0,1\}, \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

ε-Unremovability. For every stateful PPT attacker $\mathcal{A}$, mark $\tau \in \mathcal{T}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\begin{array}{c} C^* \cong_\varepsilon \mathsf{Dec}(\mathsf{sk}, \cdot) \wedge \\ \mathsf{Extract}(\mathsf{xk}, C^*) \neq \tau \end{array} : \begin{array}{c} (\mathsf{mk}, \mathsf{xk}) \leftarrow \mathsf{WM.Setup}(1^\lambda), \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Setup}(1^\lambda, \mathsf{mk}, \tau), \\ C^* \leftarrow \mathcal{A}(1^\lambda, \mathsf{sk}, \mathsf{pk}, \mathsf{xk}) \end{array}\right] \leq \mathsf{negl}(\lambda),$$

where $C^* \cong_\varepsilon \mathsf{Dec}(\mathsf{sk}, \cdot)$ denotes that circuits C^* and $\mathsf{Dec}(\mathsf{sk}, \cdot)$ agree on an ε fraction of their inputs.

B.2.2 Limitations of the [CHN+16] Definition

The Cohen et al. definitions for watermarkable public-key encryption are direct analogs of their corresponding definitions for watermarkable signatures and, as such, have a similar set of limitations. Namely, there is a single algorithm responsible for both key-generation and marking (as opposed to separate algorithms), and, moreover, the unremovability guarantee does not fully capture the full range of potential adversarial strategies on the scheme itself. Here, we describe a simple construction of a watermarkable public-key encryption scheme that would be considered secure under the previous security definitions, but admits a simple and direct way to remove the watermark from marked keys. The construction is analogous to Construction B.1 for watermarkable signatures, so we just give a high-level sketch here:

- The watermarking setup algorithm samples a signing key $\mathsf{sk}_{\mathsf{Sig}}$ and a verification key $\mathsf{vk}_{\mathsf{Sig}}$ for a digital signature scheme. The marking key is the signing key $\mathsf{sk}_{\mathsf{Sig}}$ and the extraction key is the verification key $\mathsf{vk}_{\mathsf{Sig}}$.
- The (marked) key-generation algorithm samples a public key pk and a secret key sk for a vanilla public-key encryption scheme. It also computes a signature $\sigma_\tau \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{Sig}}, \tau)$ on the tag τ. An encryption of a message m is the pair $(0, \mathsf{Enc}(\mathsf{pk}, m))$. The marked decryption algorithm takes in a ciphertext (b, ct) and outputs $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ if $b = 0$ and outputs (τ, σ_τ) if $b = 1$.
- Given a circuit C, the mark-extraction algorithm simply samples λ tuples of the form $(1, \mathsf{ct})$, for an arbitrary ciphertext ct, and computes $C(\mathsf{ct})$. If it ever obtain a tuple of the form $(\tau', \sigma_{\tau'})$ where $\sigma_{\tau'}$ is a valid signature on τ' under $\mathsf{vk}_{\mathsf{Sig}}$, it outputs τ'.

Essentially, the mark τ is just "concatenated" to the decryption key itself and is output whenever the decryption circuit is invoked on a pair of the form $(1, \mathsf{ct})$. Since honestly-generated ciphertexts only outputs pairs of the form $(0, \mathsf{ct})$, this decryption procedure does not affect correctness or security of the underlying encryption scheme. This scheme satisfies the Cohen et al. definition of unremovability because any admissible adversary is required to preserve the *exact input/output* behavior of the marked decryption circuit on a $(1/2 + \varepsilon)$-fraction of the domain. In particular, this means that the adversary's circuit must output (τ, σ_τ) on an ε-fraction of pairs of the form $(1, \mathsf{ct})$. The signature is used to ensure that the extraction algorithm recovers the "correct" mark (specifically, signature unforgeability ensures that the adversary cannot produce a new pair $(\tau', \sigma_{\tau'})$, where $\tau' \neq \tau$ and $\sigma_{\tau'}$ is a valid signature on τ'). Thus, this construction satisfies the Cohen et al. definition. However, this seems like an unsatisfying construction as an adversary can construct a circuit that preserves the behavior on all inputs of the form $(0, \mathsf{ct})$ and ignores all inputs of the form $(1, \mathsf{ct})$. Such a circuit will still correctly decrypt *all* ciphertexts output by the encryption algorithm, but contains no information about the mark. In other words, the adversary has constructed an "equally-good" decryption circuit but has completely removed the watermark.

Relation to traitor tracing. The above construction motivates the need for a more flexible unremovability definition: namely, the definition should rule out any adversarial strategy that produces a "useful" decryption circuit (as opposed to a circuit that approximates the exact input-output behavior). This is the approach we take in this work when defining mark-unremovability. In this case, watermarking public-key encryption has a very similar flavor with traitor tracing. In fact, in the particular case where we allow a *single* algorithm that both generates the public/secret keys

together with the watermark, the notion is entirely subsumed by traitor tracing (even if we consider the stronger type of unremovability security that we do in this work). In some sense, watermarking as a stand-alone primitive is only interesting to study in the setting where we have independent key-generation and marking procedures; otherwise, it suffices to study traitor tracing. Here, we describe the construction of watermarking from any (embedded-identity) traitor-tracing scheme (Appendix A.3) in the Cohen et al. setting where there is a combined key-generation and marking algorithm:

Construction B.4 (Simple Watermarkable Public-Key Encryption Scheme)**.** Let $\mathcal{T} = \{0,1\}^\kappa$ be the mark space and $\mathcal{M}$ be the message space for our watermarkable PKE scheme for any parameter $\kappa \in \mathbb{N}$. We will use the following primitives:

- Let $\Pi_{\mathsf{TT}} = (\mathsf{TT.Setup}, \mathsf{TT.KeyGen}, \mathsf{TT.Enc}, \mathsf{TT.Dec}, \mathsf{TT.Trace})$ be an embedded-identity traitor tracing system for message space $\mathcal{M}$.
- Let $\Pi_{\mathsf{PKE}} = (\mathsf{P.Setup}, \mathsf{P.Enc}, \mathsf{P.Dec})$ be a PKE scheme for message space $\mathcal{M}$.

Below we describe our construction $\Pi_{\mathsf{WM}} = (\mathsf{WM.Setup}, \mathsf{PKE.Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Extract})$, where ε is the unremovability parameter.

$\mathsf{WM.Setup}(1^\lambda) \to (\mathsf{mk}, \mathsf{xk})$. The setup algorithm first samples $(\mathsf{tt.msk}, \mathsf{tt.pk}, \mathsf{tt.tk}) \leftarrow \mathsf{TT.Setup}(1^\lambda, 1^\ell)$. It sets the watermarking parameters as $\mathsf{mk} = (\mathsf{tt.pk}, \mathsf{tt.msk})$ and $\mathsf{xk} = \mathsf{tt.tk}$.

$\mathsf{PKE.Setup}(1^\lambda, \mathsf{mk}, \tau) \to (\mathsf{pk}, \mathsf{sk})$. Let $\mathsf{mk} = (\mathsf{tt.pk}, \mathsf{tt.msk})$. It samples a public/secret key-pair as $(\mathsf{pke.pk}, \mathsf{pke.sk}) \leftarrow \mathsf{P.Setup}(1^\lambda)$, and samples a traitor tracing private key as $\mathsf{tt.sk}_\tau \leftarrow \mathsf{TT.KeyGen}(\mathsf{tt.msk}, \tau)$. Finally, it sets key pair as $\mathsf{pk} = (\mathsf{pke.pk}, \mathsf{tt.pk})$ and $\mathsf{sk} = (\mathsf{pke.sk}, \mathsf{tt.sk}_\tau)$.

