

Offline Witness Encryption from Witness PRF and Randomized Encoding in CRS model

Tapas Pal and Ratna Dutta

Department of Mathematics, Indian Institute of Technology Kharagpur
Kharagpur-721302, India
`tapas.pal@iitkgp.ac.in, ratna@maths.iitkgp.ernet.in`

Abstract. *Witness pseudorandom functions* (witness PRFs) generate a pseudorandom value corresponding to an instance x of an NP language and the same pseudorandom value can be recomputed if a witness w that x is in the language is known. Zhandry (TCC 2016) introduced the idea of witness PRFs and gave a construction using multilinear maps. Witness PRFs can be interconnected with the recent powerful cryptographic primitive called *witness encryption*. In witness encryption, a message can be encrypted with respect to an instance x of an NP language and a decryptor that knows a witness w corresponding to the instance x can recover the message from the ciphertext. Mostly, witness encryption was constructed using obfuscation or multilinear maps.

In this work, we build (single relation) witness PRFs using a puncturable pseudorandom function and a randomized encoding in common reference string (CRS) model. Next, we propose construction of an *offline witness encryption* having short ciphertexts from a public-key encryption scheme, an extractable witness PRF and a randomized encoding in CRS model. Furthermore, we show how to convert our single relation witness PRF into a *multi-relation* witness PRF and the offline witness encryption into an *offline functional witness encryption* scheme.

Keywords: Witness PRF, offline witness Encryption, randomized encoding.

1 Introduction

Witness PRF. Zhandry [25] generalizes the idea of witness encryption to initiate the study of a relatively modern and rich primitive *witness pseudorandom functions* (w PRFs). The power of w PRFs lie in the fact that it can be used in place of obfuscation to build many cryptographic tools that do not need to hide a programme P completely, like multiparty non-interactive key exchange without trusted setup, poly-many hardcore bits, re-usable witness encryption, Rudich secret sharing for monotone NP language and fully distributed broadcast encryption.

Witness PRF for an NP language L is capable of computing a pseudorandom function F on an input statement x without the knowledge of secret key whenever a valid witness w for $x \in L$ is known and $F(x)$ can not be recognized

in its domain if $x \notin L$, that is there does not exist a witness explaining $x \in L$. More specifically, w PRF first computes a pair of keys (fk, ek) depending on a relation circuit R corresponding to an NP language L where fk is the function secret key and ek is the function evaluation key. We note that $R(x, w) = 1$ if w is a valid witness for $x \in L$; otherwise 0. A user having the secret key fk generates a pseudorandom value $F(\text{fk}, x) \in \mathcal{Y}$ for any input x . The same pseudorandom value can be recovered using $\text{Eval}(\text{ek}, x, w)$ without the secret key fk if a valid witness w for $x \in L$ is known. From the adversary's point of view, the pseudorandom value $F(\text{fk}, x)$ is computationally indistinguishable from a uniformly chosen element in \mathcal{Y} if there does not exist a witness w for $x \in L$. On the other hand, if $x \in L$ then, an adversary distinguishing $F(\text{fk}, x)$ from a random element in \mathcal{Y} is to mean that there exists an efficient extractor that can be used to obtain a witness for $x \in L$. A w PRF processing this security assumption is called an *extractable witness pseudorandom function* (extractable w PRF). Another variant of w PRF is termed as *multi-relation w PRF* where one can generate different function evaluation keys associated with many relation circuits of the same language L .

Witness encryption. Garg et al. [15] introduced the notion of *witness encryption* (WE) which is closely related to w PRFs. In a plain public-key encryption (PKE), we encrypt data using a public key and decryption is possible if the corresponding secret key is known. WE enables us to encrypt a message with respect to an instance x of an NP language L . Only a witness holder can recover the original message from the ciphertext if he has a valid witness w for $x \in L$. The notion of *Functional witness encryption* was introduced by Boyle et al. [7] where a decrypter can only learn a function of the message if a valid witness for the instance is known. They have established the equivalence between functional WE and differing inputs obfuscation.

Witness encryption consists of only two algorithms encryption and decryption. As a result, all the heavy-duty parts have been included in the encryption algorithm that makes WE more inefficient to use in small devices. Abusalah et al. [1] added a *Setup* phase which processes necessary tools to produce public parameters for encryption and decryption. Witness encryption with additional setup phase is called *offline witness encryption* (OWE).

Motivation. WEs and w PRFs are relatively new cryptographic primitives mostly built from either multilinear maps or obfuscation. As a result these primitives are experiencing inefficiency due to the existing noisy multilinear maps and impracticality of obfuscation. We aim to construct more efficient w PRF and OWE for any class of NP languages. Zhandry [25] used multilinear maps to construct w PRFs which are instance dependence and multilinearity level increases with respect to the size of relation circuits. The recent line of attacks on multilinear maps [12,10,6,16] is a threat to the cryptosystems where security is based on complexity assumptions related to multilinear maps. It was mentioned in [25] that w PRFs can be obtained from obfuscation but there was no explicit construction. In the same work, w PRFs were used to replace obfuscation from many cryptographic tools but those applications may not be fruitful in the practical

sense as the existing multilinear maps are only approximate and encountered many non-trivial attacks.

The OWE scheme of [1] was realized using ElGamal public-key encryption and Gorth-Sahai proofs (GS-proofs) [20]. We note that GS-proofs are efficient non-interactive witness-indistinguishable proofs for some specific languages involving pairing product equations, multi-scaler multiplication equations or quadratic equations over some groups. The ElGamal ciphertexts can be represented in a way to get a set of pairing product equations so that a statistical simulation-sound non-interactive zero-knowledge (SSS-NIZK) proof can be ensured using the GS-proofs for those equations. Therefore, for practical use of the OWE scheme of [1], we need to carefully choose the PKE scheme so that a SSS-NIZK proof can be achieved through the GS-proofs. Otherwise, we need to use the transformation of [4,14] to achieve a SSS-NIZK proof that involves indistinguishability obfuscation and it may unnecessarily increase the size of OWE-ciphertexts. More specifically, for a given circuit C , an NIZK proof [19] for circuit satisfiability problem requires a size of $O(|C|k)$ where $O(k)$ is the size of common reference string and $|C|$ denotes size of circuit C . Therefore, the SSS-NIZK proof is of size at least linear in the size of the encryption circuit of the underlying PKE. We aim to get an OWE with relatively short ciphertexts where we do not require to generate such proofs and can use any PKE schemes as far as our requirement. Getting an efficient encryption algorithm producing short ciphertexts is a desirable property while constructing OWE so that one can use it in other cryptographic constructions.

Our contribution. In this work we construct a single relation w PRF (Section 3) using a puncturable pseudorandom function and sub-exponentially secure randomized encoding scheme in CRS model. Our approach is to use the puncturable programming technique of [24] and incorporate the idea of getting obfuscation from randomized encoding (RE) scheme in common reference string (CRS) model [23]. A sub-exponentially secure randomized encoding scheme in CRS model can be achieved from a sub-exponentially secure public key functional encryption scheme and learning with error assumptions with sub-exponential hardness [23]. The security proof of our w PRF is independent of instances and does not rely on non-standard assumptions. We turn our single relation w PRF into a multi-relation w PRF (Remark 2) where one can use the scheme with a class of relations related to an NP language.

Furthermore, we build an OWE scheme (Section 4) utilizing an extractable w PRF. We replace SSS-NIZK by w PRF from the construction of [1] to reduce the size of ciphertext by at least linear to the size of encryption circuit of the elementary PKE scheme required in the building block. More precisely, our scheme is based on a public-key encryption, an extractable w PRF and employs a sub-exponentially secure randomized encoding scheme in CRS model. Consequently, the ciphertexts contain a pseudorandom string of fixed size instead of a SSS-NIZK proof. Using the same blueprint of [1], our OWE can be turned into an offline functional witness encryption (OFWE) scheme (Remark 3) where decryption releases a function of a message and witness as output. Inherently, our

OFWE also possesses short ciphertext as compared to that of [1]. Unfortunately, the only extractable w PRF is known to be constructed from multilinear maps [25]. Our construction of OWE would be more interesting if w PRF with extracting feature can be built from standard assumptions without multilinear maps which is still an open problem.

Previous work. Zhandry formalized the notion of witness PRF [25] and constructed it using *subset-sum encoding scheme*. The notion of subset-sum encoding was instantiated from multilinear maps [13,11] and the security is based on *multilinear subset-sum Diffie-Hellman assumption*. The w PRF obtained from subset-sum encoding cannot immediately compute a pseudorandom value for an instance x of NP language L , rather a reduction procedure is followed where x is converted into a subset-sum instance. The reduced subset-sum instance depends on L and the size of instance x . The security of the w PRF is also based on multilinear subset-sum Diffie-Hellman assumption and hardness of the assumption is shown in a generic multilinear map model [25].

Abusalah et al. gave a construction of OWE using plain public-key encryption, SSS-NIZK proof system and obfuscation. We note that SSS-NIZK can be obtained from one-way functions and indistinguishable obfuscation [4,14]. They built an OFWE utilizing the same primitives.

2 Preliminaries

We use the notations in Table 1 throughout this paper. We take \perp as a distinguishing symbol. Now, we go through the definition of randomized encoding

$a \leftarrow A$	a is an output of the procedure A .
$a \xleftarrow{\$} X$	a is chosen uniformly at random from set X .
negligible function	$\mu : \mathbb{N} \rightarrow \mathbb{R}$ is a negligible function if $\mu(n) \leq \frac{1}{p(n)}$ holds for every polynomial $p(\cdot)$ and all sufficiently large $n \in \mathbb{N}$.
$(\lambda_0, S(\cdot))$ -indistinguishability	Two ensembles $\{X_\lambda\}$ and $\{Y_\lambda\}$ are $(\lambda_0, S(\cdot))$ -indistinguishable means $ \Pr[x \xleftarrow{\$} X_\lambda : \mathcal{D}(x) = 1] - \Pr[y \xleftarrow{\$} Y_\lambda : \mathcal{D}(y) = 1] \leq \frac{1}{S(\lambda)}$ for any security parameter $\lambda > \lambda_0$ and every $S(\lambda)$ -size distinguisher \mathcal{D} , $S : \mathbb{N} \rightarrow \mathbb{N}$.
δ -sub-exponential indistinguishability	Two ensembles $\{X_\lambda\}$ and $\{Y_\lambda\}$ are δ -sub-exponential indistinguishable means $ \Pr[x \xleftarrow{\$} X_\lambda : \mathcal{D}(x) = 1] - \Pr[y \xleftarrow{\$} Y_\lambda : \mathcal{D}(y) = 1] < \delta(\lambda)^{\Omega(1)}$, for any security parameter λ and every poly-size distinguisher \mathcal{D} , where $\delta(\lambda) < 2^{\lambda^\epsilon}$, $0 < \epsilon < 1$.
$\text{Expt}(1^\lambda, 0) \approx_\delta \text{Expt}(1^\lambda, 1)$	For any polynomial size distinguisher \mathcal{D} , the advantage $\Delta = \Pr[\mathcal{D}(\text{Expt}(1^\lambda, 0)) = 1] - \Pr[\mathcal{D}(\text{Expt}(1^\lambda, 1)) = 1] $ is bounded by δ .

Table 1. Notations

(RE) scheme in CRS model and briefly discuss how one can get indistinguishability obfuscation from RE [23]. The rest of the definitions related to this work can be found in Appendix A.

2.1 Randomized Encoding Scheme in CRS Model

Randomized encoding was introduced by Ishai and Kushilevitz [21] to encode a complex deterministic function Π along with an input x through an encoding algorithm whose output distribution $\widehat{\Pi}(x)$ can efficiently compute $\Pi(x)$ and reveals no information beyond $\Pi(x)$. Recently, Lin et al. [23] studied randomized encoding scheme in both plain model and common reference string (CRS) model with compactness and sub-linear compactness of the size of encodings.

Definition 1 (Randomized encoding schemes in CRS model). A randomized encoding scheme $\text{RE} = (\text{Setup}, \text{Enc}, \text{Eval})$ in CRS model for a class of Turing machines $\{\mathcal{M}_\lambda\}$ where Setup and Enc are randomized algorithms and Eval is a deterministic algorithm, performs as follows:

- $(\text{crs}, pk) \leftarrow \text{RE.Setup}(1^\lambda, 1^m, 1^n, 1^T, 1^l)$: A trusted third party takes as input a security parameter λ , a machine size bound m , input length bound n , time bound T and output length l . It outputs a common reference string crs and a public key pk .
- $\widehat{\Pi}_x \leftarrow \text{RE.Enc}(pk, \Pi, x)$: The encoding algorithm uses a public key pk , a Turing machine $\Pi \in \mathcal{M}_\lambda$ together with an input x and outputs an encoding $\widehat{\Pi}_x$.
- $y \leftarrow \text{RE.Eval}(\widehat{\Pi}_x, \text{crs})$: An evaluator makes use of an encoding $\widehat{\Pi}_x$ and a common reference string crs and outputs some y .

Correctness: For any $\lambda \in \mathbb{N}$, $m(\lambda), n(\lambda), T(\lambda), l(\lambda) \in \mathbb{N}$, Turing machine $\Pi \in \mathcal{M}_\lambda$ and input x with $|\Pi| \leq m$, $|x| \leq n$ and $|\Pi^T(x)| \leq l$, we have that

$$\Pr \left[\begin{array}{l} \text{RE.Eval}(\widehat{\Pi}_x, \text{crs}) = \Pi^T(x) : (\text{crs}, pk) \leftarrow \text{RE.Setup}(1^\lambda, 1^m, 1^n, 1^T, 1^l) \\ \widehat{\Pi}_x \leftarrow \text{RE.Enc}(pk, \Pi, x) \end{array} \right] = 1$$

Here $\Pi^T(x)$ denotes the output of the Turing machine Π on input x when run in at most T steps.

Definition 2 ($(\lambda_0, S(\cdot))$ -simulation security of randomized encoding in CRS model). We say that a randomized encoding scheme RE for a class of Turing machines $\{\mathcal{M}_\lambda\}$ in CRS model is $(\lambda_0, S(\cdot))$ -simulation secure if there exists a PPT algorithm Sim and a constant c such that for every $\{\Pi, x, m, n, l, T\}$ where $\Pi \in \mathcal{M}_\lambda$ and $|\Pi|, |x|, m, n, l, T \leq B(\lambda)$ for some polynomial B , the ensembles

$$\left\{ (\text{crs}, pk, \widehat{\Pi}_x) : (\text{crs}, pk) \leftarrow \text{RE.Setup}(1^\lambda, 1^m, 1^n, 1^T, 1^l), \widehat{\Pi}_x \leftarrow \text{RE.Enc}(pk, \Pi, x) \right\}$$

and $\left\{ (\text{crs}, pk, \widehat{\Pi}_x) : (\text{crs}, pk, \widehat{\Pi}_x) \leftarrow \text{Sim}(1^\lambda, \Pi^T(x), 1^{|\Pi|}, 1^{|x|}, 1^m, 1^n, 1^T, 1^l) \right\}$

are $(\lambda_0, S'(\lambda))$ -indistinguishable (see Table 1), with $S'(\lambda) = S(\lambda) - B(\lambda)^c$ for all $\lambda \in \mathbb{N}$. The RE is said to be δ -simulation secure for some specific negligible function $\delta(\cdot)$ if $S'(\lambda)$ is greater than $\delta(\lambda)^{\Omega(1)}$. Also, we say that RE is δ -sub-exponential simulation secure if $\delta(\lambda) < 2^{-\lambda^\epsilon}$, $0 < \epsilon < 1$.

