Constructing Witness PRF and Offline Witness Encryption Without Multilinear Maps

Tapas Pal, 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), introduced by Zhandry [Zha16], was defined for an NP language \( L \) and generate a pseudorandom value for any instance \( x \). The same pseudorandom value can be obtained efficiently using a valid witness \( w \) for \( x \in L \). Zhandry built a subset-sum encoding scheme from multilinear maps and then converted a relation circuit corresponding to an NP language \( L \) to a subset-sum instance to achieve a witness PRF for \( L \). The main goal in developing witness PRF in [Zha16] is to avoid obfuscation from various constructions of cryptographic primitives. Reliance on cryptographic tools built from multilinear maps may be perilous as existing multilinear maps are still heavy tools to use and suffering from many non-trivial attacks.

In this work, we give constructions of the following cryptographic primitives without using multilinear maps and instantiating obfuscation from randomized encoding:

- We construct witness PRFs using a puncturable pseudorandom function and sub-exponentially secure randomized encoding scheme in common reference string (CRS) model. 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.
- We turn our witness PRF into a multi-relation witness PRF where one can use the scheme with a class of relations related to an NP language.
- Furthermore, we construct an offline witness encryption scheme using any extractable witness PRF. The offline witness encryption scheme of Abusalah et al. [AFP16] was built from a plain public-key encryption, a statistical simulation-sound non-interactive zero knowledge (SSS-NIZK) proof system and obfuscation. In their scheme, a(n) SSS-NIZK proof is needed for the encryption whose efficiency depends on the underlying public key encryption. We replace SSS-NIZK by extractable witness PRF and construct an offline witness encryption scheme. More precisely, our scheme is based on a public-key encryption, a witness PRF and employs a sub-exponentially secure randomized encoding scheme in CRS model instantiating obfuscation. Our offline witness encryption can be turned into an offline functional witness encryption scheme where decryption releases a function of a message and witness as output.

Keywords: Witness PRF, Offline witness encryption, Randomized encoding.

1 Introduction

Witness PRF. Witness pseudorandom function (witness PRF) is a relatively new cryptographic primitive introduced by Zhandry [Zha16] to avoid obfuscation in various cryptographic applications 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 that do not need to hide a programme $P$ completely. For instance, the Boneh-Zhandry [BZ17] protocol of multiparty non-interactive key exchange without a trusted setup needs a program $P$ to be obfuscated where $P$ computes a pseudorandom function using a secret key on its inputs. Zhandry has shown how obfuscation can be avoided by introducing witness PRF.

A witness PRF for an NP language $L$ enables one to compute a pseudorandom function $f$ on a statement $x$ without a secret key if (s)he has a valid witness for $x \in L$ where $f(x)$ is indistinguishable from a random element if $x \notin L$. More specifically, witness PRF first generates a pair of keys $(fk, ek)$ based on a relation circuit $R$ corresponding to an NP language $L$ where $fk$ is the function secret key and $ek$ is the function public key. A user with $fk$ computes a pseudorandom value $F(fk, x)$ for any input $x$ while a witness holder can obtain the same pseudorandom value using $\text{Eval}(ek, x, w)$ if $w$ is a valid witness for $x \in L$.

Witness PRF is closely related to constrained PRFs and smooth projective hash functions (SPHFs). In constrained PRF, we can generate multiple keys for different circuits. In witness PRF, we generate only one key $ek$ for a given relation circuit $R$ in the setup phase. Existing constructions of SPHFs cannot handle arbitrary NP languages while witness PRFs support any NP language. There is another variant of witness PRF which is extractable in nature. In extractable witness PRF, existence of an adversary capable of distinguishing $F(fk, x)$ from a random value implies existence of a polynomial time extractor which produces a witness for the instance $x$ if $x \in L$. There is another notion related to witness PRF which is flexible for a class of relations with a specified size bound and is called multi-relation witness PRF. Here we generate different keys for each relation and the security is based on the indistinguishability of $F(fk, x)$ from a random element if $x$ has no valid witness relative to any of the queried relations.

Applications of witness PRF. There are many efficient tools that can be built from witness PRFs and extractable witness PRFs. We mention the schemes where Zhandry uses witness PRFs fending off obfuscation [Zha16]:

- Boneh and Zhandry [BZ17] designed the first multi-party key exchange (MIKE) protocol (for $n$-users with $n > 3$) using obfuscation that does not require a trusted party or setup. Obfuscation can be replaced with witness PRFs to obtain such an MIKE.

- Bellare et al. [BST14] constructed a hardcore function of arbitrary output size for any one-way function using differing inputs obfuscation (or extractable obfuscation). Witness PRFs suffices here.

- It has been seen that obfuscation implies witness encryption [GGSW13]. Zhandry introduced and constructed re-usable witness encryption from witness PRFs. Re-usable witness encryption has a special feature of producing very short ciphertexts with size proportional to the security parameter and independent of the size of the relation. This re-usable witness encryption can be used in the transformation of [GGSW13] to get an attribute-based encryption with similar short ciphertexts.

- Witness PRFs also replace obfuscation in secret sharing [KNY17] and fully distributed broadcast encryption scheme [Zha16].