$\mathsf{Enc}(\mathsf{pk}, m) \to \mathsf{ct}$. Let $\mathsf{pk} = (\mathsf{pke.sk}, \mathsf{tt.pk})$. The encryption algorithm first samples a random message $r \leftarrow \mathcal{M}$, and encrypts messages $m \oplus r$ and r under the PKE and traitor tracing schemes, respectively, as $\mathsf{ct}_1 \leftarrow \mathsf{P.Enc}(\mathsf{pke.pk}, m \oplus r)$ and $\mathsf{ct}_2 \leftarrow \mathsf{TT.Enc}(\mathsf{tt.pk}, r)$. It outputs the ciphertext as $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to m/\perp$. Let $\mathsf{sk} = (\mathsf{pke.sk}, \mathsf{tt.sk})$ and $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$. The decryption algorithm decrypts ct_1 and ct_2 as $z_1 \leftarrow \mathsf{P.Dec}(\mathsf{pke.sk}, \mathsf{ct}_1)$ and $z_2 \leftarrow \mathsf{TT.Dec}(\mathsf{tt.sk}, \mathsf{ct}_2)$. It outputs $\perp$ if either z_1 or z_2 are $\perp$. Otherwise it outputs the decrypted message as $z_1 \oplus z_2$.

$\mathsf{Extract}(\mathsf{xk}, (m_0, m_1), C) \to \tau/\perp$.[13] The extraction algorithm samples $r \leftarrow \mathcal{M}$ and runs the tracing algorithm to obtain $S \leftarrow \mathsf{TT.Trace}^{D_C}(\mathsf{xk}, 1^{1/\varepsilon}, m_0 \oplus r, m_1 \oplus r)$, where every query $\widetilde{\mathsf{ct}}$ made by the tracing algorithm to decoder D_C is answered as follows:

1. On each query $\widetilde{\mathsf{ct}}$, the extraction algorithm encrypts r as $\mathsf{ct}_1 \leftarrow \mathsf{P.Enc}(\mathsf{pke.pk}, r)$.
2. Next, it runs the circuit C on $\mathsf{ct} = (\mathsf{ct}_1, \widetilde{\mathsf{ct}})$ to obtain a bit b. It gives b to the tracing algorithm.

Finally, if $S = \perp$, then the extraction algorithm outputs $\perp$. Otherwise, it outputs the mark S as the output.[14]

[13] Here, we consider the stronger unremovability notion we consider in this work where we only require that the adversary's circuit C is able to distinguish between (honest) encryptions of messages m_0 and m_1.

[14] Note that our extraction algorithm could also perform a list decoding style extraction and output more than one mark $\tau \in \mathcal{T}$ if the unremovability parameter ε was lower than $1/2$.

Correctness and security. Correctness of the scheme follows directly from the correctness of the underlying PKE and traitor tracing schemes. For IND-CPA security, we can simply rely on IND-CPA security of the underlying PKE scheme. This is because the encryption procedure for the watermarkable PKE scheme splits the message into two independent shares, and encrypts one share under the PKE scheme and the other under the traitor tracing system. This means that even if *all* the traitor tracing secret keys were leaked, we can still argue IND-CPA security for the watermarkable PKE scheme by relying on IND-CPA security of the underlying PKE scheme. For unforgeability, we only need to argue that if the adversarially created circuit C^* is ε-close to the decryption circuit, then the decoder algorithm D_{C^*} (as describe above) is a ε-good decoder according to the traitor tracing definition. This follows directly by our construction, and thus by security of tracing of the traitor tracing scheme, we obtain ε-unremovability.

Here we want to point out that for security, we do not require a fully collusion resistant secure traitor tracing scheme. On the contrary, a 1-bounded collusion secure traitor tracing scheme already suffices. Since a bounded collusion secure traitor tracing scheme can be built from any regular PKE scheme, our construction above shows that any PKE scheme in fact implies a watermarkable PKE scheme under the Cohen et al. [CHN+16] definition (even augmented with a stronger unremovability definition).

C Watermarking Predicate Encryption from Hierarchical FE

In this section, we show how to construct a bounded collusion resistant watermarkable predicate encryption scheme for general predicates from any bounded collusion resistant hierarchical functional encryption scheme for general circuits.

C.1 Building Block: Hierarchical FE

We begin by reviewing the notion of a hierarchical FE scheme that we will use in our construction. A hierarchical functional encryption (hierarchical FE) scheme [ABG+13, BCG+17] with domain $\mathcal{X}$, range $\mathcal{Y}$, and function space $\mathcal{F}$ is a tuple of algorithms $\Pi_{\mathsf{HFE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Delegate})$ with the following syntax:

$\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$. On input the security parameter λ, the setup algorithm outputs the master public key mpk and the master secret key msk.

$\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$. On input the master secret key msk and a function $f \in \mathcal{F}$, the key-generation algorithm outputs a secret key sk_f.

$\mathsf{Enc}(\mathsf{mpk}, x) \to \mathsf{ct}_x$. On input the master public key mpk and an input $x \in \mathcal{X}$, the encryption algorithm outputs a ciphertext ct_x.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to y/\perp$. On input a secret key sk and a ciphertext ct, the decryption algorithm either outputs a value $y \in \mathcal{Y}$ or a special symbol $\perp$.

$\mathsf{Delegate}(\mathsf{sk}_f, g) \to \mathsf{sk}_{g \circ f}$. On input a secret key sk_f and a function $g \in \mathcal{F}$, the delegate algorithm outputs a secret key $\mathsf{sk}_{g \circ f}$.

Correctness. For all $x \in \mathcal{X}$ and functions $f \in \mathcal{F}$, if we sample $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$, and $\mathsf{ct}_x \leftarrow \mathsf{Enc}(\mathsf{mpk}, x)$, then

$$\Pr[\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}_x) = f(x)] = 1.$$

Correctness of delegation. For all $x \in \mathcal{X}$ and functions $f, g \in \mathcal{F}$ where $g \circ f \in \mathcal{F}$, if we sample $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$, $\mathsf{sk}_{g\circ f} \leftarrow \mathsf{Delegate}(\mathsf{sk}_f, g)$, and $\mathsf{ct}_x \leftarrow \mathsf{Enc}(\mathsf{mpk}, x)$, then

$$\Pr[\mathsf{Dec}(\mathsf{sk}_{g\circ f}, \mathsf{ct}_x) = g(f(x))] = 1.$$

Note that this definition only considers correctness for *single-hop* delegation. We can define a corresponding notion of multi-hop delegation correctness. However, single-hop delegation already suffices for our construction.

Security. A hierarchical FE scheme $\Pi_{\mathsf{HFE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Delegate})$ is said to be secure if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$, such that for every $\lambda \in \mathbb{N}$ the following holds:

$$\Pr\left[\mathcal{A}^{O(\mathsf{msk},\cdot)}(\mathsf{ct}) = b : \begin{array}{c}(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (x_0, x_1) \leftarrow \mathcal{A}^{O(\mathsf{msk},\cdot)}(1^\lambda, \mathsf{mpk}) \\ b \leftarrow \{0,1\},\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x_b)\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where the oracle $O(\mathsf{msk}, \cdot)$ is a stateful oracle initialized with parameter $n := 1$, and takes as input a tuple $(f, \mathsf{ind}, \mathsf{mode}) \in \mathcal{F} \times \mathbb{N} \times \{\mathsf{StoreKey}, \mathsf{OutputKey}, \mathsf{DelegateKey}\}$, and answers each query as follows:

- If $\mathsf{mode} = \mathsf{StoreKey}$, then the challenger generates $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$, stores (n, f, sk_f) and replies with $(n, \perp)$. It also updates $n := n + 1$.
- If $\mathsf{mode} = \mathsf{OutputKey}$, then the challenger first checks if there exists a key tuple of the form $(\mathsf{ind}, g, \mathsf{sk}_g)$. If no such tuple exists or if $g(x_0) = g(x_1)$, it outputs $\perp$. Otherwise, it replies with $(\mathsf{ind}, \mathsf{sk}_g)$.
- If $\mathsf{mode} = \mathsf{DelegateKey}$, then the challenger first checks if there exists a key tuple of the form $(\mathsf{ind}, g, \mathsf{sk}_g)$. If no such tuple exists or if $f(g(x_0)) = f(g(x_1))$, it outputs $\perp$. Otherwise, it generates $\mathsf{sk}_{g,f} \leftarrow \mathsf{Delegate}(\mathsf{sk}_g, f)$ and replies with $(\mathsf{ind}, \mathsf{sk}_{g,f})$.

Remark C.1 (Collusion Resistance)**.** For any fixed polynomial $q(\cdot)$. Similar to Remark 4.6, we say that a (hierarchical) functional encryption scheme Π_{HFE} is q-bounded collusion secure if the security property holds against all efficient adversaries that make at most $q(\lambda)$ key-generation queries.