<p>Hardwired: $\vec{\Pi}[pk_1, C, \epsilon, \alpha], \vec{\text{crs}}$ Input: an input $z = (z_1 z_2 \dots z_n)$</p> <ol style="list-style-type: none"> 1. $\vec{\Pi} \leftarrow \vec{\Pi}[pk_1, C, \epsilon, \alpha], i \leftarrow 0$ 2. while $i < n$ do 3. $(\vec{\Pi}[pk_{i+2}, C, z_1 z_2 \dots z_i 0, \alpha_0^{i+1}], \vec{\Pi}[pk_{i+2}, C, z_1 z_2 \dots z_i 1, \alpha_1^{i+1}]) \leftarrow \text{RE.Eval}(\vec{\Pi}, \text{crs}_i)$ 4. $\vec{\Pi} \leftarrow \vec{\Pi}[pk_{i+2}, C, z_1 z_2 \dots z_i z_{i+1}, \alpha_{z_{i+1}}^{i+1}]$ 5. end do 6. return $\text{RE.Eval}(\vec{\Pi}, \text{crs}_i)$
--

Fig. 1. The Special Circuit $\mathcal{G}[\vec{\Pi}[pk_1, C, \epsilon, \alpha], \vec{\text{crs}}]$

Remark 1 In [23], an $i\mathcal{O}$ is instantiated from a sub-exponentially secure and sub-linearly compact RE scheme in CRS model (Definition 21) and a sub-exponentially secure pseudorandom generator (PRG). They followed the technique of GGM construction [17] of building a PRF from a PRG using a tree. To get an $i\mathcal{O}$, the PRG in the GGM construction is replaced with a sub-exponentially secure sub-linear compact RE in CRS model. Let $\{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ be a circuit class with maximum size S , input size n , output size l and the running time bound T . The obfuscation procedure for a circuit $C \in \mathcal{C}_\lambda$ works as follows:

- We generate $(\text{crs}_i, pk_i) \leftarrow \text{RE.Setup}(1^\lambda, 1^S, 1^n, 1^T, 1^l)$, for $i \in \{0, 1, \dots, n\}$, where crs_i is a common reference string and pk_i is an encoding key. Let $\vec{\text{crs}} = \{\text{crs}_i\}_{i=0}^n$, $\vec{pk}_i = \{pk_j\}_{j=i}^n$.
- We construct an input less Turing machine $\Pi[\vec{pk}_{i+1}, C, z, \alpha_{z_i}^i]$ where hard-coded entities are $\vec{pk}_{i+1}, C, z = z_1 z_2 \dots z_i \in \{0, 1\}^i$ and a string $\alpha_{z_i}^i \in \{0, 1\}^{2p(\lambda, i)}$ (p being a polynomial depending on λ, i)¹ for all $i \in \{0, 1, \dots, n-1\}$. When $i = 0$, z is the null string ϵ and $\alpha_{z_i}^i$ is a random string $\alpha \xleftarrow{\$} \{0, 1\}^{2p(\lambda, 0)}$. The Turing machine $\Pi[\vec{pk}_1, C, \epsilon, \alpha]$ computes randomized encodings of $\Pi[\vec{pk}_2, C, 0, \alpha_0^1]$ and $\Pi[\vec{pk}_2, C, 1, \alpha_1^1]$ where $(\alpha_0^1, \alpha_1^1) \leftarrow \text{PRG}(\alpha)$ with $|\alpha_0^1| = |\alpha_1^1| = 2p(\lambda, 1)$, PRG being a sub-exponentially secure pseudorandom generator. To be more specific, the Turing machine $\Pi[\vec{pk}_1, C, \epsilon, \alpha]$ first generates $(\alpha_0^1, \alpha_1^1) \leftarrow \text{PRG}(\alpha)$ and uses the randomness α_0^1 to compute encoding $\vec{\Pi}[\vec{pk}_2, C, 0, \alpha_0^1] \leftarrow \text{RE.Enc}(pk_1, \Pi[\vec{pk}_2, C, 0, \alpha_0^1], \epsilon)$ and the randomness α_1^1 to compute the encoding $\vec{\Pi}[\vec{pk}_2, C, 1, \alpha_1^1] \leftarrow \text{RE.Enc}(pk_1, \Pi[\vec{pk}_2, C, 1, \alpha_1^1], \epsilon)$. More generally, the Turing machine $\Pi[\vec{pk}_{i+1}, C, z, \alpha_{z_i}^i]$ computes randomized encodings $\vec{\Pi}[\vec{pk}_{i+2}, C, z0, \alpha_0^{i+1}] \leftarrow \text{RE.Enc}(pk_{i+1}, \Pi[\vec{pk}_{i+2}, C, z0, \alpha_0^{i+1}], \epsilon)$ and $\vec{\Pi}[\vec{pk}_{i+2}, C, z1, \alpha_1^{i+1}] \leftarrow \text{RE.Enc}(pk_{i+1}, \Pi[\vec{pk}_{i+2}, C, z1, \alpha_1^{i+1}], \epsilon)$, where $(\alpha_0^{i+1}, \alpha_1^{i+1}) \leftarrow \text{PRG}(\alpha_{z_i}^i)$ for $i \in \{1, 2, \dots, n-1\}$. When $i = n$, the machine $\Pi[\vec{pk}_{i+1}, C, z, \alpha_{z_i}^i]$ outputs $C(z)$. We denote the class of all such Turing machines associated with the class of circuits $\{\mathcal{C}_\lambda\}$ as $\{\mathcal{M}_\lambda\}$.

¹ For every $\lambda \in \mathbb{N}, i \leq 2^\lambda$, $p(\lambda, i) = p(\lambda, i-1) + (2d\lambda)^{1/\epsilon}$ and $p(\lambda, -1) = \lambda$ where ϵ is a constant associated with the sub-exponential security of PRG, $d > 0$ is any constant strictly greater than c and the constant c represents the security loss associated with the indistinguishability security of RE (section 4, [23]).

- We compute an encoding $\tilde{\Pi}[\vec{pk}_1, C, \epsilon, \alpha] \leftarrow \text{RE.Enc}(pk_0, \Pi[\vec{pk}_1, C, \epsilon, \alpha], \epsilon)$. Next, we construct the special circuit $\mathcal{G}[\tilde{\Pi}[\vec{pk}_1, C, \epsilon, \alpha], \vec{\text{crs}}]$ as described in Figure 1 which takes input an n bit string $z = z_1 z_2 \cdots z_n$. For each $i \in \{0, 1, \dots, n-1\}$, the circuit recursively computes $\text{RE.Eval}(\tilde{\Pi}[\vec{pk}_{i+1}, C, z_1 z_2 \cdots z_i, \alpha_{z_i}^i], \text{crs}_i)$ which by correctness of RE, is equal to the output of the Turing machine $\Pi[\vec{pk}_{i+1}, C, z_1 z_2 \cdots z_i, \alpha_{z_i}^i]$ i.e. two randomized encodings $\tilde{\Pi}[\vec{pk}_{i+2}, C, z_1 z_2 \cdots z_i 0, \alpha_0^{i+1}]$ and $\tilde{\Pi}[\vec{pk}_{i+2}, C, z_1 z_2 \cdots z_i 1, \alpha_1^{i+1}]$ (as in line 3 of Figure 1). Finally, the circuit returns $\text{RE.Eval}(\tilde{\Pi}[\vec{pk}_{n+1}, C, z, \alpha_{z_n}^n], \text{crs}_n)$ which actually is equal to $C(z)$. The obfuscation of the circuit C is $i\mathcal{O}(1^\lambda, C) = \mathcal{G}[\tilde{\Pi}[\vec{pk}_1, C, \epsilon, \alpha], \vec{\text{crs}}]$.
 - To evaluate the circuit C for an input z , we compute $\mathcal{G}[\tilde{\Pi}[\vec{pk}_1, C, \epsilon, \alpha], \vec{\text{crs}}](z)$.
- Lin et al. [23] proved that for any pair of functionally equivalent circuits $C_0, C_1 \in \mathcal{C}_\lambda$, the joint distribution $(\tilde{\Pi}[\vec{pk}_1, C_0, \epsilon, \alpha], \vec{\text{crs}})$ is indistinguishable from $(\tilde{\Pi}[\vec{pk}_1, C_1, \epsilon, \alpha], \vec{\text{crs}})$. In particular, they have shown using the method of induction that for any label $i \in \{0, 1, \dots, n\}$, $z \in \{0, 1\}^i$ the joint distributions $(\tilde{\Pi}[\vec{pk}_{i+1}, C_0, z, \alpha_{z_i}^i], \vec{\text{crs}}_i, \vec{pk}_i)$ and $(\tilde{\Pi}[\vec{pk}_{i+1}, C_1, z, \alpha_{z_i}^i], \vec{\text{crs}}_i, \vec{pk}_i)$ are indistinguishable. The indistinguishability was achieved by the simulation security of the RE scheme as described in the following theorem.

Theorem 1 [23] *Assuming the existence of sub-exponentially secure one-way functions, if there exists a sublinearly compact randomized encoding scheme in the CRS model with sub-exponential simulation security, then there exists an bounded-input indistinguishability obfuscator for Turing machines.*

We stress that $\text{RE.Enc}(pk_0, \Pi[\vec{pk}_1, C, \epsilon, \alpha], \epsilon)$ is actually a ciphertext obtained from the encryption algorithm of underlying PKFE that uses $(\Pi[\vec{pk}_1, C, \epsilon, \alpha], \epsilon, 0^{\lambda+1})$ as the plaintext. The size of the special circuit \mathcal{G} is bounded by $\text{poly}(\lambda, |C|, T)$ and runtime of \mathcal{G} on input z is bounded by $\text{poly}(\lambda, |z|, |C|, T)$. We will use the notation $\mathcal{G}[\tilde{\Pi}[\vec{pk}_1, C, \epsilon, \alpha], \vec{\text{crs}}]$ for obfuscating a circuit C using a randomized encoding scheme in CRS model.

3 Our Witness PRF

Construction 1. We describe our construction of witness PRF ($w\text{PRF}$) that uses a puncturable pseudorandom function $\text{pPRF} = (\text{Gen}, \text{Eval}, \text{Punc})$ with domain $\mathcal{X} = \{0, 1\}^k$ and range \mathcal{Y} and a randomized encoding scheme $\text{RE} = (\text{Setup}, \text{Enc}, \text{Eval})$ which is a bounded input sub-linearly compact randomized encoding scheme in CRS model. Our scheme $w\text{PRF} = (\text{Gen}, \text{F}, \text{Eval})$ for an NP language L with relation circuit $R : \mathcal{X} \times \mathcal{W} \rightarrow \{0, 1\}$, $\mathcal{X} = \{0, 1\}^k$, $\mathcal{W} = \{0, 1\}^{n-k}$ and $|R| \leq s$, is given by the following algorithms.

- $(\text{fk}, \text{ek}) \leftarrow w\text{PRF.Gen}(1^\lambda, R)$: A trusted third party generates a secret function key fk and a public evaluation key ek for a relation R by executing the following steps where λ is a security parameter.

<p>Hardwired: a pPRF key K.</p> <p>Input: an instance $x \in \mathcal{X} = \{0, 1\}^k$ and a witness $w \in \mathcal{W} = \{0, 1\}^{n-k}$.</p> <p>Padding: the circuit is padded to size $\text{pad} = \text{pad}(s, n, \lambda)$, determined in the analysis.</p> <ol style="list-style-type: none"> 1. if $R(x, w) = 1$ then 2. $y \leftarrow \text{pPRF.Eval}(K, x)$. 3. else $y \leftarrow \perp$ 4. end if 5. return y
--

Fig. 2. Evaluation Circuit $E = \text{EC}[K]$

- Choose a pPRF key $K \leftarrow \text{pPRF.Gen}(1^\lambda)$ where $K \in \{0, 1\}^\lambda$.
- Construct the circuit $E = \text{EC}[K] \in \{E_\lambda\}$ as defined in Figure 2. Let the circuit E be of size S with input size n , output size l and T is the runtime bound of the circuit.
- Generate $(\text{crs}_i, pk_i) \leftarrow \text{RE.Setup}(1^\lambda, 1^S, 1^n, 1^T, 1^l)$ for $i \in \{0, 1, \dots, n\}$ where crs_i is a common reference string and pk_i is an encoding key. We define $\vec{\text{crs}} = \{\text{crs}_i\}_{i=0}^n$ and $\vec{\text{pk}}_i = \{\text{pk}_j\}_{j=i}^n$.
- Compute the randomized encoding $\tilde{\Pi}[\vec{\text{pk}}_1, E, \epsilon, \alpha] \leftarrow \text{RE.Enc}(pk_0, \Pi[\vec{\text{pk}}_1, E, \epsilon, \alpha], \epsilon)$ where ϵ is a null string, α is a random binary string and $\Pi[\vec{\text{pk}}_1, E, \epsilon, \alpha]$ is a Turing machine defined in Remark 1.
- Build the special circuit $\mathcal{G}[\tilde{\Pi}[\vec{\text{pk}}_1, E, \epsilon, \alpha], \vec{\text{crs}}]$ as described in Figure 1.
- Set $\text{fk} = K$, $\text{ek} = \mathcal{G}[\tilde{\Pi}[\vec{\text{pk}}_1, E, \epsilon, \alpha], \vec{\text{crs}}]$ and output (fk, ek) . The secret function key fk is sent to a user over a secure channel and the evaluation key ek is made public.
- $y \leftarrow w\text{PRF.F}(\text{fk}, x)$: This algorithm is run by the user who has a secret function key fk and outputs a $w\text{PRF}$ value $y \leftarrow \text{pPRF.Eval}(K, x) \in \mathcal{Y}$ for an instance $x \in \mathcal{X}$ using the secret function key fk as a pPRF key K .
- $w\text{PRF.Eval}(\text{ek}, x, w)$: An evaluator takes a witness $w \in \mathcal{W}$ for $x \in L$ and uses the public evaluation key $\text{ek} = \mathcal{G}[\tilde{\Pi}[\vec{\text{pk}}_1, E, \epsilon, \alpha], \vec{\text{crs}}]$ to get back the $w\text{PRF}$ value as $\mathcal{G}[\tilde{\Pi}[\vec{\text{pk}}_1, E, \epsilon, \alpha], \vec{\text{crs}}](z)$ where $z = (x, w) \in \{0, 1\}^n$.