C.2 Watermarkable PE from Hierarchical FE

We now show how to construct a fully-public watermarkable PE scheme with bounded collusion resistance from any hierarchical FE scheme (which can in turn be based on the existence of any public-key encryption scheme [AV19]).

Construction C.2 (Watermarkable PE from Hierarchical FE)**.** Let $\mathcal{M} = \{0,1\}^n$ be a message space, $\mathcal{X} = \{0,1\}^\ell \setminus \{1^\ell\}$ be an attribute space, $\mathcal{F} \subseteq \mathsf{Funs}[\{0,1\}^\ell, \{0,1\}]$ be a class of predicates, and $\mathcal{T} \subseteq \mathcal{X} = \{0,1\}^\ell \setminus \{1^\ell\}$ be a mark space. Let $\varepsilon = 1/\mathsf{poly}(\lambda)$ be an unremovability parameter. We rely on the following ingredients:

- For a function $f \in \mathcal{F}$, let $g_f \colon \{0,1\}^{\ell+n+1} \to \{0,1\}^{\ell+n}$ be the function defined as follows:

$$g_f(x,m,b) = \begin{cases} (x,m) & b = 0 \\ (0^\ell, 0^n) & b = 1 \text{ and } f(x) = 0 \\ (1^\ell, m) & b = 1 \text{ and } f(x) = 1, \end{cases} \tag{C.1}$$

  where $x \in \{0,1\}^\ell$, $m \in \{0,1\}^n$, and $b \in \{0,1\}$. Define the function class $\mathcal{G} = \{f \in \mathcal{F} : g_f\}$.

- For a mark $\tau \in \{0,1\}^\ell$, define the function $h_\tau \colon \{0,1\}^{\ell+n} \to \{0,1\}^{\ell+n}$ as follows:

$$h_\tau(x,m) = \begin{cases} (0^\ell, 0^n) & x < \tau \\ (1^\ell, m) & x \geq \tau, \end{cases} \tag{C.2}$$

  where $x \in \{0,1\}^\ell$ and $m \in \{0,1\}^n$ and are interpreted as values in $[0, 2^\ell - 1]$ and $[0, 2^n - 1]$, respectively.

- Let $\Pi_{\mathsf{HFE}} = (\mathsf{HFE.Setup}, \mathsf{HFE.KeyGen}, \mathsf{HFE.Enc}, \mathsf{HFE.Dec}, \mathsf{HFE.Delegate})$ be a hierarchical FE scheme with domain $\{0,1\}^{\ell+n+1}$, range $\{0,1\}^{\ell+n}$, and function class $\mathcal{G}$. Let $\mathcal{CT}$ be the space of ciphertexts for Π_{HFE}.

We construct a watermarkable predicate encryption scheme $\Pi_{\mathsf{WM}} = (\mathsf{WMSetup}, \mathsf{PESetup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Mark}, \mathsf{Extract})$ as follows:

- $\mathsf{WMSetup}(1^\lambda) \to (\mathsf{wpp}, \mathsf{mk}, \mathsf{xk})$: On input the security parameter λ, the setup algorithm outputs $\mathsf{wpp}, \mathsf{mk}, \mathsf{xk} = \bot$. Namely, the scheme does not require any watermarking parameters.

- $\mathsf{PESetup}(1^\lambda, \mathsf{wpp}) \to (\mathsf{mpk}, \mathsf{msk})$: On input the security parameter λ and the public parameters $\mathsf{wpp} = \bot$, the key-generation algorithm outputs a key-pair $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{HFE.Setup}(1^\lambda)$.

- $\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$: On input the master secret key msk and a function $f \in \mathcal{F}$, the key-generation algorithm outputs a secret key $\mathsf{sk}_f \leftarrow \mathsf{HFE.KeyGen}(\mathsf{msk}, g_f)$, where g_f is defined in Eq. (C.1).

- $\mathsf{Enc}(\mathsf{mpk}, x, m) \to \mathsf{ct}_{x,m}$: On input the master public key mpk, an attribute $x \in \{0,1\}^\ell$, and a message $m \in \{0,1\}^n$, the encryption algorithm outputs $\mathsf{ct}_{x,m} \leftarrow \mathsf{HFE.Enc}(\mathsf{mpk}, (x, m, 1)) \in \mathcal{CT}$.

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to m/\bot$: On input a secret key sk and a ciphertext ct, the decryption algorithm computes $y \leftarrow \mathsf{HFE.Dec}(\mathsf{sk}, \mathsf{ct})$. If $y = \bot$, then output $\bot$. Otherwise, it parses $y = (x, m')$ where $x \in \{0,1\}^\ell$ and $m' \in \{0,1\}^n$. It outputs m' if $x = 1^\ell$ and $\bot$ otherwise.

- $\mathsf{Mark}(\mathsf{mk}, \mathsf{sk}, \tau) \to C_\tau$: On input the marking key $\mathsf{mk} = \bot$, a secret key sk, and a mark $\tau \in \{0,1\}^t$, the marking algorithm constructs a new key $\mathsf{sk}_\tau \leftarrow \mathsf{HFE.Delegate}(\mathsf{sk}, h_\tau)$, where h_τ is defined in Eq. (C.2). Finally, it outputs the circuit $C \colon \mathcal{CT} \to \mathcal{M} \cup \{\bot\}$ that computes the marked function $P[z, \mathsf{sk}_\tau]$ defined as follows:

On input a ciphertext $\mathsf{ct} \in \mathcal{CT}$:

1. Compute $y \leftarrow \mathsf{HFE.Dec}(\mathsf{sk}_\tau, \mathsf{ct})$. If $y = \bot$, output $\bot$.
2. Otherwise, parse $y = (x, m')$, where $x \in \{0,1\}^\ell$ and $m' \in \{0,1\}^n$. Output m' if $x = 1^\ell$ and $\bot$ otherwise.

Figure 4: The marked function $P[\mathsf{sk}_\tau]$

- $\mathsf{Extract}(\mathsf{xk}, \mathsf{mpk}, (m_0, m_1), C, q) \to \tau/\bot$:[15] On input the extraction key $\mathsf{xk} = \bot$, a master public key mpk, two messages $m_0, m_1 \in \{0,1\}^n$, and a circuit $C\colon \mathcal{CT} \to \mathcal{M} \cup \{\bot\}$, the extraction algorithm constructs the following function $Q_C\colon \{0,1\}^\ell \to \{0,1\}$:

  On input $x \in \{0,1\}^\ell$ (interpreted as a value in $[0, 2^\ell - 1]$):

  - Sample a random bit $b \leftarrow \{0,1\}$ and construct the ciphertext $\mathsf{ct} \leftarrow \mathsf{HFE.Enc}(\mathsf{mpk}, (x, m_b, 0))$.
  - Run the circuit C on ct to obtain a message $m' \leftarrow C(\mathsf{ct})$.
  - Output 1 if $m' = m_b$ and 0 otherwise.

  Figure 5: The extraction test function Q_C

  Let $\delta = \varepsilon/(5 + 2(\ell - 1)q)$ and compute $\tau \leftarrow \mathsf{QTrace}^{Q_C}(\lambda, 2^\ell - 1, q, \delta, \varepsilon)$. If $\tau = 1^\ell$, then output $\bot$. Otherwise, output τ.

Correctness and security analysis. We now state our correctness and security theorems, but defer their formal analysis to Section C.3.

Theorem C.3 (Correctness)**.** *If Π_{HFE} is correct and secure, then the watermarkable predicate encryption scheme Π_{WM} from Construction C.2 satisfies correctness, meaningfulness, and functionality-preserving.*

Theorem C.4 (Predicate Encryption Security)**.** *If Π_{HFE} is secure, then the watermarkable predicate encryption scheme Π_{WM} Construction C.2 satisfies encryption security in the presence of a malicious authority.*

Theorem C.5 (Unremovability)**.** *Take any $\varepsilon = 1/\mathsf{poly}(\lambda)$. If Π_{HFE} is secure, then the watermarkable predicate encryption scheme Π_{WM} from Construction C.2 is ε-unremovable.*

[15] This particular $\mathsf{Extract}$ algorithm does *not* need to take in an attribute $x \in \mathcal{X}$ as input. Similar to our construction from Section 4.3, we provide a bound q as an additional argument to the algorithm. We can use the same technique from Remark 4.12 to obtain an algorithm that does not depend on q.