Correctness. The output of $w\text{PRF.F}$ for an instance x is a pPRF evaluation $y \leftarrow \text{pPRF.Eval}(K, x) \in \mathcal{Y}$ on x using the secret key $K \in \{0, 1\}^\lambda$. On the other hand, for $w\text{PRF.Eval}$ an witness-holder computes $\mathcal{G}[\tilde{\Pi}[\vec{\text{pk}}_1, E, \epsilon, \alpha], \vec{\text{crs}}](z)$ where $z = (x, w)$. By the correctness of randomized encoding scheme as discussed in Remark 1, we have $\mathcal{G}[\tilde{\Pi}[\vec{\text{pk}}_1, E, \epsilon, \alpha], \vec{\text{crs}}](z) = E(x, w)$. The circuit E (Figure 2) on input x, w , first checks whether $R(x, w) = 1$ holds. If this condition is satisfied, then it outputs $y \leftarrow \text{pPRF.Eval}(K, x) \in \mathcal{Y}$ using the hardcoded key K . Therefore a valid witness-holder of $x \in L$ can recompute the $w\text{PRF}$ value $y \in \mathcal{Y}$ associated with x using the witness w and the evaluation key $\text{ek} = \mathcal{G}[\tilde{\Pi}[\vec{\text{pk}}_1, E, \epsilon, \alpha], \vec{\text{crs}}]$. Note that, if w is not valid witness for $x \in L$ then the output of $\mathcal{G}[\tilde{\Pi}[\vec{\text{pk}}_1, E, \epsilon, \alpha], \vec{\text{crs}}](z) = E(x, w)$ is the distinguished symbol \perp . Therefore, our $w\text{PRF}$ follows the correctness property stated in Equation 1, Definition 8.

Padding Parameter. The proof of security relies on the indistinguishability of randomized encodings of the machines $\Pi[\vec{\text{pk}}_1, E, \epsilon, \alpha]$ and $\Pi[\vec{\text{pk}}_1, E^*, \epsilon, \alpha]$ (where

-
1. The adversary \mathcal{A} submits a challenge statement $x^* \in \mathcal{X} \setminus L$.
 2. The challenger generates $(\text{fk}, \text{ek}) \leftarrow w\text{PRF.Gen}(1^\lambda, R)$ as follows and sends ek to \mathcal{A} :
 - 2.1 Chose $K \leftarrow \text{pPRF.Gen}(1^\lambda)$ and set $\text{fk} = K$
 - 2.2 Construct the circuit $E = \text{EC}[K]$ as defined in Figure 2
 - 2.3 Generate $(\text{crs}_i, \text{pk}_i) \leftarrow \text{RE.Setup}(1^\lambda, 1^S, 1^n, 1^T, 1^l)$ for $i \in \{0, 1, \dots, n\}$ where S, n, T, l are the same as in Construction 1 and set $\vec{\text{crs}} = \{\text{crs}_i\}_{i=0}^n$ and $\vec{\text{pk}}_i = \{\text{pk}_j\}_{j=i}^n$
 - 2.4 Build the special circuit $\mathcal{G}[\vec{\Pi}[\vec{\text{pk}}_1, E, \epsilon, \alpha], \vec{\text{crs}}]$ as described in Figure 1 where $\vec{\Pi}[\vec{\text{pk}}_1, E, \epsilon, \alpha] \leftarrow \text{RE.Enc}(\text{pk}_0, \Pi[\vec{\text{pk}}_1, E, \epsilon, \alpha], \epsilon)$ and $\Pi[\vec{\text{pk}}_1, E, \epsilon, \alpha]$ is a Turing machine defined in Remark 1.
 - 2.5 Set $\text{ek} = \mathcal{G}[\vec{\Pi}[\vec{\text{pk}}_1, E, \epsilon, \alpha], \vec{\text{crs}}]$
 3. The challenger computes $y^* \leftarrow w\text{PRF.F}(\text{fk}, x^*) \in \mathcal{Y}$ and sends it to \mathcal{A} .
 4. The adversary \mathcal{A} can make polynomial number of queries for $w\text{PRF.F}$ on some $x \in \mathcal{X} \setminus \{x^*\}$ to the challenger and receives $w\text{PRF.F}(\text{fk}, x)$.
 5. The adversary \mathcal{A} outputs a bit b' .
-

Fig. 3. Hybd_0 associated with our $w\text{PRF}$

E and E^* are defined in Figure 2 and 4 respectively). For this we set $\text{pad} = \max(|E|, |E^*|)$. The circuits E and E^* compute the relation circuit R on an input (x, w) of size n and evaluate a puncturable PRF over the domain $\mathcal{X} = \{0, 1\}^k$ of size 2^k using a hardwired element which is a pPRF key for E or a punctured pPRF key for E^* . Thus, $\text{pad} \leq \text{poly}(\lambda, s, k)$ where s is the size of the relation circuit R .

Efficiency. In this analysis, we discuss the size of $w\text{PRF.F}$ and $w\text{PRF.Eval}$. The size of \mathcal{X} is 2^k and $w\text{PRF.F}$ includes a PRF evaluation over the domain \mathcal{X} . Therefore, size of $w\text{PRF.F}$ is bounded by $\text{poly}(\lambda, k)$. We note that, $w\text{PRF.Eval}$ only runs the circuit $\mathcal{G}[\vec{\Pi}[\vec{\text{pk}}_1, E, \epsilon, \alpha], \vec{\text{crs}}]$ over an input of size n . The running time of $\mathcal{G}[\vec{\Pi}[\vec{\text{pk}}_1, E, \epsilon, \alpha], \vec{\text{crs}}]$ is $\text{poly}(\lambda, n, |E|, T) = \text{poly}(\lambda, n, k, s, T)$ and the size of $\mathcal{G}[\vec{\Pi}[\vec{\text{pk}}_1, E, \epsilon, \alpha], \vec{\text{crs}}]$ is $\text{poly}(\lambda, |E|, T) = \text{poly}(\lambda, k, s, T)$. In particular, the running time and size of $w\text{PRF.Eval}$ are respectively $\text{poly}(\lambda, n, k, s, T)$ and $\text{poly}(\lambda, k, s, T)$. Here we note that the runtime T of the circuit E is bounded by the runtime of the relation R and the runtime of a pPRF evaluation. So, $T \leq T_R + \text{poly}(\lambda, k)$ where T_R is the runtime of the relation circuit R on input (x, w) of size n . Hence, the runtime of $w\text{PRF.Eval}$ is bounded by $\text{poly}(\lambda, n, k, s, T_R)$ and size of $w\text{PRF.Eval}$ is bounded by $\text{poly}(\lambda, k, s, T_R)$.

Theorem 2 *Assume existence of δ -sub-exponentially secure one-way functions. Our construction 1 of $w\text{PRF} = (\text{Gen}, \text{F}, \text{Eval})$ is δ -selectively secure witness PRF if the pPRF is a δ -secure puncturable PRF and the RE is a bounded input sub-linearly compact randomized encoding scheme in CRS model with δ -sub-exponential simulation security for the class of Turing machines $\{\mathcal{M}_\lambda\}$ associated with the circuit class $\{E_\lambda\}$.*

Proof. We prove this by showing that for any non-uniform PPT adversary \mathcal{A} , the distinguishing advantage between the two experiments $\text{Expt}_{\mathcal{A}}^{w\text{PRF}}(1^\lambda, 0)$ and $\text{Expt}_{\mathcal{A}}^{w\text{PRF}}(1^\lambda, 1)$ (Figure 11) is negligible. Consider the following hybrid games:

Hybd₀ This is the standard experiment $\text{Expt}_{\mathcal{A}}^{w\text{PRF}}(1^\lambda, 0)$ described in Figure 3.

<p>Hardwired: a punctured key $K\{x^*\}$. Input: an instance $x \in \mathcal{X}$ and a witness $w \in \mathcal{W}$</p> <ol style="list-style-type: none"> 1. if $R(x, w) = 1$ then 2. if $x = x^*$ then 3. return y^* 4. else return $\text{pPRF.Eval}(K\{x^*\}, x)$ 5. end if 6. return \perp

Fig. 4. Evaluation Circuit $E^* = \text{EC}[K\{x^*\}]$

Hybd₁ In this hybrid game we change $K \leftarrow \text{pPRF.Gen}(1^\lambda)$ into a punctured key $K\{x^*\} \leftarrow \text{pPRF.Punc}(K, x^*)$ and $\text{ek} = \mathcal{G}[\tilde{\Pi}[\vec{pk}_1, E^*, \epsilon, \alpha], \overrightarrow{\text{crs}}]$ instead of $\mathcal{G}[\tilde{\Pi}[\vec{pk}_1, E, \epsilon, \alpha], \overrightarrow{\text{crs}}]$ where $E^* = \text{EC}[K\{x^*\}]$ is the circuit as defined in Figure 4 and $y^* \leftarrow \text{pPRF.Eval}(K, x^*) \in \mathcal{Y}$. We note that the functionality and running time of both the circuits E and E^* are the same. Also, the size of the two machines $\tilde{\Pi}[\vec{pk}_1, E, \epsilon, \alpha]$ and $\tilde{\Pi}[\vec{pk}_1, E^*, \epsilon, \alpha]$ is the same due to padding. Therefore, the joint distribution $(\tilde{\Pi}[\vec{pk}_{i+1}, E, z, \alpha_{z_i}^i], \overrightarrow{\text{crs}}_i, \vec{pk}_i)$ is indistinguishable from $(\tilde{\Pi}[\vec{pk}_{i+1}, E^*, z, \alpha_{z_i}^i], \overrightarrow{\text{crs}}_i, \vec{pk}_i)$ for every label $i \in \{0, 1, \dots, n\}$ and $z \in \{0, 1\}^i$ (as discussed in Remark 1). Hence by simulation security of the RE scheme, we have $\text{Hybd}_0 \approx_\delta \text{Hybd}_1$ (notation explained in Table 1), i.e., these two hybrid games are computationally indistinguishable.

Hybd₂ This hybrid game is the same as previous one except that here we take y^* as a uniformly random element from \mathcal{Y} instead of setting $y^* \leftarrow \text{pPRF.Eval}(K, x^*) \in \mathcal{Y}$. From the pseudorandomness at punctured points (Definition 5) of the pPRF we have,

$$\begin{aligned} \mu(\lambda) &\geq |\Pr[\mathcal{A}(K\{x^*\}, \text{pPRF.Eval}(K, x^*)) = 1] - \Pr[\mathcal{A}(K\{x^*\}, U) = 1]| \\ &\geq |\Pr[\text{Hybd}_1(\lambda) = 1] - \Pr[\text{Hybd}_2(\lambda) = 1]| \end{aligned}$$

for infinitely many λ and a negligible function μ where U denotes uniform distribution over the domain \mathcal{Y} of pPRF.Eval . Since the pPRF is δ -secure, we have $\mu(\lambda) \leq \delta(\lambda)^{\omega(1)}$. Thus it holds that $\text{Hybd}_1 \approx_\delta \text{Hybd}_2$.

Hybd₃ In this hybrid game, again we consider $\text{ek} = \mathcal{G}[\tilde{\Pi}[\vec{pk}_1, E, \epsilon, \alpha], \overrightarrow{\text{crs}}]$ corresponding to the circuit $E = \text{EC}[K]$ as in the original experiment Hybd_0 . Everything else is the same as in Hybd_2 . Following the similar argument as in Hybd_1 , we have $\text{Hybd}_2 \approx_\delta \text{Hybd}_3$.

Note that Hybd_3 is actually the regular experiment $\text{Expt}_A^{w\text{PRF}}(1^\lambda, 1)$. Hence, by the above sequence of hybrid arguments, $\text{Expt}_A^{w\text{PRF}}(1^\lambda, 0)$ is indistinguishable from $\text{Expt}_A^{w\text{PRF}}(1^\lambda, 1)$ and we write $\text{Expt}_A^{w\text{PRF}}(1^\lambda, 0) \approx_\delta \text{Expt}_A^{w\text{PRF}}(1^\lambda, 1)$. This completes the proof of Theorem 2.

Corollary 1. *Assuming LWE with sub-exponential hardness and the existence of δ -sub-exponentially secure one-way functions, if there exists a weakly sub-linear compact public key functional encryption scheme for P/poly with δ -sub-exponential security, then there exists a δ -secure witness PRF scheme.*

The proof of this corollary is sketched in Appendix B.

Remark 2 A multi-relation w PRF can be obtained from the above single-relation w PRF by generating evaluation keys for various relation circuits. This can be accomplished by splitting the key generation algorithm into two separate parts, one for function secret-key and the other is for function evaluation key. We describe this in Appendix C.

4 Our Offline Witness Encryption

Construction 2. We now construct an offline witness encryption scheme OWE = (Setup, Enc, Dec) for any NP language L with relation circuit $R : \mathcal{X} \times \mathcal{W} \rightarrow \{0, 1\}$. The main ingredients and notations used in this proposed construction are the following:

- (i) A public-key encryption PKE = (Gen, Enc, Dec) semantically secure under chosen plaintext attack.
- (ii) An extractable witness PRF w PRF = (Gen, F, Eval) for the NP language $L' = \{(c_1, c_2, PK_1, PK_2) : \exists (x, m, r_1, r_2) \text{ such that } c_i = \text{PKE.Enc}(PK_i, (x, m); r_i) \text{ for } i = 1, 2\}$ with the relation $R' : \mathcal{X}' \times \mathcal{W}' \rightarrow \{0, 1\}$. Therefore, $R'((c_1, c_2, PK_1, PK_2), (x, m, r_1, r_2)) = 1$ if c_1 and c_2 are both encryptions of the same message (x, m) using public keys PK_1, PK_2 and randomness r_1, r_2 respectively; otherwise 0. Here we assume that message, ciphertext of the PKE and the w PRF value can be represented as bit-strings.
- (iii) A sub-linearly compact bounded input randomized encoding scheme RE = (Setup, Enc, Eval) in CRS model with δ -sub-exponential simulation security for Turing machines.
 - $(pp_e, pp_d) \leftarrow \text{OWE.Setup}(1^\lambda, R)$: This is run by a trusted authority to generate public parameters for both encryption and decryption where R is a relation circuit and λ is a security parameter. It works as follows:
 - Obtain two pairs of PKE keys $(SK_1, PK_1) \leftarrow \text{PKE.Gen}(1^\lambda)$ and $(SK_2, PK_2) \leftarrow \text{PKE.Gen}(1^\lambda)$.
 - Generate $(fk, ek) \leftarrow w\text{PRF.Gen}(1^\lambda, R')$ for the relation circuit R' .
 - Construct the circuit $C_1 = \text{MOC}[SK_1, fk] \in \{\mathcal{C}_\lambda\}$ as defined in Figure 5. Let S be the size of the circuit C_1 with input size n , output size l and T is the runtime bound of the circuit on an input of size n .
 - Generate $(crs_i, pk_i) \leftarrow \text{RE.Setup}(1^\lambda, 1^S, 1^n, 1^T, 1^l)$ for $i \in \{0, 1, \dots, n\}$ where crs_i is a common reference string and pk_i is an encoding key. Set $\vec{crs} = \{crs\}_{i=0}^n$ and $\vec{pk}_i = \{pk_j\}_{j=i}^n$.
 - Compute the randomized encoding $\tilde{\Pi}[\vec{pk}_1, C_1, \epsilon, \alpha] \leftarrow \text{RE.Enc}(pk_0, \Pi[\vec{pk}_1, C_1, \epsilon, \alpha], \epsilon)$ where ϵ is a null string, α is a random binary string and $\Pi[\vec{pk}_1, C_1, \epsilon, \alpha]$ is a Turing machine defined in Remark 1.
 - Build the special circuit $\mathcal{G}[\tilde{\Pi}[\vec{pk}_1, C_1, \epsilon, \alpha], \vec{crs}]$ described in Figure 1.
 - Set and output $(pp_e = (PK_1, PK_2, ek), pp_d = \mathcal{G}[\tilde{\Pi}[\vec{pk}_1, C_1, \epsilon, \alpha], \vec{crs}])$.
 - $c \leftarrow \text{OWE.Enc}(1^\lambda, x, m, pp_e)$: An encrypter encrypts a message $m \in \mathcal{M}$ with respect to an NP statement $x \in \mathcal{X}$ using the public parameters for encryption pp_e and produces a ciphertext as follows:

<p>Hardwired: a PKE secret key SK_j, a wPRF function key fk. Input: a ciphertext c and a witness $w \in \mathcal{W}$</p> <ol style="list-style-type: none"> 1. Parse $c = (c_1, c_2, x, y)$ 2. if $(w\text{PRF.F}(fk, (c_1, c_2, PK_1, PK_2)) = y)$ then 3. $(\hat{x}, \hat{m}) \leftarrow \text{PKE.Dec}(SK_j, c_j)$ 4. if $((\hat{x} = x) \wedge (R(\hat{x}, w) = 1))$ then 5. return \hat{m} 6. end if 7. end if 8. return \perp

Fig. 5. Message Output Circuit $C_j = \text{MOC}[SK_j, fk]$, $j = 1, 2$

- Choose two random strings $r_1, r_2 \xleftarrow{\$} \{0, 1\}^{l_{\text{PKE}}(\lambda)}$ where l_{PKE} is a polynomial in λ .
- Compute two ciphertexts $c_i = \text{PKE.Enc}(PK_i, (x, m); r_i)$ for $i = 1, 2$.
- Generate a w PRF evaluation of the statement (c_1, c_2, PK_1, PK_2) with witness (x, m, r_1, r_2) as $y \leftarrow w\text{PRF.Eval}(ek, (c_1, c_2, PK_1, PK_2), (x, m, r_1, r_2))$ and output $c = (c_1, c_2, x, y)$ as ciphertext.
- $\text{OWE.Dec}(c, w, pp_d)$: On receiving a ciphertext c , a receiver who has a witness w for $x \in L$, runs this algorithm using public parameter $pp_d = \mathcal{G}[\vec{\Pi}[pk_1, C_1, \epsilon, \alpha], \vec{crs}]$ for decryption to learn the message by outputting $\mathcal{G}[\vec{\Pi}[pk_1, C_1, \epsilon, \alpha], \vec{crs}](z)$ where $z = (c, w)$.

Correctness. The ciphertext c has four components where first two components c_1, c_2 are the encryptions of the same message (x, m) , the third one is a statement $x \in L$ and the last component y is a w PRF evaluation of the statement (c_1, c_2, PK_1, PK_2) with witness (x, m, r_1, r_2) . Therefore, we pass the check at line 2 of the circuit C_1 (Figure 5) as the correctness of w PRF scheme (Equation 1, Definition 8) implies $w\text{PRF.Eval}(ek, (c_1, c_2, PK_1, PK_2), (x, m, r_1, r_2)) = w\text{PRF.F}(fk, (c_1, c_2, PK_1, PK_2))$. We recover $(x, m) \leftarrow \text{PKE.Dec}(SK_1, c_1)$ in line 3 of the circuit C_1 assuming the exactness of the PKE scheme. If $w \in \mathcal{W}$ is a valid witness for the statement $x \in L$, then $R(x, w) = 1$ and the circuit C_1 returns the message $m \in \mathcal{M}$. Finally, by the correctness of RE as described in Remark 1, we have $\mathcal{G}[\vec{\Pi}[pk_1, C_1, \epsilon, \alpha], \vec{crs}](z) = C_1(z) = m$ where $z = (c, w)$. Hence Equation 4 of Definition 15 holds and the correctness of our OWE is established.

Efficiency. The encryption algorithm OWE.Enc computes two public-key encryption on a message of size $(|x| + |m|)$ and one w PRF evaluation of an input of the form (c_1, c_2, PK_1, PK_2) with size-bound $\text{poly}(\lambda, |x| + |m|)$ using a witness of the form (x, m, r_1, r_2) with size-bound $(|x| + |m| + 2 \cdot \text{poly}(\lambda))$. Therefore, time of encryption is bounded by the time of PKE.Enc and evaluation time of w PRF and we have that $\text{Time}_{\text{OWE.Enc}} \leq 2 \cdot \text{poly}(\lambda, |x| + |m|) + \text{Time}_{w\text{PRF.Eval}}$. Also, the size of the ciphertext is $\text{Size}_{\text{OWE.c}} = 2 \text{Size}_{\text{PKE.c}} + |x| + |y|$ where $\text{Size}_{\text{PKE.c}}$ denotes the size of PKE-ciphertext. We note that $|y|$ can be bounded by a constant that does not depend on the PKE scheme.

Theorem 3 *Assuming the existence of sub-exponentially secure one-way functions, our construction 2 of $\text{OWE} = (\text{Setup}, \text{Enc}, \text{Dec})$ is δ -selectively secure offline witness encryption if the underlying PKE is a δ -secure public-key encryption under chosen plaintext attack (Definition 6), the w PRF is a δ -secure*

-
1. The adversary chooses $(x, m_0, m_1, st) \leftarrow \mathcal{A}(1^\lambda)$ and sends it to the challenger. Here $x \notin L$, $|m_0| = |m_1|$ and st contains some auxiliary information.
 2. The challenger generates public parameters $(pp_e, pp_d) \leftarrow \text{OWE.Setup}(1^\lambda, R)$ as follows and sends it to \mathcal{A} :
 - 2.1 Generate $(\text{SK}_i, \text{PK}_i) \leftarrow \text{PKE.Gen}(1^\lambda)$ for $i = 1, 2$ and $(\text{fk}, \text{ek}) \leftarrow w\text{PRF.Gen}(1^\lambda, R')$ where relation R' is the same as in Construction 2
 - 2.2 Set $pp_e = (\text{PK}_1, \text{PK}_2, \text{ek})$
 - 2.3 Construct the message output circuit $C_1 = \text{MOC}[\text{SK}_1, \text{fk}]$ (see Figure 5)
 - 2.4 Generate $(\text{crs}_i, pk_i) \leftarrow \text{RE.Setup}(1^\lambda, 1^S, 1^n, 1^T, 1^l)$ for $i \in \{0, 1, \dots, n\}$ where S, n, T, l are the same as in Construction 2 and set $\vec{\text{crs}} = \{\text{crs}_i\}_{i=0}^n$ and $\vec{\text{pk}}_i = \{\text{pk}_j\}_{j=i}^n$
 - 2.5 Construct the special circuit $\mathcal{G}[\vec{\Pi}[\vec{\text{pk}}_1, C_1, \epsilon, \alpha], \vec{\text{crs}}]$ as described in Figure 1, where $\vec{\Pi}[\vec{\text{pk}}_1, C_1, \epsilon, \alpha] \leftarrow \text{RE.Enc}(pk_0, \Pi[\vec{\text{pk}}_1, C_1, \epsilon, \alpha], \epsilon)$ and $\Pi[\vec{\text{pk}}_1, C_1, \epsilon, \alpha]$ is a Turing machine defined in Remark 1.
 - 2.6 Set $pp_d = \mathcal{G}[\vec{\Pi}[\vec{\text{pk}}_1, C_1, \epsilon, \alpha], \vec{\text{crs}}]$
 3. The challenger produces the ciphertext $c \leftarrow \text{OWE.Enc}(1^\lambda, x, m_0, pp_e)$ as follows and submits it to \mathcal{A} :
 - 3.1 Chose $r_1, r_2 \xleftarrow{\$} \{0, 1\}^{\text{PKE}(\lambda)}$
 - 3.2 Compute $c_1 \leftarrow \text{PKE.Enc}(\text{PK}_1, (x, m_0); r_1)$, $c_2 \leftarrow \text{PKE.Enc}(\text{PK}_2, (x, m_0); r_2)$
 - 3.3 Evaluate $y \leftarrow w\text{PRF.Eval}(\text{ek}, (c_1, c_2, \text{PK}_1, \text{PK}_2), (x, m_0, r_1, r_2))$
 - 3.4 Set $c \leftarrow (c_1, c_2, x, y)$
 4. The adversary observing (st, c, pp_e, pp_d) , outputs a bit $b' \leftarrow \mathcal{A}(st, c, pp_e, pp_d)$.
-

Fig. 6. Hybd_0 associated with our OWE

extractable witness PRF (Definition 10) and the RE is a bounded input δ -sub-exponential simulation secure (Definition 2) sub-linear compact randomized encoding scheme (Definition 21) in CRS model for the class of Turing machines $\{\mathcal{M}_\lambda\}$ associated with the class of circuits $\{\mathcal{C}_\lambda\}$.

Proof. We show that the distinguishing advantage between two experiments $\text{Expt}_{\mathcal{A}}^{\text{OWE}}(1^\lambda, 0)$ and $\text{Expt}_{\mathcal{A}}^{\text{OWE}}(1^\lambda, 1)$ (see Figure 13) for any non-uniform PPT adversary \mathcal{A} is negligible by defining the following sequence of hybrid games and thereby, prove the indistinguishability between them. Let the challenge messages be m_0 and m_1 .

Hybd₀ The first game is the standard selective security experiment $\text{Expt}_{\mathcal{A}}^{\text{OWE}}(1^\lambda, 0)$ where the adversary \mathcal{A} is given the ciphertext corresponding to the message m_0 . We describe it in Figure 6.

Hybd₁ In this hybrid game we choose y randomly from \mathcal{Y} instead of computing $y \leftarrow w\text{PRF.Eval}(\text{ek}, (c_1, c_2, \text{PK}_1, \text{PK}_2), (x, m_0, r_1, r_2))$. We show that these games are indistinguishable under the extractable security of $w\text{PRF}$ and the semantic security of the PKE scheme.

Claim 1. *Assuming the PKE is a semantically secure public-key encryption² and the $w\text{PRF}$ is an extractable witness PRF, Hybd_0 and Hybd_1 are δ -indistinguishable.*

Proof. Suppose the OWE-adversary \mathcal{A} has a non-negligible advantage in distinguishing Hybd_0 and Hybd_1 . Then \mathcal{A} is now an adversary for $w\text{PRF}$ relative to the relation R' and the security of extractable $w\text{PRF}$ (Definition 10) implies that there is an extractor (Equation 3) \mathcal{E} that on input $\text{ek}, (c_1, c_2, \text{PK}_1, \text{PK}_2), \text{Aux}, y, \{x_i\}, r$, is able to find a witness $w' = (x, m_0, r_1,$

² We know that indistinguishability security implies semantic security for a public key encryption scheme.

-
1. The PKE-challenger runs $\text{PKE.Gen}(1^\lambda) \rightarrow (\text{SK}_2, \text{PK}_2)$ and makes PK_2 public.
 2. The PKE-adversary \mathcal{B} submits the challenge messages m'_0, m'_1 to the PKE-challenger as follows with some auxiliary information st and $|m'_0| = |m'_1|$:
 - 2.1 Invoke OWE-adversary \mathcal{A} to obtain $(x, m_0, m_1, st_{\mathcal{A}}) \leftarrow \mathcal{A}(1^\lambda)$ where $x \notin L$ and $|m_0| = |m_1|$
 - 2.2 Generate $(\text{SK}_1, \text{PK}_1) \leftarrow \text{PKE.Gen}(1^\lambda)$
 - 2.3 Compute $(\text{fk}, \text{ek}) \leftarrow w\text{PRF.Gen}(1^\lambda, R')$ for the relation R' defined in Construction 2
 - 2.4 Choose $r_1 \xleftarrow{\$} \{0, 1\}^{l_{\text{PKE}}(\lambda)}$
 - 2.5 Compute $c_1 = \text{PKE.Enc}(\text{PK}_1, (x, m_0); r_1)$
 - 2.6 Set $m'_0 = (x, m_0), m'_1 = (x, m_1)$ and $st = (\text{SK}_1, \text{PK}_1, \text{fk}, \text{ek}, c_1, r_1, x, m_0, m_1, st_{\mathcal{A}})$
 3. The PKE-challenger chooses a random bit $b \in \{0, 1\}$ and sends the ciphertext $c'_b \leftarrow \text{PKE.Enc}(\text{PK}_2, m'_b = (x, m_b); r_2)$ to \mathcal{B} , where $r_2 \xleftarrow{\$} \{0, 1\}^{l_{\text{PKE}}(\lambda)}$.
 4. The PKE-adversary \mathcal{B} simulates \mathcal{A} to output a guess for b by observing (st, c'_b) as follows:
 - 4.1 Set $c_2 = c'_b$
 - 4.2 Compute $y \xleftarrow{\$} \mathcal{Y}$
 - 4.3 Construct the message output circuit $C_1 = \text{MOC}[\text{SK}_1, \text{fk}]$ (see Figure 5)
 - 4.4 Generate $(\text{crs}_i, \text{pk}_i) \leftarrow \text{RE.Setup}(1^\lambda, 1^S, 1^n, 1^T, 1^l)$ for $i \in \{0, 1, \dots, n\}$ where S, n, T, l are the same as in Construction 2 and set $\vec{\text{crs}} = \{\text{crs}_i\}_{i=0}^n$ and $\vec{\text{pk}}_i = \{\text{pk}_j\}_{j=i}^n$
 - 4.5 Construct the special circuit $\mathcal{G}[\vec{\Pi}[\vec{\text{pk}}_1, C_1, \epsilon, \alpha], \vec{\text{crs}}]$ as described in Figure 1, where $\vec{\Pi}[\vec{\text{pk}}_1, C_1, \epsilon, \alpha] \leftarrow \text{RE.Enc}(\text{pk}_0, \Pi[\vec{\text{pk}}_1, C_1, \epsilon, \alpha], \epsilon)$ and $\Pi[\vec{\text{pk}}_1, C_1, \epsilon, \alpha]$ is a Turing machine defined in Remark 1.
 - 4.6 Set $c = (c_1, c_2, x, y)$, $pp_e = (\text{PK}_1, \text{PK}_2, \text{ek})$, $pp_d = \mathcal{G}[\vec{\Pi}[\vec{\text{pk}}_1, C_1, \epsilon, \alpha], \vec{\text{crs}}]$ and send $(st_{\mathcal{A}}, c, pp_e, pp_d)$ to the OWE-adversary \mathcal{A}
 - 4.7 Output a guess $b' \leftarrow \mathcal{A}(st_{\mathcal{A}}, c, pp_e, pp_d)$ for b
-

Fig. 7. The PKE-adversary \mathcal{B} simulating Hybd_2

r_2), where Aux contains $st_{\mathcal{A}}$, the $w\text{PRF}$ queries $\{x_i\}$ and r indicates the random coin used by \mathcal{A} . Therefore, \mathcal{E} breaks the semantic security of the underlying PKE scheme used in our construction and we arrive at a contradiction. This proves that $\text{Hybd}_0 \approx_\delta \text{Hybd}_1$.

Hybd₂ In this hybrid game, we set $c_2 \leftarrow \text{PKE.Enc}(\text{PK}_2, (x, m_1); r_2)$ instead of $c_2 \leftarrow \text{PKE.Enc}(\text{PK}_2, (x, m_0); r_2)$. The distribution of ciphertexts in Hybd_1 and Hybd_2 are computationally indistinguishable due to the CPA-security of underlying PKE scheme (Definition 7). We prove this in the following claim.

Claim 2. *Assuming the PKE is a δ -secure public-key encryption under chosen plaintext attack, Hybd_1 and Hybd_2 are δ -indistinguishable.*

Proof. To prove this we construct a PKE-adversary \mathcal{B} against the security (Definition 7) of PKE scheme for the key PK_2 as described in Figure 7. From the construction we see that if $c'_b \leftarrow \text{PKE.Enc}(\text{PK}_2, (x, m_0); r_2)$, then \mathcal{B} simulates Hybd_0 . If $c'_b \leftarrow \text{PKE.Enc}(\text{PK}_2, (x, m_1); r_2)$, then \mathcal{B} simulates Hybd_1 . Therefore, the distinguishing advantage of \mathcal{A} between Hybd_0 and Hybd_1 can be bounded as

$$\begin{aligned} & |\Pr[\text{Hybd}_0(\lambda) = 1] - \Pr[\text{Hybd}_1(\lambda) = 1]| \\ & \leq |\Pr[\text{Expt}_{\mathcal{B}}^{\text{PKE}}(1^\lambda, 0) = 1] - \Pr[\text{Expt}_{\mathcal{B}}^{\text{PKE}}(1^\lambda, 1) = 1]|. \end{aligned}$$

Then, we can use the indistinguishability guarantee of the underlying PKE and it implies that the above advantage can be made less than a negligible function of λ . Therefore, we have $\text{Hybd}_1 \approx_\delta \text{Hybd}_2$.

-
1. The OWE-adversary chooses $(\bar{x}, \bar{m}_0, \bar{m}_1, st) \leftarrow \mathcal{A}(1^\lambda)$ and sends it to the RE-adversary \mathcal{B} , where $|\bar{x}| \leq L$, $|\bar{m}_0| = |\bar{m}_1|$ and st contains some auxiliary information.
 2. The RE-adversary \mathcal{B} generates public parameters (pp_e, pp_d) as follows and sends it to \mathcal{A} :
 - 2.1 Generate $(SK_i, PK_i) \leftarrow \text{PKE.Gen}(1^\lambda)$ for $i = 1, 2$ and $(fk, ek) \leftarrow w\text{PRF.Gen}(1^\lambda, R')$ where relation R' is the same as in Construction 2
 - 2.2 Set $pp_e = (PK_1, PK_2, ek)$
 - 2.3 Construct the message output circuits $C_j = \text{MOC}[SK_j, fk]$ for $j = 1, 2$ (see Figure 5)
 - 2.4 Generate $(crs_i, pk_i) \leftarrow \text{RE.Setup}(1^\lambda, 1^S, 1^n, 1^T, 1^l)$ for $i \in \{0, 1, \dots, n\}$ where S, n, T, l are the same as in Construction 2 and set $\vec{crs} = \{crs\}_{i=0}^n$ and $\vec{pk}_i = \{pk_i\}_{j=i}^n$
 - 2.5 Submit the circuits C_j to the RE-challenger for $j = 1, 2$
 - 2.6 The RE-challenger pick a random $j \xleftarrow{\$} \{1, 2\}$ and sends $\tilde{\Pi}[\vec{pk}_1, C_j, \epsilon, \alpha] \leftarrow \text{RE.Enc}(pk_0, \Pi[\vec{pk}_1, C_j, \epsilon, \alpha], \epsilon)$ to \mathcal{B} where $\Pi[\vec{pk}_1, C_j, \epsilon, \alpha]$ is a Turing machine defined in Remark 1.
 - 2.7 Construct the special circuit $\mathcal{G}[\tilde{\Pi}[\vec{pk}_1, C_j, \epsilon, \alpha], \vec{crs}]$ as described in Figure 1.
 - 2.8 Set $pp_d = \mathcal{G}[\tilde{\Pi}[\vec{pk}_1, C_1, \epsilon, \alpha], \vec{crs}]$
 3. The RE-adversary \mathcal{B} produces a OWE-ciphertext \bar{c} as follows and submits it to the OWE-adversary \mathcal{A} :
 - 3.1 Chose $r_1, r_2 \xleftarrow{\$} \{0, 1\}^l_{\text{PKE}(\lambda)}$
 - 3.2 Compute $\bar{c}_1 \leftarrow \text{PKE.Enc}(PK_1, (\bar{x}, \bar{m}_0); r_1)$, $\bar{c}_2 \leftarrow \text{PKE.Enc}(PK_2, (\bar{x}, \bar{m}_1); r_2)$
 - 3.3 Choose $\bar{y} \xleftarrow{\$} \mathcal{Y}$
 - 3.4 Set $\bar{c} = (\bar{c}_1, \bar{c}_2, \bar{x}, \bar{y})$ and send it to the OWE-adversary \mathcal{A}
 4. Output $b' \leftarrow \mathcal{A}(st, \bar{c}, pp_e, pp_d)$.
-

Fig. 8. The RE-adversary \mathcal{B} simulating **Hybd₃**

Hybd₃ This hybrid game is the same as the previous game except that we take pp_d as the circuit $\mathcal{G}[\tilde{\Pi}[\vec{pk}_1, C_2, \epsilon, \alpha], \vec{crs}]$ instead of setting $pp_d \leftarrow \mathcal{G}[\tilde{\Pi}[\vec{pk}_1, C_1, \epsilon, \alpha], \vec{crs}]$. We show that the adversary \mathcal{A} 's distinguishing advantage between **Hybd₂** and **Hybd₃** is negligible in the following claim.

Claim 3. *Assuming the RE is a δ -sub-exponential simulation secure sub-linear compact randomized encoding scheme in CRS model for the class of Turing machines $\{\mathcal{M}_\lambda\}$ associated with the class of circuits $\{\mathcal{C}_\lambda\}$, **Hybd₂** and **Hybd₃** are δ -indistinguishable.*

Proof. We need to show that the joint distributions $(\tilde{\Pi}[\vec{pk}_{i+1}, C_1, z, \alpha_{z_i}^i], \vec{crs}_i, \vec{pk}_i)$ and $(\tilde{\Pi}[\vec{pk}_{i+1}, C_2, z, \alpha_{z_i}^i], \vec{crs}_i, \vec{pk}_i)$ for every label $i \in \{0, 1, \dots, n\}$ and $z \in \{0, 1\}^i$, are indistinguishable. It will imply that the two hybrids **Hybd₂**, **Hybd₃** are indistinguishable. If the functionality, runtime and size of two circuits C_1 and C_2 are the same then the above indistinguishability follows from the underlying simulation security of RE scheme in CRS model according to the discussion in Remark 1.

We define an RE-adversary \mathcal{B} against the indistinguishability secure RE scheme in Figure 8. We note RE is δ -indistinguishability secure implies that, if the two ensembles $\{\Pi_1(x_1), |\Pi_1|, |x_1|, T_1 : (\Pi_1, x_1, T_1) \xleftarrow{\$} X_{1,\lambda}\}$ and $\{\Pi_2(x_2), |\Pi_2|, |x_2|, T_2 : (\Pi_2, x_2, T_2) \xleftarrow{\$} X_{2,\lambda}\}$ are δ -indistinguishable then the two distributions $\{\text{RE.Enc}(pk, \Pi_1, x_1) : (\Pi_1, x_1, T_1) \xleftarrow{\$} X_{1,\lambda}\}$ and $\{\text{RE.Enc}(pk, \Pi_2, x_2) : (\Pi_2, x_2, T_2) \xleftarrow{\$} X_{2,\lambda}\}$ are also δ -indistinguishable, where $\Pi_j \in \mathcal{M}_\lambda$ and T_j denotes the runtime of Π_j on input x_j for $j = 1, 2$.

Therefore, if $pp_d = \mathcal{G}[\tilde{\Pi}[\vec{pk}_1, C_1, \epsilon, \alpha], \vec{crs}]$ then \mathcal{B} simulates **Hybd₁** and if pp_d

$= \mathcal{G}[\tilde{\Pi}[\vec{pk}_1, C_2, \epsilon, \alpha], \vec{crs}]$ then \mathcal{B} simulates Hybd_2 . Now we show the functional equivalence of the circuits C_1 and C_2 . Let (c, w) be any arbitrary input to the circuits C_j , $j = 1, 2$ where $c = (c_1, c_2, x, y)$.

Case 1. ($x = \bar{x}, c_1 = \bar{c}_1$ and $c_2 = \bar{c}_2$): Since $\bar{x} \notin L$, we have $R(x, w) = 0$ in line 4 of C_j (Figure 5), thus C_1 and C_2 both output \perp .

Case 2. ($x \neq \bar{x}, c_1 = \bar{c}_1$ and $c_2 = \bar{c}_2$): Correctness of PKE scheme implies $\text{PKE.Dec}(\text{SK}_j, c_j) = (\bar{x}, \bar{m}_j)$ in line 3 of C_j (Figure 5) and both the circuits returns \perp as $x \neq \bar{x}$ in line 4.

Case 3. ($c_1 \neq \bar{c}_1$ or $c_2 \neq \bar{c}_2$): If c_1 and c_2 are encryptions of the same message then we have $\text{PKE.Dec}(\text{SK}_1, c_1) = \text{PKE.Dec}(\text{SK}_2, c_2)$. Therefore, the behavior of both circuits C_1 and C_2 are the same as they differ only in line 3. If the decryptions of c_1 and c_2 are not equal then $(c_1, c_2, \text{PK}_1, \text{PK}_2) \notin L'$ and by the correctness of $w\text{PRF}$ scheme we have $y \neq w\text{PRF.F}(\text{fk}, (c_1, c_2, \text{PK}_1, \text{PK}_2))$. Hence, the circuits C_1 and C_2 return \perp due to line 2 (Figure 5).

This shows that C_1 and C_2 are functionally equivalent. Also, we note that size and time bound for both the circuits are the same. Hence, we have $\text{Hybd}_2 \approx_\delta \text{Hybd}_3$. This completes the proof of Claim 3.

Hybd₄ The only difference of this hybrid from Hybd_3 is that we compute $c_1 \leftarrow \text{PKE.Enc}(\text{PK}_1, (x, m_1); r_1)$ instead of $c_1 \leftarrow \text{PKE.Enc}(\text{PK}_1, (x, m_0); r_1)$. Therefore, Hybd_3 and Hybd_4 are computationally indistinguishable by the CPA security of the underlying PKE scheme for the key PK_1 .

Claim 4. *Assuming the PKE is a δ -secure public-key encryption under chosen plaintext attack, Hybd_3 and Hybd_4 are δ -indistinguishable.*

Hybd₅ In this hybrid game we take pp_d as the circuit $\mathcal{G}[\tilde{\Pi}[\vec{pk}_1, C_1, \epsilon, \alpha], \vec{crs}]$ instead of $\mathcal{G}[\tilde{\Pi}[\vec{pk}_1, C_2, \epsilon, \alpha], \vec{crs}]$ as in the standard scheme. Therefore, by the underlying simulation secure RE scheme we have Hybd_4 and Hybd_5 are computationally indistinguishable as stated in the following claim.

Claim 5. *Assuming the RE is a δ -sub-exponential simulation secure sub-linear compact randomized encoding scheme in CRS model for the class of Turing machines $\{\mathcal{M}_\lambda\}$ associated with the class of circuits $\{\mathcal{C}_\lambda\}$, Hybd_4 and Hybd_5 are δ -indistinguishable.*

Hybd₆ In this hybrid we compute $y \leftarrow w\text{PRF.Eval}(\text{ek}, (c_1, c_2, \text{PK}_1, \text{PK}_2), (x, m_0, r_1, r_2))$ instead of choosing y randomly from \mathcal{Y} . The indistinguishability is guaranteed by the following claim.

Claim 6. *Assuming the PKE is a semantically secure public-key encryption and the $w\text{PRF}$ is an extractable witness PRF, Hybd_5 and Hybd_6 are δ -indistinguishable.*

The proofs of claim 4, 5, and 6 are analogous to that of claim 2, 3, and 1 respectively. Observe that Hybd_6 is the experiment $\text{Expt}_A^{\text{OWE}}(1^\lambda, 1)$. The indistinguishability between the above hybrid games implies that $\text{Expt}_A^{\text{OWE}}(1^\lambda, 0) \approx_\delta$

<p>Hardwired: a PKE secret key SK_j, a wPRF function key fk. Input: a ciphertext c and a witness $w \in \mathcal{W}$</p> <ol style="list-style-type: none"> 1. Parse $c = (c_1, c_2, x, y)$ 2. if $(w\text{PRF.F}(fk, (c_1, c_2, PK_1, PK_2)) = y)$ then 3. $(\hat{x}, (\hat{f}, \hat{m})) \leftarrow \text{PKE.Dec}(SK_j, c_j)$ 4. if $((\hat{x} = x) \wedge (R(\hat{x}, w) = 1))$ then 5. return $\hat{f}(\hat{m}, w)$ 6. end if 7. end if 8. return \perp
--

Fig. 9. Modified Message Output Circuit $F_j = \text{MMOC}[SK_j, fk]$, $j = 1, 2$

$\text{Expt}_{\mathcal{A}}^{\text{OWE}}(1^\lambda, 1)$ and the distinguishing advantage for the adversary \mathcal{A} is strictly less than $\mu(\lambda)$, where μ is a negligible function of λ . This completes the proof.

Remark 3 We convert our OWE scheme into an offline functional witness encryption (OFWE) scheme for a class of functions $\{f_\lambda\}_{\lambda \in \mathbb{N}}$. The encryption algorithm of OFWE is the same as our OWE except that it takes an additional input a function $f \in f_\lambda$ and then encrypts the pair of the function f and a message m with the statement x using the PKE encryption to produce ciphertexts $c_i \leftarrow \text{PKE.Enc}(PK_i, (x, (f, m))); r_i$ for $i = 1, 2$. In line 3 of the circuit C_j (Figure 5), we will have $\text{PKE.Dec}(SK_j, c_j) = (\hat{x}, (\hat{f}, \hat{m}))$ and in line 5 it will return $\hat{f}(\hat{m}, w)$ instead of \hat{m} (see circuit F_j in Figure 9). Rest of the algorithms of OFWE.Setup and OFWE.Dec will be the same as that of our OWE scheme. The time of encryption of the OFWE is bounded by $\text{poly}(\lambda, |x| + |m| + |f|)$ where $|x|, |m|, |f|$ are the size of x, m, f respectively. The correctness and the security of the OFWE depend on the same assumptions as in the case of our OWE.

References

1. H. Abusalah, G. Fuchsbauer, and K. Pietrzak. Offline witness encryption. In *International Conference on Applied Cryptography and Network Security*, pages 285–303. Springer, 2016.
2. P. Ananth, Z. Brakerski, G. Segev, and V. Vaikuntanathan. The trojan method in functional encryption: From selective to adaptive security, generically. *IACR Cryptology ePrint Archive*, 2014:917, 2014.
3. N. Bitansky, R. Nishimaki, A. Passelègue, and D. Wichs. From cryptomania to obfustopia through secret-key functional encryption. In *Theory of Cryptography Conference*, pages 391–418. Springer, 2016.
4. N. Bitansky and O. Paneth. Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In *Theory of Cryptography Conference*, pages 401–427. Springer, 2015.
5. D. Boneh and B. Waters. Constrained pseudorandom functions and their applications. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 280–300. Springer, 2013.