C.3 Analysis of Watermarkable Predicate Encryption

In this section, we give the correctness and security analysis of the watermarkable predicate encryption scheme from Appendix C.2.

Proof of Theorem C.3 (Correctness). We check each of the properties below:

- **Correctness of decryption:** Take any message $m^* \in \mathcal{M}$, any tag $\tau \in \mathcal{T}$, any attribute $x \in \mathcal{X}$, and a function $f \in \mathcal{F}$. Let $(\mathsf{wpp}, \mathsf{mk}, \mathsf{xk}) \leftarrow \mathsf{WMSetup}(1^\lambda)$. Take $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{PESetup}(1^\lambda, \mathsf{wpp})$, $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$, and $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, m^*)$. By construction $\mathsf{ct} \leftarrow \mathsf{HFE.Enc}(\mathsf{mpk}, (x, m, 1))$, and $\mathsf{sk}_f \leftarrow \mathsf{HFE.KeyGen}(\mathsf{msk}, g_f)$. By definition of g_f (Eq. (C.1)), we have that $g_f(x, m, 1) = (1^\ell, m)$ if $f(x) = 1$ and $g_f(x, m, 0) = (0^\ell, 0^n)$ if $f(x) = 0$. Thus, by correctness of of Π_{HFE}, $\mathsf{HFE.Dec}(\mathsf{sk}_f, \mathsf{ct}_{x,m})$ outputs $(1^\ell, m)$ when $f(x) = 1$ and $(0^\ell, 0^n)$ when $f(x) = 0$. In this case, Dec outputs m when $f(x) = 1$ and $\perp$ when $f(x) = 0$.

- **Correctness of extraction:** Take any two messages $m_0, m_1 \in \mathcal{M}$, any tag $\tau \in \mathcal{T}$, an attribute $x \in \mathcal{X}$, and a function $f \in \mathcal{F}$. Let $(\mathsf{wpp}, \mathsf{mk}, \mathsf{xk}) \leftarrow \mathsf{WMSetup}(1^\lambda)$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{PESetup}(1^\lambda, \mathsf{wpp})$, $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$, and $C \leftarrow \mathsf{Mark}(\mathsf{mk}, \mathsf{sk}_f, \tau)$. Fix any polynomial q in λ. Let Q_C be the extraction test function from Fig. 5. We claim that on every input $\tau^* \in \mathcal{T}$, the function Q_C outputs 1 if $\tau^* \geq \tau$ and and 0 otherwise.

  Take $\tau^* \in \mathcal{T}$, and consider the value $Q_C(\tau^*)$. Then, Q_C constructs a ciphertext $\mathsf{ct} \leftarrow \mathsf{HFE.Enc}(\mathsf{mpk}, (\tau^*, m_b, 0))$, and computes $C(\mathsf{ct}) = P[\mathsf{sk}_\tau](\mathsf{ct})$. By definition, $P[\mathsf{sk}_\tau]$ starts by computing $\mathsf{HFE.Dec}(\mathsf{sk}_\tau, \mathsf{ct})$. By correctness of Π_{HFE}, $\mathsf{HFE.Dec}(\mathsf{sk}_\tau, \mathsf{ct})$ outputs

  $$h_\tau(g_f(\tau^*, m_b, 0)) = h_\tau(\tau^*, m_b).$$

  When $\tau^* \geq \tau$, decryption outputs $(1^\ell, m_b)$ and $P[\mathsf{sk}_\tau]$ outputs m_b. Otherwise, if $\tau^* < \tau$, decryption outputs $(0^\ell, 0^n)$ and $P[\mathsf{sk}_\tau]$ outputs $\perp$. Thus, Q_C outputs 1 if $\tau^* \geq \tau$ and 0 otherwise. The claim then follows from Theorem 4.9.

- **Meaningfulness:** We show the two conditions separately:

  - *Most circuits unmarked.* Take any fixed circuit $C \colon \mathcal{CT} \to \mathcal{M} \cup \{\perp\}$ and any pair of messages $m_0, m_1 \in \mathcal{M}$. Sample $(\mathsf{wpp}, \mathsf{mk}, \mathsf{xk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{PESetup}(1^\lambda, \mathsf{wpp})$. Consider $\mathsf{Extract}(\mathsf{xk}, \mathsf{mpk}, (m_0, m_1), C)$, and, correspondingly, the behavior of the extraction test algorithm Q_C. By semantic security of Π_{HFE}, the probability that the circuit C (which is chosen independently of the parameters for the hierarchical FE scheme) is able to distinguish between honestly-generated encryptions of $(x, m_0, 0)$ and $(x, m_1, 1)$ for any $x \in \mathcal{X}$ is negligible. Thus, for all $x \in \{0,1\}^\ell$, $Q_C(x)$ outputs 1 with probability that is negligibly close to $1/2$. In this case, QTrace^{Q_C} always outputs $\perp$, and the claim follows.
  - *Extraction fails on unmarked keys.* Let $(\mathsf{wpp}, \mathsf{mk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and take any two messages $m_0, m_1 \in \mathcal{M}$ and a function $f \in \mathcal{F}$. Let $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{PESetup}(1^\lambda, \mathsf{wpp})$ and $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$. Consider the output of $\mathsf{Extract}(\mathsf{xk}, \mathsf{mpk}, (m_0, m_1), \mathsf{Dec}(\mathsf{sk}_f, \cdot))$. We consider the properties of $Q_{\mathsf{Dec}(\mathsf{sk}_f, \cdot)}$. By correctness of Π_{HFE}, for any $x \in \{0,1\}^\ell$, $\mathsf{HFE.Dec}(\mathsf{sk}_f, \mathsf{HFE.Enc}(\mathsf{mpk}, (x, m_b, 0))) = (x, m_b)$. This means that $Q_{\mathsf{Dec}(\mathsf{sk}_f, \cdot)}(x) = 0$ for all $x \neq 1^\ell$, and $Q_{\mathsf{Dec}(\mathsf{sk}_f, \cdot)}(x) = 1$ when $x = 1^\ell$. By Theorem 4.9, this means that

$\mathsf{QTrace}^{Q_{\mathsf{Dec}(\mathsf{sk}_f,\cdot)}}$ will always output 1^ℓ and $\mathsf{Extract}(\mathsf{xk}, \mathsf{mpk}, (m_0, m_1), \mathsf{HFE.Dec}(\mathsf{sk}, \cdot), q)$ outputs $\perp$ for all $q = q(\lambda)$.

- **Functionality-preserving:** Functionality-preserving follows from correctness of Π_{HFE}. Take any attribute $x \in \mathcal{X}$, message $m \in \mathcal{M}$, function $f \in \mathcal{F}$, and tag $\tau \in \mathcal{T}$. Sample $(\mathsf{wpp}, \mathsf{mk}, \mathsf{xk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$, $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, m)$, and $C \leftarrow \mathsf{Mark}(\mathsf{mk}, \mathsf{sk}_f, \tau)$. In this case, ct is an encryption of $(x, m, 1)$ under Π_{HFE}. Suppose $f(x) = 1$, and consider the value of $C(\mathsf{ct})$. The circuit C first computes $\mathsf{HFE.Dec}(\mathsf{sk}_\tau, \mathsf{ct})$ which by correctness of Π_{HFE} outputs $h_\tau(g_f(x, m, 1)) = h_\tau(1^\ell, m) = (1^\ell, m)$. By definition, the output of $C(\mathsf{ct})$ is then m, and the claim follows.

Proof of Theorem C.4 (Predicate Encryption Security). Suppose there exists an (admissible) adversary $\mathcal{A}$ that breaks security of the underlying predicate encryption scheme in the presence of a malicious adversary. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ for security of Π_{HFE}:

1. Algorithm $\mathcal{B}$ receives the master public key mpk for the hierarchical FE scheme from the Π_{HFE} challenger. It gives mpk to $\mathcal{A}$. Note that since there are no public parameters in Construction C.2, the adversary $\mathcal{A}$ does not get to choose any public parameters.

2. When $\mathcal{A}$ makes a key-generation query on a function $f \in \mathcal{F}$, algorithm $\mathcal{B}$ makes a key-generation query for the function g_f to Π_{HFE} and obtains a key sk_f. Algorithm $\mathcal{B}$ sends sk_f to $\mathcal{A}$.