6. D. Boneh, D. J. Wu, and J. Zimmerman. Immunizing multilinear maps against zeroizing attacks. Cryptology ePrint Archive, Report 2014/930, 2014. <https://eprint.iacr.org/2014/930>.
7. E. Boyle, K.-M. Chung, and R. Pass. On extractability (aka differing-inputs) obfuscation. TCC, 2014.
8. E. Boyle, S. Goldwasser, and I. Ivan. Functional signatures and pseudorandom functions. In *International Workshop on Public Key Cryptography*, pages 501–519. Springer, 2014.
9. Z. Brakerski, I. Komargodski, and G. Segev. Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions. Cryptology ePrint Archive, Report 2015/158, 2015. <https://eprint.iacr.org/2015/158>.
10. J. H. Cheon, K. Han, C. Lee, H. Ryu, and D. Stehle. Cryptanalysis on the multilinear map over the integers and its related problems. Cryptology ePrint Archive, Report 2014/906, 2014. <https://eprint.iacr.org/2014/906>.
11. J.-S. Coron, T. Lepoint, and M. Tibouchi. Practical multilinear maps over the integers. In *Advances in Cryptology-CRYPTO 2013*, pages 476–493. Springer, 2013.
12. J.-S. Coron, T. Lepoint, and M. Tibouchi. Cryptanalysis of two candidate fixes of multilinear maps over the integers. *IACR Cryptology ePrint Archive*, 2014:975, 2014.
13. S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 1–17. Springer, 2013.
14. S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM Journal on Computing*, 45(3):882–929, 2016.
15. S. Garg, C. Gentry, A. Sahai, and B. Waters. Witness encryption and its applications. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 467–476. ACM, 2013.
16. C. Gentry, S. Halevi, H. K. Maji, and A. Sahai. Zeroizing without zeroes: Cryptanalyzing multilinear maps without encodings of zero. Cryptology ePrint Archive, Report 2014/929, 2014. <https://eprint.iacr.org/2014/929>.
17. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33:792–807, 1986.
18. S. Goldwasser, Y. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 555–564. ACM, 2013.
19. J. Groth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero knowledge for np. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 339–358. Springer, 2006.
20. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 415–432. Springer, 2008.
21. Y. Ishai and E. Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 294–304. IEEE, 2000.
22. L. A. Levin. One way functions and pseudorandom generators. *Combinatorica*, 7(4):357–363, 1987.

23. H. Lin, R. Pass, K. Seth, and S. Telang. Output-compressing randomized encodings and applications. In *Theory of Cryptography Conference*, pages 96–124. Springer, 2016.
24. A. Sahai and B. Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 475–484. ACM, 2014.
25. M. Zhandry. How to avoid obfuscation using witness prfs. In *Theory of Cryptography Conference*, pages 421–448. Springer, 2016.

A Useful Definitions

A.1 Pseudorandom Function

A finite set of functions $\{F_s : \mathcal{X} \rightarrow \mathcal{Y}\}_s$ with a seed or key s is said to form a pseudorandom function family [17] if F_s can be efficiently computed for given s and is computationally indistinguishable from a random function $R : \mathcal{X} \rightarrow \mathcal{Y}$ given oracle access to R .

Definition 3 (Pseudorandom function). A *pseudorandom function* (PRF) is a function $F : \{0, 1\}^\lambda \times \mathcal{X} \rightarrow \mathcal{Y}$ with polynomial runtime satisfying

$$|\Pr[A^{F(K, \cdot)}(1^\lambda) = 1 : K \xleftarrow{\$} \{0, 1\}^\lambda] - \Pr[A^{R(\cdot)}(1^\lambda) = 1 : R \xleftarrow{\$} \mathcal{U}]| \leq \mu(\lambda)$$

for every probabilistic polynomial time (PPT) adversary \mathcal{A} , where \mathcal{U} is the set of all functions from \mathcal{X} to \mathcal{Y} and μ is a negligible function in λ . The pseudorandom function F is said to be δ -secure for some specific negligible function $\delta(\cdot)$ if the indistinguishability gap $\mu(\lambda)$ is less than $\delta(\lambda)^{\Omega(1)}$.

A.2 Puncturable Pseudorandom Function

Sahai and Waters [24] introduced a key-puncturing technique for pseudorandom functions that can be used to build many cryptographic primitives with the help of obfuscation. The punctured key of a puncturable pseudorandom function allows to evaluate PRF at all points except for the points in a (predefined) polynomial-size set.

Definition 4 (Puncturable pseudorandom function). A *puncturable pseudorandom function* (pPRF) consists of a tuple of algorithms $\text{pPRF} = (\text{Gen}, \text{Eval}, \text{Punc})$ over the domain \mathcal{X} and range \mathcal{Y} and is defined as follows:

- $K \leftarrow \text{pPRF.Gen}(1^\lambda)$: It is a randomized algorithm run by a trusted authority which takes as input a security parameter λ and outputs a secret key $K \in \{0, 1\}^\lambda$.
- $y \leftarrow \text{pPRF.Eval}(K', x)$: It is a deterministic algorithm which on input a key K' and an element $x \in \mathcal{X}$, outputs the PRF value $y \in \mathcal{Y}$.
- $K\{\mathcal{S}\} \leftarrow \text{pPRF.Punc}(K, \mathcal{S})$: It is a deterministic algorithm that takes a secret key K and a polynomial-size set $\mathcal{S} \subset \mathcal{X}$ as input and outputs a punctured key $K\{\mathcal{S}\}$. If \mathcal{S} contains a single element, say x , then we simply write $K\{\mathcal{S}\}$ as $K\{x\}$.

Correctness: (Functionality preserving under puncturing) For all polynomial-size subset \mathcal{S} of \mathcal{X} , and for all $x \in \mathcal{X} \setminus \mathcal{S}$ we have that

$$\Pr[\text{pPRF.Eval}(K, x) = \text{pPRF.Eval}(K\{\mathcal{S}\}, x)] = 1.$$

Definition 5 (Pseudorandomness at punctured points). We say that a puncturable pseudorandom function $\text{pPRF} = (\text{Gen}, \text{Eval}, \text{Punc})$ preserves pseudorandomness at punctured points if

$$|\Pr[\mathcal{A}(K\{\mathcal{S}\}, \{\text{pPRF.Eval}(K, x)\}_{x \in \mathcal{S}}) = 1] - \Pr[\mathcal{A}(K\{\mathcal{S}\}, U^{|\mathcal{S}|}) = 1]| \leq \mu(\lambda)$$

for every PPT adversary \mathcal{A} and any polynomial-size subset \mathcal{S} of \mathcal{X} , where $K \leftarrow \text{pPRF.Gen}(1^\lambda)$, $K\{\mathcal{S}\} \leftarrow \text{pPRF.Punc}(K, \mathcal{S})$, U denotes the uniform distribution over \mathcal{Y} and μ is a negligible function in λ . The pPRF is said to be δ -secure for some specific negligible function $\delta(\cdot)$ if the above indistinguishability gap $\mu(\lambda)$ is less than $\delta(\lambda)^{\Omega(1)}$.

It has been observed by [5,8] that puncturable PRFs can be constructed from one-way functions using the GGM tree-based construction of PRFs [17] where the size of the punctured key grows polynomially with the number of elements in the set \mathcal{S} . The GGM construction [17] of pseudorandom functions uses a cryptographically strong bit (CSB) generator or pseudorandom generator (PRG) which can be obtained from one-way functions. Moreover, if the one-way functions are assumed to be sub-exponentially hard then the PRG is also sub-exponentially secure. We describe these facts in the following theorems:

Theorem 4 [17,22] *Assuming the existence of sub-exponentially secure one-way functions, there exists an efficiently computable sub-exponentially secure pseudorandom generator for any desired poly-size input length.*

Theorem 5 [17,5,8] *Assuming the existence of one-way functions, there exists an efficiently computable puncturable pseudorandom function for any desired poly-size input length.*

A.3 Public-Key Encryption

Definition 6 (Public-key encryption). A *public-key encryption* scheme for a message space \mathcal{M} is a tuple of PPT algorithms $\text{PKE} = (\text{Gen}, \text{Enc}, \text{Dec})$ with the following properties:

- $(\text{SK}, \text{PK}) \leftarrow \text{PKE.Gen}(1^\lambda)$: This is a randomized key generation algorithm which is run by a trusted third party with a security parameter λ as input. It outputs a public key PK and a secret key SK. A user who obtains a key pair (PK, SK) from a trusted party, keeps the secret key SK and publishes the public key PK.
- $c \leftarrow \text{PKE.Enc}(\text{PK}, m; r)$: The encrypter uses the public key PK to encrypt a message $m \in \mathcal{M}$ using a randomness r and produces a ciphertext c which is broadcasted over a public domain.
- $\text{PKE.Dec}(\text{SK}, c) \in \mathcal{M} \cup \{\perp\}$: The recipient of a ciphertext c runs this algorithm using the secret key SK and gets either a message $m \in \mathcal{M}$ or \perp where \perp indicates a failure of the algorithm.

Correctness: For every $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$, we have

$$\Pr[\text{PKE.Dec}(\text{SK}, c) = m : (\text{SK}, \text{PK}) \leftarrow \text{PKE.Gen}(1^\lambda), c \leftarrow \text{PKE.Enc}(\text{PK}, m; r)] = 1$$

-
1. The challenger runs $\text{PKE.Gen}(1^\lambda) \rightarrow (\text{SK}, \text{PK})$ and makes PK public.
 2. The adversary \mathcal{A} selects $m_0, m_1 \in \mathcal{M}$ such that $|m_0| = |m_1|$ and sends (m_0, m_1, st) to the challenger where st contains some auxiliary information.
 3. Next, the challenger chooses a random bit $b \in \{0, 1\}$, a randomness r and sends $c_b \leftarrow \text{PKE.Enc}(\text{PK}, m_b; r)$ to \mathcal{A} .
 4. The adversary \mathcal{A} observes c_b and st and outputs a guess b' for b .
-

Fig. 10. $\text{Expt}_{\mathcal{A}}^{\text{PKE}}(1^\lambda, b)$: The security game of a CPA-secure public-key encryption

Definition 7 (Indistinguishability under chosen-plaintext attacks). We say that a public-key encryption scheme $\text{PKE} = (\text{Gen}, \text{Enc}, \text{Dec})$ is indistinguishable under chosen plaintext attacks (CPA) if

$$|\Pr[\text{Expt}_{\mathcal{A}}^{\text{PKE}}(1^\lambda, 0) = 1] - \Pr[\text{Expt}_{\mathcal{A}}^{\text{PKE}}(1^\lambda, 1) = 1]| \leq \mu(\lambda)$$

for any $\lambda \in \mathbb{N}$ and every PPT adversary \mathcal{A} in the experiments $\text{Expt}_{\mathcal{A}}^{\text{PKE}}(1^\lambda, b)$ defined in Figure 10 where $b \in \{0, 1\}$ and μ is a negligible function of λ . The PKE is said to be δ -selectively secure for some specific negligible function $\delta(\cdot)$ if the above indistinguishability gap $\mu(\lambda)$ is less than $\delta(\lambda)^{\Omega(1)}$.

A.4 Witness PRF

Informally, a witness PRF scheme [25] produces a somewhat random value from a set with respect to an instance $x \in L$ for an NP language L and a user can recompute the value provided he has a witness w for $x \in L$.

Definition 8 (Witness PRF). A *witness PRF* (*wPRF*) for an NP language L with the witness relation $R : \chi \times \mathcal{W} \rightarrow \{0, 1\}$ consists of three algorithms $w\text{PRF} = (\text{Gen}, \text{F}, \text{Eval})$ and works as follows:

- $(\text{fk}, \text{ek}) \leftarrow w\text{PRF.Gen}(1^\lambda, R)$: A trusted authority ³ takes as input the security parameter λ and a relation circuit $R : \chi \times \mathcal{W} \rightarrow \{0, 1\}$ and randomly generates a secret function key fk and a public evaluation key ek . A user receiving (fk, ek) through a secure channel, keeps fk as a secret key and publishes ek . We note that $R(x, w) = 1$ if and only if w is a valid witness for $x \in L$.
- $y \leftarrow w\text{PRF.F}(\text{fk}, x)$: Using a function key fk and an input $x \in \chi$, the user runs this algorithm which deterministically outputs some $y \in \mathcal{Y}$.
- $w\text{PRF.Eval}(\text{ek}, x, w) \in \mathcal{Y} \cup \{\perp\}$: An witness holder runs this algorithm using an evaluation key ek , an input $x \in \chi$ and a witness $w \in \mathcal{W}$ and deterministically recovers either $y \in \mathcal{Y}$ or \perp .

Correctness: For all $x \in \mathcal{X}$, $w \in \mathcal{W}$, we have that

$$w\text{PRF.Eval}(\text{ek}, x, w) = \begin{cases} w\text{PRF.F}(\text{fk}, x) & \text{if } R(x, w) = 1 \\ \perp & \text{if } R(x, w) = 0 \end{cases} \quad (1)$$

Definition 9 (Selectively secure witness PRF). We say that a witness PRF scheme $w\text{PRF} = (\text{Gen}, \text{F}, \text{Eval})$ for an NP language L , a relation $R : \chi \times \mathcal{W} \rightarrow \{0, 1\}$, a set \mathcal{Y} , is selectively secure if

³ We note that a user may itself run this algorithm to get the secret function key fk and the evaluation key ek which is made public.

-
1. The adversary \mathcal{A} chooses a single challenge query on an instance $x^* \in \mathcal{X} \setminus L$ to the challenger.
 2. The challenger generates $(\text{fk}, \text{ek}) \leftarrow w\text{PRF.Gen}(1^\lambda, R)$ and gives ek to the adversary \mathcal{A} .
 3. The challenger computes $y_0 \leftarrow w\text{PRF.F}(\text{fk}, x^*)$, selects $y_1 \xleftarrow{\$} \mathcal{Y}$ and sends y_b to \mathcal{A} for a randomly chosen $b \in \{0, 1\}$.
 4. The adversary \mathcal{A} makes polynomially many queries on instance $x_i \in \mathcal{X}$, $i = 1, 2, \dots, u$, to which the challenger responds with $w\text{PRF.F}(\text{fk}, x_i)$ if $x_i \neq x^*$; otherwise terminates the game.
 5. Then \mathcal{A} outputs a guess b' for b .
-

Fig. 11. $\text{Expt}_{\mathcal{A}}^{w\text{PRF}}(1^\lambda, b)$: The security game of a selectively-secure witness PRF

$$\left| \Pr[\text{Expt}_{\mathcal{A}}^{w\text{PRF}}(1^\lambda, 0) = 1] - \Pr[\text{Expt}_{\mathcal{A}}^{w\text{PRF}}(1^\lambda, 1) = 1] \right| \leq \mu(\lambda)$$

for any $\lambda \in \mathbb{N}$ and every PPT adversary \mathcal{A} in the experiments $\text{Expt}_{\mathcal{A}}^{w\text{PRF}}(1^\lambda, b)$ defined in Figure 11 where $b \in \{0, 1\}$ and μ is a negligible function of λ . The $w\text{PRF}$ is said to be δ -selectively secure for some specific negligible function $\delta(\cdot)$ if the above indistinguishability gap $\mu(\lambda)$ is smaller than $\delta(\lambda)^{\Omega(1)}$.

Definition 10 (Extractable witness PRFs). A witness PRF scheme $w\text{PRF} = (\text{Gen}, \text{F}, \text{Eval})$ for an NP language L with relation R is said to be a secure *extractable witness PRF* with respect to an R -instance sampler \mathcal{D} if there exists a polynomial $p(\cdot)$ such that

$$\left| \Pr \left[\begin{array}{l} \mathcal{A}^{w\text{PRF.F}(\text{fk}, \cdot)}(\text{ek}, x^*, \text{Aux}, y^*) = 1 : (\text{fk}, \text{ek}) \leftarrow w\text{PRF.Gen}(\lambda, R), \\ (x^*, \text{Aux}) \xleftarrow{\$} \mathcal{D}^{w\text{PRF.F}(\text{fk}, \cdot)}(\text{ek}), y^* \leftarrow w\text{PRF.F}(\text{fk}, x^*) \end{array} \right] - \Pr \left[\begin{array}{l} \mathcal{A}^{w\text{PRF.F}(\text{fk}, \cdot)}(\text{ek}, x^*, \text{Aux}, y^*) = 1 : (\text{fk}, \text{ek}) \leftarrow w\text{PRF.Gen}(\lambda, R), \\ (x^*, \text{Aux}) \xleftarrow{\$} \mathcal{D}^{w\text{PRF.F}(\text{fk}, \cdot)}(\text{ek}), y^* \xleftarrow{\$} \mathcal{Y} \end{array} \right] \right| \geq \frac{1}{2} + \frac{1}{p(\lambda)} \quad (2)$$

for every PPT adversary \mathcal{A} and infinitely many λ , then there exists a PPT extractor \mathcal{E} and a polynomial $q(\cdot)$ such that

$$\Pr \left[\begin{array}{l} w^* \xleftarrow{\$} \mathcal{E}(\text{ek}, x^*, \text{Aux}, y^*, \{x_i\}, r) : R(x^*, w^*) = 1, \{x_i\} \text{ are the } w\text{PRF.F} \text{ queries} \\ \text{of } \mathcal{A} \text{ and } r \text{ is the random coin of } \mathcal{A} \end{array} \right] \geq \frac{1}{q(\lambda)} \quad (3)$$

for infinitely many λ .

A.5 Multi-Relation Witness PRF

The notion of multi-relation witness PRFs was introduced by Zhandry [25] to work with multiple relations but with the same secret function key.

Definition 11 (Multi-Relation Witness PRFs). A multi-relation witness PRF scheme for a set of relations $\mathcal{R} = \{R : |R| \leq s, R : \chi \times \mathcal{W} \rightarrow \{0, 1\}\}$ consists of three algorithms $mw\text{PRF} = (\text{Gen}, \text{F}, \text{Eval})$ and works as follows:

- $\text{fk} \leftarrow mw\text{PRF.Gen}(\lambda, s)$: It is a randomized algorithm run by a user⁴ which takes as input a security parameter λ and a bound s on the size of supported relations and produces a secret function key fk .

⁴ This algorithm can be processed by a trusted third party to generate the secret function key fk and in that case the key is sent to a user through a secure channel.

-
1. The adversary \mathcal{A} chooses a single challenge query on an instance $x^* \in \mathcal{X}$ to the challenger.
 2. The challenger generates $\text{fk} \leftarrow \text{mwPRF.Gen}(1^\lambda, s)$, and delivers s to the adversary \mathcal{A} .
 3. The challenger computes $y_0 \leftarrow \text{mwPRF.F}(\text{fk}, x^*)$ and $y_1 \leftarrow \mathcal{Y}$ and sends y_b to \mathcal{A} for a randomly chosen $b \in \{0, 1\}$.
 4. The adversary \mathcal{A} makes polynomially many evaluation key queries $R_i \in \mathcal{R}$ for $i = 1, 2, \dots, l$, to which the challenger responds with $\text{ek}_{R_i} \leftarrow \text{mwPRF.KeyGen}(\text{fk}, R_i)$ if x^* has no witness corresponding to the relation R_i ; otherwise stops the game.
 5. Next, \mathcal{A} makes polynomially many queries on instance $\{x_{i_j}\}_{j=1}^{u(i)} \in \mathcal{X}$, for $i = 1, 2, \dots, l$, $u(i)$ is a polynomial in λ . The challenger responds with $\text{mwPRF.F}(\text{fk}, x_{i_j})$ if $x_{i_j} \neq x^*$ and x_{i_j} is an instance of interest corresponding to the relation R_i ; otherwise the challenger terminates the game.
 6. Finally, \mathcal{A} outputs a guess b' for b .
-

Fig. 12. $\text{Expt}_{\mathcal{A}}^{\text{mwPRF}}(1^\lambda, b)$: The security game of a selectively-secure multi-relation witness PRF scheme

- $y \leftarrow \text{mwPRF.F}(\text{fk}, x)$: It is a deterministic algorithm that takes as input a secret function key fk and an instance $x \in \mathcal{X}$, and outputs an element $y \in \mathcal{Y}$ for some set \mathcal{Y} .
- $\text{ek}_R \leftarrow \text{mwPRF.KeyGen}(\text{fk}, R)$: It is possibly a randomized algorithm run by a user having fk , which needs as input a secret function key fk and a relation circuit R , and produces a public evaluation key ek_R corresponding to the relation R .
- $\text{mwPRF.Eval}(\text{ek}_R, x, w) \in \mathcal{Y} \cup \{\perp\}$: It is a deterministic algorithm run by a witness holder that on input an evaluation key ek_R , an instance x and a witness w , outputs an element $y \in \mathcal{Y}$ or \perp .

Correctness: For all $x \in \mathcal{X}$, $w \in \mathcal{W}$, we have that

$$\text{mwPRF.Eval}(\text{ek}_R, x, w) = \begin{cases} \text{mwPRF.F}(\text{fk}, x) & \text{if } R(x, w) = 1 \\ \perp & \text{if } R(x, w) = 0 \end{cases}$$

Definition 12 (Selectively secure multi-relation witness PRF). We say that a multi-relation witness PRF scheme $\text{mwPRF} = (\text{Gen}, \text{F}, \text{Eval})$ for a set of relations $\mathcal{R} = \{R : |R| \leq s, R : \chi \times \mathcal{W} \rightarrow \{0, 1\}\}$, a set \mathcal{Y} , is selectively secure if

$$\left| \Pr[\text{Expt}_{\mathcal{A}}^{\text{mwPRF}}(1^\lambda, 0) = 1] - \Pr[\text{Expt}_{\mathcal{A}}^{\text{mwPRF}}(1^\lambda, 1) = 1] \right| \leq \mu(\lambda)$$

for any $\lambda \in \mathbb{N}$ and every PPT adversary \mathcal{A} in the experiments $\text{Expt}_{\mathcal{A}}^{\text{mwPRF}}(1^\lambda, b)$ defined in Figure 12 where $b \in \{0, 1\}$ and μ is a negligible function of λ . The mwPRF is said to be δ -selectively secure for some specific negligible function $\delta(\cdot)$ if the above indistinguishability gap $\mu(\lambda)$ is smaller than $\delta(\lambda)^{\Omega(1)}$.

A.6 Witness Encryption

Witness encryption was introduced by Garg et al. [15] to encrypt a message with an NP statement and decryption is successful with a valid witness to the statement.

Definition 13 (Witness encryption). A *witness encryption* (WE) scheme for an NP language L with the witness relation $R : \chi \times \mathcal{W} \rightarrow \{0, 1\}$ consists of two algorithms $\text{WE} = (\text{Enc}, \text{Dec})$ satisfying the following:

-
1. The adversary \mathcal{A} chooses $x \in \mathcal{X} \setminus L$, $m_0, m_1 \in \mathcal{M}$ such that $|m_0| = |m_1|$ and sends (x, m_0, m_1, st) to the challenger where st is a state containing some auxiliary information.
 2. The challenger generates $(pp_e, pp_d) \leftarrow \text{OWE.Setup}(1^\lambda, R)$ and sends this to \mathcal{A} .
 3. The challenger selects $b \in \{0, 1\}$ and sends $c_b \leftarrow \text{OWE.Enc}(1^\lambda, x, m_b, pp_e)$ to \mathcal{A} .
 4. The adversary \mathcal{A} outputs a bit b' for b by observing (st, c_b, pp_e, pp_d) .
-

Fig. 13. $\text{Expt}_{\mathcal{A}}^{\text{OWE}}(1^\lambda, b)$: The security game of a selectively-secure offline witness encryption

- $c \leftarrow \text{WE.Enc}(1^\lambda, x, m)$: An encrypter takes as input a security parameter λ , an instance $x \in \mathcal{X}$ and a message $m \in \mathcal{M}$ and outputs a ciphertext c .
- $\text{WE.Dec}(c, w) \in \mathcal{M} \cup \{\perp\}$: A witness holder takes a ciphertext c and a witness $w \in \mathcal{W}$ as input and outputs a message $m \in \mathcal{M}$ or \perp .

Correctness: For any $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$, $(x, w) \in \mathcal{X} \times \mathcal{W}$ such that $x \in L$, $R(x, w) = 1$, we have that

$$\Pr[\text{WE.Dec}(c, w) = m : c \leftarrow \text{WE.Enc}(1^\lambda, x, m)] = 1.$$

Definition 14 (Soundness security of witness encryption). A tuple of algorithms $\text{WE} = (\text{Enc}, \text{Dec})$ is soundness secure witness encryption scheme for an NP language L and a relation $R : \mathcal{X} \times \mathcal{W} \rightarrow \{0, 1\}$, if

$$|\Pr[\mathcal{A}(\text{WE.Enc}(1^\lambda, x, m_0))=1] - \Pr[\mathcal{A}(\text{WE.Enc}(1^\lambda, x, m_1))=1]| \leq \mu(\lambda)$$

for any $x \notin L$, for any PPT adversary \mathcal{A} and messages $m_0, m_1 \in \mathcal{M}$ with $|m_0| = |m_1|$ where μ is a negligible function of λ . The WE scheme is said to be δ -soundness secure for some specific negligible function $\delta(\cdot)$ if the above indistinguishability gap $\mu(\lambda)$ is less than $\delta(\lambda)^{\Omega(1)}$.

A.7 Offline Witness Encryption

Witness encryption scheme with an offline phase [1] reduces time of encryption by shifting the heavy-computing part into a setup algorithm. We note that setup is independent of the statement and message to be encrypted.

Definition 15 (Offline witness encryption). An *offline witness encryption* (OWE) scheme for an NP language L with witness relation $R : \mathcal{X} \times \mathcal{W} \rightarrow \{0, 1\}$ is a tuple of algorithms $\text{OWE} = (\text{Setup}, \text{Enc}, \text{Dec})$ with the following requirements:

- $(pp_e, pp_d) \leftarrow \text{OWE.Setup}(1^\lambda, R)$: This algorithm is run by a trusted third party which takes as input a security parameter λ and publishes a public parameter pp_e for encryption and a public parameter pp_d for decryption.
- $c \leftarrow \text{OWE.Enc}(1^\lambda, x, m, pp_e)$: The encryption algorithm takes as input the security parameter λ , an instance $x \in \mathcal{X}$, a message $m \in \mathcal{M}$ and encryption parameter pp_e . It computes a ciphertext c and broadcasts it over a public channel.
- $\text{OWE.Dec}(c, w, pp_d) \in \mathcal{M} \cup \{\perp\}$: A witness holder on receiving a ciphertext c runs this algorithm using a witness w and decryption parameter pp_d to recover either $m \in \mathcal{M}$ or \perp .

Correctness: For any $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$, $(x, w) \in \chi \times \mathcal{W}$ such that $x \in L$, $R(x, w) = 1$, we have that

$$\Pr \left[\begin{array}{l} \text{OWE.Dec}(c, w, pp_d) = m : (pp_e, pp_d) \leftarrow \text{OWE.Setup}(1^\lambda, R), \\ c \leftarrow \text{OWE.Enc}(1^\lambda, x, m, pp_e) \end{array} \right] = 1. \quad (4)$$

Definition 16 (Selectively secure offline witness encryption). We say that an offline witness encryption $\text{OWE} = (\text{Setup}, \text{Enc}, \text{Dec})$ for an NP language L and a relation $R : \chi \times \mathcal{W} \rightarrow \{0, 1\}$, is selectively secure if

$$|\Pr[\text{Expt}_{\mathcal{A}}^{\text{OWE}}(1^\lambda, 0) = 1] - \Pr[\text{Expt}_{\mathcal{A}}^{\text{OWE}}(1^\lambda, 1) = 1]| \leq \mu(\lambda)$$

for any $\lambda \in \mathbb{N}$ and every PPT adversary \mathcal{A} in the experiments $\text{Expt}_{\mathcal{A}}^{\text{OWE}}(1^\lambda, b)$ defined in Figure 13 where $b \in \{0, 1\}$ and μ is a negligible function of λ . The OWE is said to be δ -selectively secure for some specific negligible function $\delta(\cdot)$ if the above indistinguishability gap $\mu(\lambda)$ is less than $\delta(\lambda)^{\Omega(1)}$.

A.8 Offline Functional Witness Encryption

The notion of functional witness encryption scheme was given by Boyel et al. [8] who established the equivalence between extractable obfuscation and functional witness encryption with extractable security. Abusalah et al. [1] introduced functional witness encryption with a setup algorithm and named it as offline functional witness encryption.

Definition 17 (Offline functional witness encryption). An *offline functional witness encryption* (OFWE) scheme for an NP language L with witness relation $R : \chi \times \mathcal{W} \rightarrow \{0, 1\}$ and a class of functions $\{f_\lambda\}_{\lambda \in \mathbb{N}}$ is a tuple of algorithms $\text{OFWE} = (\text{Setup}, \text{Enc}, \text{Dec})$ with the following requirement:

- $(pp_e, pp_d) \leftarrow \text{OFWE.Setup}(1^\lambda, R)$: A trusted third party runs this algorithm taking input a security parameter λ and publishes a public parameter pp_e for encryption and a public parameter pp_d for decryption.
- $c \leftarrow \text{OFWE.Enc}(1^\lambda, x, (f, m), pp_e)$: The encryption algorithm takes as input a security parameter λ , an instance $x \in \chi$, a function $f \in f_\lambda$, a message $m \in \mathcal{M}$ and encryption parameter pp_e . It outputs a ciphertext c over a public channel. The domain of the function class is $\mathcal{M} \times \mathcal{W}$.
- $\text{OFWE.Dec}(c, w, pp_d) \in \mathcal{M} \cup \{\perp\}$: A witness holder on receiving a ciphertext c runs this algorithm using a witness w and decryption parameter pp_d and recovers either $f(m, w)$ or \perp .