3. When $\mathcal{A}$ makes a challenge query on a pair of attributes $x_0, x_1 \in \mathcal{X}$ and messages $m_0, m_1 \in \mathcal{M}$, algorithm $\mathcal{B}$ submits $(x_0, m_0, 1)$ and $(x_1, m_1, 1)$ as its challenge query to the Π_{HFE} challenger and receives a ciphertext ct. Algorithm $\mathcal{B}$ replies to $\mathcal{A}$ with ct.

4. At the end of the game, algorithm $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$, which $\mathcal{B}$ outputs.

First, we argue that $\mathcal{B}$ is admissible. Since $\mathcal{A}$ is an admissible adversary for the predicate encryption scheme, it holds that for all key-generation queries $f \in \mathcal{F}$ that $\mathcal{A}$ makes, one of the following properties hold:

- $f(x_0) = 0 = f(x_1)$. In this case, $g_f(x_0, m_0, 1) = (0^\ell, 0^n) = g_f(x_1, m_1, 1)$.
- $f(x_0) = 1 = f(x_1)$ and $m_0 = m_1$. In this case,

$$g_f(x_0, m_0, 1) = (1^\ell, m_0) = (1^\ell, m_1) = g_f(x_1, m_1, 1).$$

Thus, $\mathcal{B}$ is admissible. Finally, if ct' is an encryption of $(x_0, m_0, 1)$, then $\mathcal{B}$ perfectly simulates the experiment where $b = 0$ while if ct' is an encryption of $(x_1, m_1, 1)$, then $\mathcal{B}$ perfectly simulates the experiment where $b = 1$. Thus, if $\mathcal{A}$ breaks security of the induced predicate encryption scheme with non-negligible advantage, $\mathcal{B}$ breaks security of Π_{HFE} with the same advantage. □

Proof of Theorem C.5 (Unremovability). Let $\mathcal{A}$ be an efficient ε-unremoving-admissible adversary for Π_{WM}. Let C^* be the circuit that $\mathcal{A}$ outputs at the end of the security game. First, define $P_{C^*}(x) := \Pr[Q_{C^*}(x) = 1]$. We now show that the extraction test function Q_{C^*} from Figure 5 used by the $\mathsf{Extract}$ function satisfies each of the conditions from Theorem 4.9.

Lemma C.6. *If Π_{HFE} is secure, then $P_{C^*}(0) \leq 1/2 + \mathsf{negl}(\lambda)$.*

Proof. Suppose that with non-negligible probability ε_1, algorithm $\mathcal{A}$ outputs a circuit C^* where $P_{C^*}(0) \geq \varepsilon_2 = 1/2 + \varepsilon_2'$ for some $\varepsilon_2' = 1/\mathsf{poly}(\lambda)$. We use $\mathcal{A}$ to build an adversary $\mathcal{B}$ for Π_{HFE}:

- **Simulating the unremovability game:** At the beginning of the game, algorithm $\mathcal{B}$ receives the master public key mpk from the hierarchical FE challenger. It gives mpk to $\mathcal{A}$. Algorithm $\mathcal{A}$ then specifies a function f and $\mathcal{B}$ issues a key-generation query for the function $(g_f, 0, \mathsf{StoreKey})$ to $\mathcal{A}$, where g_f is as defined in Eq. (C.1). Algorithm $\mathcal{B}$ receives a tuple $(1, \perp)$ from the challenger. Whenever $\mathcal{A}$ makes a marking oracle query on a mark $\tau \in \{0,1\}^\ell$, the challenger makes a key-generation query $(h_\tau, 1, \mathsf{OutputKey})$ where h_τ is as defined in Eq. (C.2), and obtains a tuple (i, sk_τ) for some $i \in \mathbb{N}$. Algorithm $\mathcal{B}$ replies to $\mathcal{A}$ with a circuit $C\colon \mathcal{CT} \to \mathcal{M} \cup \{\perp\}$ that computes the marked function $P[\mathsf{sk}_\tau]$ from Figure 4. At the end of the simulation, $\mathcal{A}$ outputs an attribute $x \in \mathcal{X}$, two messages $m_0, m_1 \in \mathcal{M}$, and a circuit $C^*\colon \mathcal{CT} \to \mathcal{M} \cup \{\perp\}$.
- **Good decoder check:** Let $\delta = \varepsilon_2'/4$, $\varepsilon' = \varepsilon_2'/2$, and $\xi = \lambda/\delta^2$. Algorithm $\mathcal{B}$ samples $b_i \leftarrow \{0,1\}$, sets $m_i \leftarrow m_{b_i}$, and computes $\mathsf{ct}_i \leftarrow \mathsf{HFE.Enc}(\mathsf{mpk}, (0^\ell, m_i, 0))$ for all $i \in [\xi]$. Let N be the number of indices $i \in [\xi]$ where $C^*(\mathsf{ct}_i) = m_i$. If $N < (1/2 + \varepsilon' + \delta)\xi$, then $\mathcal{B}$ sets the Bad flag.
- **Output computation:** If algorithm $\mathcal{B}$ has not set the Bad flag, then it submits the pair $(0^\ell, m_0, 0)$ and $(0^\ell, m_1, 0)$ as its challenge query and obtains a challenge ciphertext ct^*. It computes $m' \leftarrow C^*(\mathsf{ct}^*)$ and outputs 0 if $m' = m_0$ and 1 if $m' = m_1$. If $m' \neq m_0$ and $m' \neq m_1$, or the Bad flag has been set, then $\mathcal{B}$ outputs a random bit.

First, we argue that $\mathcal{B}$ is admissible for the hierarchical FE security game. By definition, for all $\tau \in \{0,1\}^\ell$ and all $m \in \mathcal{M}$, we have $h_\tau(g_f(0^\ell, m, 0)) = h_\tau(0^\ell, m) = (0^\ell, 0^n)$. Next, by construction, $\mathcal{B}$ perfectly simulates the unremovability security game for $\mathcal{A}$ so with probability ε_1, $\mathcal{A}$ outputs a circuit C^* where $P_{C^*}(0) \geq 1/2 + \varepsilon_2$. Let $b \in \{0,1\}$ be the bit sampled by the Π_{HFE} challenger. We compute the probability that $\mathcal{B}$ outputs b. We now show that with noticeable probability, $\mathcal{B}$ does not set the Bad flag, and moreover, when the Bad flag is not set, algorithm $\mathcal{B}$ achieves a non-negligible distinguishing advantage.

- We show that with probability at least $\varepsilon_1 - \mathsf{negl}(\lambda)$, algorithm $\mathcal{B}$ does not set the Bad flag in the decoder-check step. By assumption, with probability ε_1, the circuit C^* output by $\mathcal{A}$ satisfies $P_{C^*}(0) \geq 1/2 + \varepsilon_2'$. This means that $\Pr[C^*(\mathsf{ct}_i) = m_i] \geq 1/2 + \varepsilon_2'$. Suppose this is the case. Then, appealing to Hoeffding's inequality, we have that

$$\Pr[N < (1/2 + \varepsilon' + \delta)\xi] = \Pr[(1/2 + \varepsilon_2')\xi - N > \delta\xi] \leq 2^{-\Omega(\delta^2\xi)} = \mathsf{negl}(\lambda).$$

  Thus, if $P_{C^*}(0) \geq 1/2 + \varepsilon_2'$, then $\mathcal{B}$ does not set the Bad flag with $1 - \mathsf{negl}(\lambda)$ probability. Correspondingly, this means that with probability at least $\varepsilon_1 - \mathsf{negl}(\lambda)$, $\mathcal{B}$ does not set the Bad flag.
- Next, we show that if $\mathcal{B}$ does not set Bad, then $P_{C^*}(0) \geq 1/2 + \varepsilon'$. To see this, suppose that $P_{C^*}(0) < 1/2 + \varepsilon'$. We show that $\mathcal{B}$ sets the Bad flag with overwhelming probability in this case. By construction, $\Pr[C^*(\mathsf{ct}_i) = m_i] = P_{C^*}(0)$. We appeal to Hoeffding's inequality to conclude that in this case,

$$\Pr[N > (1/2 + \varepsilon' + \delta)\xi] = \Pr[N - (1/2 + \varepsilon')\xi > \delta\xi] \leq 2^{-\Omega(\delta^2\xi)} = \mathsf{negl}(\lambda).$$

Thus, in this case, $\mathcal{B}$ sets the Bad flag with overwhelming probability. Equivalently, if $\mathcal{B}$ does not set the Bad flag, then with overwhelming probability, it must be the case that $P_{C^*}(0) \geq 1/2 + \varepsilon'$.