Correctness: For any $\lambda \in \mathbb{N}$, $f \in f_\lambda$, $m \in \mathcal{M}$, $(x, w) \in \chi \times \mathcal{W}$ such that $x \in L$, $R(x, w) = 1$, we have that

$$\Pr \left[\begin{array}{l} \text{OFWE.Dec}(c, w, pp_d) = f(m, w) : (pp_e, pp_d) \leftarrow \text{OFWE.Setup}(1^\lambda, R), \\ c \leftarrow \text{OFWE.Enc}(1^\lambda, x, (f, m), pp_e) \end{array} \right] = 1.$$

Definition 18 (Selectively secure offline functional witness encryption). We say that an offline functional witness encryption $\text{OFWE} = (\text{Setup}, \text{Enc}, \text{Dec})$ for an NP language L with witness relation $R : \chi \times \mathcal{W} \rightarrow \{0, 1\}$ and a function class $\{f_\lambda\}_{\lambda \in \mathbb{N}}$, is selectively secure if

$$|\Pr[\text{Expt}_{\mathcal{A}}^{\text{OFWE}}(1^\lambda, 0) = 1] - \Pr[\text{Expt}_{\mathcal{A}}^{\text{OFWE}}(1^\lambda, 1) = 1]| \leq \mu(\lambda)$$

for any $\lambda \in \mathbb{N}$ and every PPT adversary \mathcal{A} in the experiments $\text{Expt}_{\mathcal{A}}^{\text{OFWE}}(1^\lambda, b)$ defined in Figure 14 where $b \in \{0, 1\}$ and μ is a negligible function of λ . The OFWE is said to be δ -selectively secure for some specific negligible function $\delta(\cdot)$ if the above indistinguishability gap $\mu(\lambda)$ is less than $\delta(\lambda)^{\Omega(1)}$.

-
1. The adversary \mathcal{A} chooses $x \in \mathcal{X}$, $(f_0, m_0), (f_1, m_1) \in f_\lambda \times \mathcal{M}$ such that $f_0(m_0, w) = f_1(m_1, w)$ for all w satisfying $R(x, w) = 1$ and $|(f_0, m_0)| = |(f_1, m_1)|$ and sends (x, m_0, m_1, st) to the challenger where st is a state containing some auxiliary information.
 2. The challenger generates $(pp_e, pp_d) \leftarrow \text{OFWE.Setup}(1^\lambda, R)$ and sends this to \mathcal{A} .
 3. The challenger selects $b \in \{0, 1\}$ and sends $c_b \leftarrow \text{OFWE.Enc}(1^\lambda, x, (f_b, m_b), pp_e)$ to \mathcal{A} .
 4. The adversary \mathcal{A} guesses a bit b' for b by observing (st, c_b, pp_e, pp_d) .
-

Fig. 14. $\text{Expt}_{\mathcal{A}}^{\text{OFWE}}(1^\lambda, b)$: The security game of a selectively-secure offline functional witness encryption

A.9 Compactness and Sub-linear Compactness of Randomized encoding schemes in CRS model

Definition 19 (Compact randomized encoding for Turing machines). A $(\lambda_0, S(\cdot))$ -simulation secure randomized encoding scheme (see Definition 2) is said to be compact if

$$\begin{aligned} \text{Time}_{\text{RE.Enc}}(1^\lambda, \Pi, x, T) &= \text{poly}(\lambda, |\Pi|, |x|, \log T) \text{ and} \\ \text{Time}_{\text{RE.Eval}}(\widehat{\Pi}_x, \text{crs}) &= \text{poly}(\lambda, |\Pi|, |x|, T) \end{aligned}$$

for every security parameter λ , Turing machine Π , input x , time-bound T and every encoding $\widehat{\Pi}_x \leftarrow \text{RE.Enc}(pk, \Pi, x)$ where $(\text{crs}, pk) \leftarrow \text{RE.Setup}(1^\lambda, \cdot)$. Here $\text{Time}_X(\cdot)$ denotes the time-bound of the algorithm X with a specified class of inputs.

Definition 20 (Succinct randomized encoding for Turing machines). A $(\lambda_0, S(\cdot))$ -simulation secure randomized encoding scheme is said to be succinct for a class of Turing machines $\{M_\lambda\}$ if the efficiency requirement for RE.Enc is defined as

$$\text{Time}_{\text{RE.Enc}}(1^\lambda, \Pi, x, T) = l \cdot \text{poly}(\lambda, |\Pi|, |x|, \log T).$$

The notations are the same as in Definition 19

Definition 21 (Sub-linear compactness of randomized encoding for Turing machines). A $(\lambda_0, S(\cdot))$ -simulation secure randomized encoding scheme is said to be sub-linearly compact for a class of Turing machines $\{M_\lambda\}$ if the efficiency requirement for RE.Enc is defined as

$$\text{Time}_{\text{RE.Enc}}(1^\lambda, \Pi, x, T) \leq \text{poly}(\lambda, |\Pi|, |x|) \cdot T^{1-\epsilon}$$

for some $\epsilon \in (0, 1)$. The notations are the same as in Definition 19

Randomized encoding schemes in CRS model can be constructed from a public key functional encryption (PKFE) scheme and a pseudorandom generator [23]. The compactness (respectively, sub-linear compactness) of RE in CRS model depends on the compactness (respectively, sub-linear compactness) of the underlying PKFE. In [3], a weakly sub-linear compact PKFE for \mathbf{P}/poly (i.e. for polynomial size circuits) is constructed using plain public key encryption and strong exponentially-efficient indistinguishability obfuscation (SXIO). They also instantiated SXIO from sub-exponentially secure secret key functional encryption (SKFE) schemes. The existence of a sub-linearly compact randomized encoding scheme for Turing machines follows from the two theorems stated below.

Theorem 6 [3] *Assuming a plain public-key encryption (PKE) and strong exponentially-efficient indistinguishability obfuscation (SXIO) with a small enough constant compressing factor, there exists a weakly sub-linear compact public key functional encryption (PKFE) scheme (for functions with long output).*

Theorem 7 [23] *Assuming hardness of Learning With Error (respectively, sub-exponential hardness), if there exists a selectively secure weakly sub-linear compact public key functional encryption (PKFE) scheme for $P/poly$ (respectively, with sub-exponential hardness), then there exists a sub-linearly compact randomized encoding (RE) scheme for Turing machines in CRS model with (respectively, sub-exponential) simulation security.*

Remark 4 The existence of sub-linearly compact RE scheme for Turing machines is almost directly followed from a succinct RE scheme and a weakly sub-linear compact RE scheme for Turing machines [23]. The succinctness of a RE scheme for Turing machines depends on the succinctness of PKFE for $P/poly$. Using only the sub-exponential hardness of LWE, there exists a succinct PKFE with sub-exponential security for NC^1 [18]. Also, there exist transformations [18,2] from symmetric key encryption with decryption circuit in NC^1 together with succinct PKFE for NC^1 to a succinct PKFE for $P/poly$. We note that if the symmetric key encryption and the succinct PKFE for NC^1 are both sub-exponentially secure then the resulting succinct PKFE for $P/poly$ is also sub-exponentially secure.

A sub-exponentially secure weakly sub-linear compact PKFE is achieved in [3] using SXIO and a public-key encryption scheme with sub-exponential security. Indistinguishability obfuscation ($i\mathcal{O}$) is a technique to make a class of circuits $\{\mathcal{C}_\lambda\}$ unintelligible in the sense that for any circuit $C \in \mathcal{C}_\lambda$, C and $i\mathcal{O}(1^\lambda, C)$ have the same output on all possible inputs and $i\mathcal{O}(1^\lambda, C_0)$ is indistinguishable from $i\mathcal{O}(1^\lambda, C_1)$ for any two circuits $C_0, C_1 \in \mathcal{C}_\lambda$ such that $C_0(x) = C_1(x)$, where λ is the security parameter. We want the size of $i\mathcal{O}(1^\lambda, C)$ bounded by $\text{poly}(\lambda, |C|)$. An SXIO has the same functionality as an indistinguishability obfuscator with nontrivial efficiency. The running time of SXIO on input $(1^\lambda, C)$ is at most $2^{n^\gamma} \cdot \text{poly}(\lambda, |C|)$ where the circuit $C \in \mathcal{C}_\lambda$ takes an input of length n and $\gamma (< 1)$ is the compressing factor. To get such an SXIO with an arbitrary compressing factor, it is required to have a multi-input SKFE [9] which supports an unbounded polynomial number of functional keys. Also, 1-input single-key SKFE suffices to get an SXIO but with a restriction on compressing factor γ satisfying $\frac{1}{2} \leq \gamma \leq 1$ and such an SXIO can be used to build a sub-exponentially secure weakly sub-linear compact PKFE [3].

B Proof of Corollary 1

The existence of sub-exponentially hard LWE and δ -sub-exponentially secure weakly sub-linear compact public key functional encryption scheme for $P/poly$ imply a δ -sub-exponential simulation secure (Definition 2) randomized encoding scheme for Turing machines in CRS model (Theorem 7). We get a δ -secure puncturable PRF (Definition 5) from δ -secure one-way functions (Theorem 5). We need sub-exponentially secure one-way functions for getting a sub-exponentially secure pseudorandom generator (Theorem 4) which is required for constructing the special circuit \mathcal{G} in Figure 1. The corollary follows directly from Theorem 2.

<p>Hardwired: a pPRF key K and a relation R.</p> <p>Input: an instance $x \in \mathcal{X} = \{0, 1\}^k$ and a witness $w \in \mathcal{W} = \{0, 1\}^{n-k}$.</p> <p>Padding: the circuit is padded to size $\text{pad} = \text{pad}_{E_R}(s, d, n, \lambda)$, determined in the analysis.</p> <ol style="list-style-type: none"> 1. if $(R(x, w) = 1)$ then 2. $y \leftarrow \text{pPRF.Eval}(K, x)$ 3. else $y \leftarrow \perp$ 4. end if 5. return y
--

Fig. 15. Evaluation Circuit $E_R = \text{EC}[K, R]$

C Construction of Multi-relation witness PRFs

We will give a generic construction of multi-relation witness PRFs using the same idea involved in the construction of our single relation witness PRFs.

Construction 3. Let $\text{pPRF} = (\text{Gen}, \text{Eval}, \text{Punc})$ be a puncturable pseudorandom function with domain $\{0, 1\}^k$, range \mathcal{Y} and $\text{RE} = (\text{Setup}, \text{Enc}, \text{Eval})$ be a bounded input δ -sub-exponential simulation secure sub-linear compact randomized encoding scheme in CRS model for Turing machines. Our $\text{mwPRF} = (\text{Gen}, \text{F}, \text{KeyGen}, \text{Eval})$ for an NP language L with a set of relations $\mathcal{R} = \{R : |R| \leq s, R : \mathcal{X} \times \mathcal{W} \rightarrow \{0, 1\}, \mathcal{X} = \{0, 1\}^k \text{ and } \mathcal{W} = \{0, 1\}^{n-k}\}$, is given by the following algorithms.

- $\text{fk} \leftarrow \text{mwPRF.Gen}(1^\lambda, s)$: This is run by a user with input a security parameter λ and a bound s on the size of the relations in \mathcal{R} .
 - Choose a pPRF key $K \leftarrow \text{pPRF.Gen}(1^\lambda)$, $K \in \{0, 1\}^\lambda$.
 - Set and output $\text{fk} = K$ as the secret function key. The user keeps fk as secret.
- $y \leftarrow \text{mwPRF.F}(\text{fk}, x)$: This algorithm generates a PRF value $y \leftarrow \text{pPRF.Eval}(K, x)$ by taking input a secret function key $\text{fk} = K$ and an instance $x \in \mathcal{X}$. The value $y \in \mathcal{Y}$ is treated as the mwPRF value corresponding to the statement $x \in \mathcal{X}$.
- $\text{ek}_R \leftarrow \text{mwPRF.KeyGen}(\text{fk}, R)$: This algorithm is used to compute an evaluation key for any given relation $R \in \mathcal{R}$. It works as follows on input a secret function key fk and a relation R :
 - Construct the circuit $E_R \in \{E_\lambda\}$ as described in Figure 15⁵. Let the circuit E_R be of size S with input size n , output size l and runtime bound T .
 - Generate $(\text{crs}_i, \text{pk}_i) \leftarrow \text{RE.Setup}(1^\lambda, 1^S, 1^n, 1^T, 1^l)$ for $i \in \{0, 1, \dots, n\}$ where crs_i is a common reference string and pk_i is an encoding key. Set $\vec{\text{crs}} = \{\text{crs}_i\}_{i=0}^n$ and $\vec{\text{pk}}_i = \{\text{pk}_j\}_{j=i}^n$.
 - Compute the randomized encoding $\tilde{\Pi}[\vec{\text{pk}}_1, E_R, \epsilon, \alpha] \leftarrow \text{RE.Enc}(\text{pk}_0, \Pi[\vec{\text{pk}}_1, E_R, \epsilon, \alpha], \epsilon)$ where ϵ is a null string, α is a random binary string and $\Pi[\vec{\text{pk}}_1, E_R, \epsilon, \alpha]$ is a Turing machine defined in Remark 1.
 - Build the special circuit $\mathcal{G}[\tilde{\Pi}[\vec{\text{pk}}_1, E_R, \epsilon, \alpha], \vec{\text{crs}}]$ as described in Figure 1 and output $\text{ek}_R = \mathcal{G}[\tilde{\Pi}[\vec{\text{pk}}_1, E_R, \epsilon, \alpha], \vec{\text{crs}}]$.
- $\text{mwPRF.Eval}(\text{ek}_R, x, w)$: An entity having a witness $w \in \mathcal{W}$ corresponding to an instance $x \in \mathcal{X}$, runs this algorithm using an evaluation key $\text{ek}_R = \mathcal{G}[\tilde{\Pi}[\vec{\text{pk}}_1, E_R, \epsilon, \alpha], \vec{\text{crs}}]$ and outputs $\mathcal{G}[\tilde{\Pi}[\vec{\text{pk}}_1, E_R, \epsilon, \alpha], \vec{\text{crs}}](z)$ where $z = (x, w)$.

⁵ The only difference from the circuit E (Figure 2) is that, E_R is now hardcoded with a relation R that can vary with evaluation key ek_R .

Correctness. We note that our $mwPRF.F(\text{fk}, x)$ is a pPRF evaluation on $x \in \mathcal{X}$ and one can only use $mwPRF.Eval$ with an evaluation key ek_R if he has a valid witness w for x such that $R(x, w) = 1$ as the circuit E_R is hardcoded with the relation circuit R . The correctness of this scheme follows from a similar argument discussed in the correctness of Construction 1.

Padding Parameter. We take $\text{pad}_{E_R}(s, d, n, \lambda) \leq \text{poly}(\lambda, k, s)$ due to a similar argument as in the case of our single relation witness PRF in Construction 1.

Efficiency. The efficiency of our multi-relation witness PRF is the same as that of our single relation witness PRF in Construction 1.

Theorem 8 *Assuming LWE with sub-exponential hardness and the existence of δ -sub-exponentially secure one-way functions, if there exists a weakly sub-linear compact public key functional encryption scheme for P/poly with δ -sub-exponential security, then there exists a δ -secure multi-relation witness PRF scheme.*

We skip the proof of this theorem as it is similar to the proof of Theorem 2.

Remark 5 Our $mwPRF$ computes a randomized encoding of a Turing machine $\Pi_{[\vec{pk}_1, E_R, \epsilon, \alpha]}$ to get an evaluation key ek_R for a relation R . Whereas, the generation of evaluation key of [25] needs to compute a multilinear map with the multilinearity level equal to the size of the description of the corresponding relation R which makes evaluation key ek_R computationally more expensive than ours.