- By construction, if $\mathcal{B}$ does not set the Bad flag, then it outputs b with probability $P_{C^*}(0)$.

Combining the above, we conclude that the distinguishing advantage of $\mathcal{B}$ is $1/2 + \varepsilon_1\varepsilon_2'/2 - \mathsf{negl}(\lambda)$, and $\varepsilon_1\varepsilon_2'/2$ is non-negligible, which concludes the proof. □

Lemma C.7. *If Π_{HFE} is secure, then $\Pr[f(x^*) \neq 1] = \mathsf{negl}(\lambda)$ where f is the function chosen by $\mathcal{A}$ at the beginning of the unremovability game and x^* is the attribute that $\mathcal{A}$ outputs at the end of the unremovability game.*

Proof. Suppose there exists an efficient and ε-admissible adversary $\mathcal{A}$ that commits to some function f and outputs an attribute x^* where $f(x^*) \neq 1$ with non-negligible probability ε_1. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that breaks security of Π_{HFE}:

- **Simulating the unremovability game:** Algorithm $\mathcal{B}$ simulates the unremovability game using the same procedure as in the proof of Lemma C.6. At the end of the simulation, $\mathcal{A}$ outputs an attribute x^*, two messages $m_0, m_1 \in \mathcal{M}$, and a circuit $C^* \colon \mathcal{CT} \to \mathcal{M} \cup \{\perp\}$.
- **Good decoder check:** Let $\delta = \varepsilon/4$, $\varepsilon' = \varepsilon/2$, and $\xi = \lambda/\delta^2$. Algorithm $\mathcal{B}$ samples $b_i \leftarrow \{0,1\}$, sets $m_i \leftarrow m_{b_i}$, and computes $\mathsf{ct}_i \leftarrow \mathsf{HFE.Enc}(\mathsf{mpk}, (x^*, m_i, 1))$ for all $i \in [\xi]$. Let N be the number of indices $i \in [\xi]$ where $C^*(\mathsf{ct}_i) = m_i$. If $N < (1/2 + \varepsilon' + \delta)\xi$, then $\mathcal{B}$ sets the Bad_1 flag.
- **Output computation:** If $f(x^*) = 1$, then algorithm $\mathcal{B}$ sets the Bad_2 flag. Otherwise, algorithm $\mathcal{B}$ samples two random messages $m_0, m_1 \leftarrow \mathcal{M}$ and submits the pair $(x, m_0, 1)$ and $(x, m_1, 1)$ as its challenge query and obtains a ciphertext ct^*. It computes $m' \leftarrow C^*(\mathsf{ct}^*)$ and outputs 0 if $m' = m_0$ and 1 if $m' = m_1$. If $m' \neq m_0$ and $m' \neq m_1$, or the Bad_1 or Bad_2 flags have been set, then algorithm $\mathcal{B}$ outputs a random bit.

Algorithm $\mathcal{B}$ is admissible since it only makes a challenge query when $f(x^*) \neq 1$, in which case we have that for all $\tau \in \mathcal{T}$ and $m \in \mathcal{M}$, $h_\tau(g_f(x, m, 1)) = h_\tau(0^\ell, 0^n)$. Since $\mathcal{B}$ perfectly simulates the unremovability game for $\mathcal{A}$, it follows that with probability ε_1, $\mathcal{A}$ outputs a circuit C^* and an attribute x^* where $f(x^*) \neq 1$. It suffices to only consider this case (since otherwise, $\mathcal{B}$ outputs $\perp$).

- Since $\mathcal{A}$ is ε-admissible, $\Pr[C^*(\mathsf{ct}_i) = m_i] \geq 1/2 + \varepsilon$ where $\varepsilon = 1/\mathsf{poly}(\lambda)$. Using an analogous argument as that in the proof of Lemma C.6, algorithm $\mathcal{B}$ will set Bad_1 with negligible probability.
- Using a similar argument as in the proof of Lemma C.6, conditioned on the Bad_1 flag not being set, algorithm $\mathcal{B}$ will output the correct value of b (where b is the challenge bit sampled by the Π_{HFE} challenger) with probability at least $1/2 + \varepsilon' - \mathsf{negl}(\lambda)$.

By assumption, $f(x^*) \neq 1$ with probability ε_1. Thus, algorithm $\mathcal{B}$ outputs b with probability at least $1/2 + \varepsilon_1\varepsilon' - \mathsf{negl}(\lambda)$, and $\varepsilon_1\varepsilon' = \varepsilon_1\varepsilon/2$ is non-negligible. □

Lemma C.8. *If Π_{HFE} is secure, then $P_{C^*}(1^\ell) \geq 1/2 + \varepsilon - \mathsf{negl}(\lambda)$.*

Proof. Suppose that with non-negligible probability ε_1, algorithm $\mathcal{A}$ outputs a circuit C^* where $P_{C^*}(1^\ell) \leq 1/2 + \varepsilon - \varepsilon_2$ for some $\varepsilon_2 = 1/\mathsf{poly}(\lambda)$. We use $\mathcal{A}$ to build an adversary $\mathcal{B}$ that breaks Π_{HFE}:

- **Simulating the unremovability game:** Algorithm $\mathcal{B}$ simulates the unremovability game using the same procedure as the corresponding algorithm in the proof of Lemma C.6. Let f be the function $\mathcal{A}$ chooses at the beginning of the unremovability game. At the end of the simulation, $\mathcal{A}$ outputs an attribute x^*, two messages $m_0, m_1 \in \mathcal{M}$, and a circuit $C^* : \mathcal{CT} \to \mathcal{M} \cup \{\perp\}$.
- **Good decoder check:** If $f(x^*) \neq 1$, then $\mathcal{B}$ sets the Bad_1 flag. Next, let $\delta = \varepsilon_2/4$, $\varepsilon' = \varepsilon - \varepsilon_2/2$, and $\xi = \lambda/\delta^2$. Algorithm $\mathcal{B}$ samples $b_i \leftarrow \{0,1\}$, sets $m_i = m_{b_i}$, and computes $\mathsf{ct}_i \leftarrow \mathsf{HFE.Enc}(\mathsf{mpk}, (1^\ell, m_i, 0))$ for all $i \in [\xi]$. Let N be the number of indices $i \in [\xi]$ where $C^*(\mathsf{ct}_i) = m_i$. If $N > (1/2 + \varepsilon')\xi$, then $\mathcal{B}$ sets the Bad_2 flag.
- **Output computation:** If both the Bad_1 and Bad_2 flag have not been set, then $\mathcal{B}$ chooses a random bit $b' \leftarrow \{0,1\}$, and submits the pair $(1^\ell, m_{b'}, 0)$ and $(x^*, m_{b'}, 1)$ as its challenge query and obtains a challenge ciphertext ct^*. It computes $m' \leftarrow C^*(\mathsf{ct}^*)$ If $m' = m_{b'}$, it outputs 1, and otherwise, it outputs 0. If either Bad_1 or Bad_2 flags have been set, then $\mathcal{B}$ outputs a random bit.

First, we argue that $\mathcal{B}$ is admissible for the hierarchical FE security game. By construction, $\mathcal{B}$ only makes a challenge query if $f(x^*) = 1$. In this case, for all $\tau \in \{0,1\}^\ell$ and $m \in \{0,1\}^n$,

$$h_\tau(g_f(1^\ell, m, 0)) = h_\tau(1^\ell, m) = h_\tau(g_f(x^*, m, 1)).$$

Now, let b be the challenge bit sampled by the Π_{HFE} challenger. We compute the probability that $\mathcal{B}$ outputs b. We focus on the case where $\mathcal{B}$ does not set the Bad_1 and Bad_2 flags (since otherwise, $\mathcal{B}$ outputs a random bit, which does not contribute to its distinguishing advantage). We first show that with probability $\varepsilon_1 - \mathsf{negl}(\lambda)$, algorithm $\mathcal{B}$ does not set Bad_1 or Bad_2.

- By Lemma C.7, then $f(x^*) = 1$ with overwhelming probability. Thus, with overwhelming probability, the Bad_1 flag is not set.
- We show that with probability at least $\varepsilon_1 - \mathsf{negl}(\lambda)$, algorithm $\mathcal{B}$ does not set the Bad_2 flag in the decoder-check step. First, $\mathcal{B}$ perfectly simulates the unforgeability game for $\mathcal{A}$, so by assumption, with probability ε_1, $P_{C^*}(1^\ell) \leq 1/2 + \varepsilon - \varepsilon_2$, or equivalently,

  $$\Pr[C^*(\mathsf{ct}_i) = m_i] \leq 1/2 + \varepsilon - \varepsilon_2 = 1/2 + \varepsilon' - 2\delta$$

  for all $i \in [\xi]$. Again appealing to Hoeffding's inequality, we have that

  $$\Pr[N > (1/2 + \varepsilon')\xi] = \Pr[N - (1/2 + \varepsilon' - 2\delta)\xi > 2\delta\xi] \leq 2^{-\Omega(\delta^2\xi)} = 2^{-\Omega(\lambda)} = \mathsf{negl}(\lambda).$$

  Thus, when $P_{C^*}(1^\ell) \leq 1/2 + \varepsilon - \varepsilon_2$, algorithm $\mathcal{B}$ sets Bad_2 with negligible probability, or equivalently, with probability at least $\varepsilon_1 - \mathsf{negl}(\lambda)$, algorithm $\mathcal{B}$ does not set the Bad_2 flag.

Now, we consider the distinguishing advantage of $\mathcal{B}$ conditioned on Bad_1 and Bad_2 being unset. We consider the two possibilities for the challenger's choice bit b:

- Suppose $b = 0$. In this case, the challenge replies with $\mathsf{HFE.Enc}(1^\ell, m_{b'}, 0)$. In this case, $\Pr[C^*(\mathsf{ct}^*) = m_{b'}] = P_{C^*}(1^\ell)$. Conditioned on Bad_2 not being set, $P_{C^*}(1^\ell) < 1/2 + \varepsilon' + \delta$ with overwhelming probability. To see this, suppose otherwise. This means that for all $i \in [\xi]$, $\Pr[C^*(\mathsf{ct}_i) = m_i] \geq 1/2 + \varepsilon' + \delta$. By Hoeffding's inequality,

$$\Pr[N \leq (1/2 + \varepsilon')\xi] = \Pr[(1/2 + \varepsilon' + \delta)\xi - N \geq \delta\xi] \leq 2^{-\Omega(\delta^2\xi)} = 2^{-\Omega(\lambda)} = \mathsf{negl}(\lambda).$$

  Thus, with overwhelming probability, if Bad_2 has not been set, then $P_{C^*}(1^\ell) < 1/2 + \varepsilon' + \delta$, and correspondingly, in this case, $\Pr[C^*(\mathsf{ct}^*) = m_{b'}] < 1/2 + \varepsilon' + \delta$. Thus, in this case, $\mathcal{B}$ will output 0 with probability at least

$$1 - (1/2 + \varepsilon' + \delta) = 1/2 - \varepsilon' - \delta = 1/2 - \varepsilon + \varepsilon_2/4.$$

- Suppose $b = 1$. In this case, the challenger replies with $\mathsf{HFE.Enc}(x^*, m_{b'}, 1)$. Since $\mathcal{A}$ is ε-admissible, it follows that $\Pr[C^*(\mathsf{ct}^*) = m_{b'}] \geq 1/2 + \varepsilon$. In this case, $\mathcal{B}$ outputs 1 with probability $1/2 + \varepsilon$.

Since the challenger samples b uniformly at random, we have that conditioned on Bad_1 and Bad_2 not being set,

$$\Pr[\mathcal{B} \text{ outputs } b] = \frac{1}{2}\left(\frac{1}{2} - \varepsilon + \frac{\varepsilon_2}{4}\right) + \frac{1}{2}\left(\frac{1}{2} + \varepsilon\right) - \mathsf{negl}(\lambda) = \frac{1}{2} + \frac{\varepsilon_2}{8} - \mathsf{negl}(\lambda).$$

The overall distinguishing advantage of $\mathcal{B}$ is then $1/2 + \varepsilon_1\varepsilon_2/8 - \mathsf{negl}(\lambda)$, and $\varepsilon_1\varepsilon_2/8$ is non-negligible, which concludes the proof. □

Lemma C.9. *Let $\mathcal{Q}$ be the set of identities the adversary $\mathcal{A}$ queries in the unremovability game. If Π_{HFE} is secure, then for any $x, y \in [0, 2^\ell - 1]$ where $x < y$ and $[x + 1, y] \cap \mathcal{Q} = \varnothing$, then $|P_{C^*}(x) - P_{C^*}(y)| = \mathsf{negl}(\lambda)$.*

Proof. This proof follows a single structure as the proof of Lemma C.8. Namely, suppose with non-negligible probability ε_1, algorithm $\mathcal{A}$ outputs a circuit C^* where $|P_{C^*}(x) - P_{C^*}(y)| = \varepsilon_2 = 1/\mathsf{poly}(\lambda)$. Without loss of generality, suppose that $P_{C^*}(x) < P_{C^*}(y)$, in which case our assumption becomes $P_{C^*}(y) - P_{C^*}(x) = \varepsilon_2$. The case where $P_{C^*}(x) > P_{C^*}(y)$ is analogous. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that breaks Π_{HFE}:

- **Simulating the unremovability game:** Algorithm $\mathcal{B}$ simulates the unremovability game using the same procedure as the corresponding algorithm in the proof of Lemma C.6. Let f be the function $\mathcal{A}$ chooses at the beginning of the unremovability game. Let f be the function $\mathcal{A}$ chooses at the beginning of the unremovability game. At the end of the simulation, $\mathcal{A}$ outputs an attribute x^*, two messages $m_0, m_1 \in \mathcal{M}$, and a circuit $C^* \colon \mathcal{CT} \to \mathcal{M} \cup \{\perp\}$.

- **Good decoder check:** Let $\delta = \varepsilon_2/8$ and let $\xi = \lambda/\delta^2$. Algorithm $\mathcal{B}$ samples $b_i \leftarrow \{0, 1\}$, sets $m_i = m_{b_i}$, and computes $\mathsf{ct}_i \leftarrow \mathsf{HFE.Enc}(\mathsf{mpk}, (x, m_i, 0))$ for all $i \in [\xi]$. Let N_x be the number of indices $i \in [\xi]$ where $C^*(\mathsf{ct}_i) = m_i$. Algorithm $\mathcal{B}$ then repeats this procedure by sampling a fresh set of $b'_i \leftarrow \{0, 1\}$, sets $m'_i = m_{b'_i}$, and computes ciphertexts $\mathsf{ct}'_i \leftarrow \mathsf{HFE.Enc}(\mathsf{mpk}, (y, m'_i, 0))$, and sets N_y to be the number of indices $i \in [\xi]$ where $C^*(\mathsf{ct}'_i) = m'_i$. If $N_y - N_x \leq (\varepsilon_2/2)\xi$, algorithm $\mathcal{B}$ sets the Bad flag.

- **Output computation:** If $\mathcal{B}$ has not set the Bad flag, then it samples a random bit $b' \leftarrow \{0,1\}$ and submits the pair $(x, m_{b'}, 0)$ and $(y, m_{b'}, 0)$ as its challenge query and obtains the challenge ciphertext ct^*. It outputs 1 if $C^*(\mathsf{ct}^*) = m_{b'}$ and 0 otherwise.

First, we argue that $\mathcal{B}$ is admissible. Take any $\tau \in \mathcal{Q}$. Suppose $x \leq \tau$, which means by assumption, $y \leq \tau$. Then, for all messages $m \in \{0,1\}^n$,

$$h_\tau(g_f(x, m, 0)) = h_\tau(x, m) = (0^\ell, 0^n) = h_\tau(y, m) = h_\tau(g_f(y, m, 0)). \tag{C.3}$$

Similarly, suppose $y > x > \tau$. Then, for all messages $m \in \{0,1\}^n$,

$$h_\tau(g_f(x, m, 0)) = h_\tau(x, m) = (1^\ell, m) = h_\tau(y, m) = h_\tau(g_f(y, m, 0)). \tag{C.4}$$

Let $b \in \{0,1\}$ be the challenge bit chosen by the Π_{HFE} challenger. We compute the probability that $\mathcal{B}$ outputs b conditioned on the Bad flag not being set. We consider the two possibilities:

- Suppose $b = 0$. Then, $\mathsf{ct}^* \leftarrow \mathsf{HFE.Enc}(\mathsf{mpk}, (x, m_{b'}, 0))$. In this case, $\mathcal{B}$ outputs 0 with probability $1 - P_{C^*}(x)$.
- Suppose $b = 1$. Then, $\mathsf{ct}^* \leftarrow \mathsf{HFE.Enc}(\mathsf{mpk}, (y, m_{b'}, 0))$. In this case, $\mathcal{B}$ outputs 1 with probability $P_{C^*}(y)$.

The distinguishing advantage (conditioned on Bad not being set) is then given by

$$\frac{1}{2}(1 - P_{C^*}(x)) + \frac{1}{2}P_{C^*}(y) = \frac{1}{2} + \frac{1}{2}(P_{C^*}(y) - P_{C^*}(x)). \tag{C.5}$$

Next, we note by Hoeffding's inequality that the following two events occur with overwhelming probability (over the choice of b_i and b'_i):

$$|N_x - P_{C^*}(x)\xi| \leq (\varepsilon_2/8)\xi \quad \text{and} \quad |N_y - P_{C^*}(y)\xi| \leq (\varepsilon_2/8)\xi.$$

Moreover, by the triangle inequality, this means that

$$\begin{aligned}(\varepsilon_2/4)\xi &\geq |N_x - P_{C^*}(x)\xi| + |N_y - P_{C^*}(y)\xi| \\ &\geq |N_x - N_y - P_{C^*}(x)\xi + P_{C^*}(y)\xi| \\ &\geq \big|\, |N_x - N_y| - |P_{C^*}(x)\xi - P_{C^*}(y)\xi| \,\big| \end{aligned} \tag{C.6}$$

To conclude the proof, we need to show that with noticeable probability, $\mathcal{B}$ does not set the Bad flag, and moreover, conditioned on $\mathcal{B}$ not setting the Bad flag, the distinguishing advantage in Eq. (C.5) is non-negligible.

- First, $\mathcal{B}$ perfectly simulates the unremovability game for $\mathcal{A}$, so with probability ε_1, $\mathcal{A}$ outputs a circuit C^* such that $P_{C^*}(y) - P_{C^*}(x) \geq \varepsilon_2$. Applying Eq. (C.6), this means that with overwhelming probability (over the choice of b_i and b'_i), $(\varepsilon_2/4)\xi \geq |\varepsilon_2\xi - |N_x - N_y||$. Thus, with overwhelming probability $|N_x - N_y| \geq (\varepsilon_2/2)\xi$, and algorithm $\mathcal{B}$ does not set the Bad flag. Thus, with probability at least $\varepsilon_1 - \mathsf{negl}(\lambda)$, algorithm $\mathcal{B}$ does not set the Bad flag.
- We show that if $\mathcal{B}$ does not set the Bad flag, then $P_{C^*}(y) - P_{C^*}(x)$ is noticeable. Since $\mathcal{B}$ does not set the Bad flag, we have that $|N_x - N_y| \geq (\varepsilon_2/2)\xi$. Again appealing to Eq. (C.6), this means that with overwhelming probability, $|P_{C^*}(x) - P_{C^*}(y)| \geq \varepsilon_2/4$, which is noticeable.

Putting the pieces together, we have that the distinguishing advantage of $\mathcal{B}$ is $1/2 + \varepsilon_1\varepsilon_2/8 - \mathsf{negl}(\lambda)$ and the claim follows. □

The claim now follows from Theorem 4.9: namely, the function Q_{C^*} satisfies the following properties:

- $\Pr[Q_{C^*}(1^\ell) = 1] - \Pr[Q_{C^*}(0^\ell) = 1] \geq \varepsilon - \mathsf{negl}(\lambda)$ (Lemmas C.6 and C.8).
- For any $x, y \in [0, 2^\ell - 1]$ where $[x+1, y] \cap \mathcal{Q} = \varnothing$ queried by the adversary, $|P_{C^*}(x) - P_{C^*}(y)| = \mathsf{negl}(\lambda)$ (Lemma C.9).

Thus, by Theorem 4.9, $\mathsf{QTrace}^{Q_{C^*}}$ with output an element in $\mathcal{Q}$ (i.e., one of the marks queried by the adversary) with overwhelming probability. □

C.4 Instantiations and Extensions

In this section, we describe two possible instantiations of our watermarkable predicate encryption scheme: one secure against bounded collusions based on the existence of public-key encryption, and one secure against unbounded collusions based on indistinguishability obfuscation (and one-way functions). The main ingredient we require to instantiate our construction from Appendix C.2 is a hierarchical functional encryption scheme for general circuits (that supports depth-2 delegations). Brakerski et al. [BCG+17] previously showed that any bounded collusion secure functional encryption scheme for general circuits implies a bounded collusion secure hierarchical FE scheme for general circuits with constant-depth delegations. Such an FE scheme can be built generically from public-key encryption [SS10, GVW12, AV19]. Moreover, Brakerski et al. also shows that any fully collusion resistant FE scheme for general circuits implies a fully collusion resistant hierarchical FE scheme. Such FE schemes can be built from indistinguishability obfuscation [GGH+13, Wat15] or concrete assumptions on multilinear maps [GGHZ16]. We state two of these instantiations below together with their implications on realizing watermarkable predicate encryption schemes:

Theorem C.10 (Bounded Collusion Resistant Hierarchical FE [BCG+17, AV19])**.** *Take any polynomial $q = \mathsf{poly}(\lambda)$. Assuming public-key encryption, there exists a q-bounded collusion resistant hierarchical functional encryption scheme that supports constant-depth delegation for general circuit families.*

Corollary C.11 (Bounded Collusion Resistant Watermarkable Predicate Encryption)**.** *Take any $\varepsilon = 1/\mathsf{poly}(\lambda)$, any fixed polynomials $q, q_{\mathsf{key}}, q_{\mathsf{mark}} = \mathsf{poly}(\lambda)$, and mark space $\mathcal{T} = \{0,1\}^\ell$, where $\ell = \mathsf{poly}(\lambda)$. Assuming public-key encryption, there exists a $(q_{\mathsf{key}}, q_{\mathsf{mark}})$-bounded collusion resistant watermarkable family of predicate encryption schemes with mark space $\mathcal{T}$ that satisfies ε-unremovability. Moreover, the associated predicate encryption scheme is q-bounded collusion resistant and remains secure even in the presence of a malicious watermarking authority.*

Theorem C.12 (Fully Collusion Resistant Hierarchical FE [GGH+13, Wat15, BCG+17])**.** *Assuming indistinguishability obfuscation and one-way functions, there exists a fully collusion resistant hierarchical functional encryption scheme that support any polynomial-depth delegation for general circuit families.*

Corollary C.13 (Fully Collusion Resistant Watermarkable Predicate Encryption)**.** *Take any $\varepsilon = 1/\mathsf{poly}(\lambda)$ and mark space $\mathcal{T} = \{0,1\}^\ell$. Assuming indistinguishability obfuscation and the existence of one-way functions, there exists a fully collusion resistant watermarkable family of*

predicate encryption schemes with mark space $\mathcal{T}$ that provides ε-unremovability and where the associated predicate encryption scheme is fully collusion resistant and secure even in the presence of a malicious watermarking authority.

Watermarking unforgeability in the secret-key setting. Just as we can consider a notion of watermarking unforgeability for signatures (Section 3.4), we can consider a corresponding notion for watermarkable predicate encryption schemes in the secret-marking setting. We can use an analogous approach as the one we described for signatures. Namely, instead of embedding only the mark $\tau \in \mathcal{T}$ in the decryption circuit (associated with a public key mpk), the marking authority marks the circuit with the string $(\tau, \sigma_{\mathsf{mpk},\tau})$, where $\sigma_{\mathsf{mpk},\tau}$ is a signature on (mpk, τ) under the watermarking authority's secret signing key. With this modification, the extraction algorithm will recover (τ, σ_τ) as the embedded data and can check that σ_τ is indeed a valid signature on (mpk, τ). Unforgeability of the resulting watermarking scheme then reduces to unforgeability of the signature scheme.