Constructing witness PRF of [Zha16]. The only existing construction of witness PRF [Zha16] is based on subset-sum encoding scheme. A subset-sum encoding scheme corresponding to a (multi-)set of integers $S$ is capable of encoding any integer secretly such that a public evaluation function on input a subset $T$ of $S$ can compute an encoding of the integer $t = \sum_{i \in T} s_i$. In this scheme, the subset-sum encoding scheme is used to encode the relation circuit's output as a subset of $S$ and the witness PRF is used to compute the pseudorandom value. The security of this construction is based on the subset-sum problem, which is believed to be hard to solve even for small values of $S$.
$\sum_{i \in T} i$. In [Zha16], a new notion of subset-sum encoding is introduced from multilinear maps [GGH13, CLT13] and the security is based on the fact that if there does not exist a subset of $S$ for which the target sum $t$ is achieved then the encoding $\hat{t}$ is indistinguishable from a random element. The hardness of this problem gives rise to a new complexity assumption (based on multilinear maps) which was named as the *multilinear subset-sum Diffie-Hellman assumption*. Witness PRF is instantiated from the subset-sum encoding. Consequently, it is based on multilinear maps and the security depends upon *multilinear subset-sum Diffie-Hellman assumption*. We emphasize that the pseudorandom value of the witness PRF cannot be immediately computed for any instance $x$, rather a reduction procedure is followed where an instance $x$ of an NP language $L$ is converted into a subset-sum instance. The reduced subset-sum instance depends on $L$ and the size of instance $x$.

All currently known constructions of multilinear maps [GGH13, CLT13] are only approximations to the ideal multilinear maps and the noise increases with the number of multiplications and pairing operations. The multilinearity level of witness PRF increases with the number of gates used in the relation circuit corresponding to the NP language. Existing witness PRF [Zha16] is approximate in the sense that the underlying subset-sum encoding is based on multilinear maps and thereby approximate and noisy. Furthermore, complications arise when the size of the relation circuit grows.

Zhandry showed the hardness of the *multilinear subset-sum Diffie-Hellman assumption* in the generic multilinear map model [Zha16]. The recent line of attacks on multilinear maps [CLT14a, CHL+14, BWZ14b, GHMS14b] breaks many useful assumptions and hence threats to the cryptosystems where security is based on complexity assumptions related to multilinear maps. Therefore, one may not want to rely on cryptographic tools that are instantiated from multilinear maps.

**Witness encryption.** *Witness encryption* (WE) was introduced by Garg et al. [GGSW13]. There are many applications of WE in secret sharing, identity based encryption, attribute-based encryption, asymmetric password based encryption, differing inputs obfuscation ([GGSW13, GKP+13b, BH15, BCP14]). In a plain public-key encryption (PKE) scheme, 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 $x \in L$. *Functional witness encryption* was introduced by Boyle et al. [BCP14] where a decrypter can only learn a function of the message if a valid witness for the instance is known. The equivalence of functional WE and differing inputs obfuscation was also observed in [BCP14].

WE with an additional setup phase is called *offline witness encryption* (OWE) [AFP16]. In OWE, the heavy-duty part is done by a trusted third party in an offline phase making encryption more efficient than the existing WE constructions. The only OWE construction [AFP16] uses a standard public key encryption and a statistically simulation-sound non-interactive zero knowledge (SSS-NIZK) proof system to produce a ciphertext and an obfuscated circuit created in the setup phase is used for decryption.

Mostly, WE have been constructed using either multilinear maps or using obfuscation directly. Impracticality or computationally expensive nature of multilinear maps and obfuscation have made all these WE schemes unusable in practical devices.

**Our contribution and technical overview.** In this work, we construct a witness PRF without using multilinear maps and an offline witness encryption without SSS-NIZK. Our construction of witness PRFs is inspired by the puncturable programming technique introduced by Sahai and Waters [SW14]. They used indistinguishability obfuscation (iO) with puncturable pseudorandom function to built many interesting cryptographic tools. The job of iO is to make
a class of circuits \( \{ C_\lambda \}_{\lambda \in \mathbb{N}} \) unintelligible in such a way that for any circuit \( C \in \mathcal{C}_\lambda \) for some \( \lambda \in \mathbb{N} \), we have \( iO(1^\lambda, C)(x) = C(x) \). The security demands that given two equivalent circuits (or Turing machines) \( C_0, C_1 \in \mathcal{C}_\lambda \), one cannot distinguish between \( iO(1^\lambda, C_0) \) and \( iO(1^\lambda, C_1) \). Instead of using \( iO \) directly, we look into recent developments on \( iO \) from functional encryption \([\text{AJ15, BV15, AJ15, BNPW16, KNT17, LPST16b}]\) which is a relatively weaker primitive.

In our constructions of witness PRFs and offline witness encryption, we integrate the technique of getting \( iO \) from randomized encoding scheme in common reference string (CRS) model \([\text{LPST16b}]\).

We built our witness PRFs by coupling a puncturable pseudorandom function (pPRF) and a sub-exponentially secure sub-linearly compact randomized encoding (RE) scheme in CRS model. The functional secret key \( fK \) is a pPRF key \( K \) and the functional evaluation key \( eK \) consists of \( crs = \{ crs_i \}_{i=0}^n \) and a randomized encoding of an input less Turing machine \( \Pi[pk_1, E, \epsilon, \alpha] \) (defined in Remark 2, section 2) where the hardcoded elements are \( pk_1 = \{ pk_j \}_{j=1}^n \), a circuit \( E \), the null string \( \epsilon \) and a randomly chosen bit-string \( \alpha \). The \( n+1 \) pairs of common reference string-encoding key \( (crs_i, pk_i) \) for \( i = 0, 1, 2, \ldots, n \) are generated in the setup phase of RE. Here \( n \) denotes the total size of the instance-witness pair. The circuit \( E \) on taking input an instance-witness \((x, w)\) pair, verifies that \( w \) is a valid witness for the instance \( x \) and then outputs the evaluation of pPRF with input \( x \) using the key \( K \).

Our witness PRF computes the pseudorandom value corresponding to some \( x \) as the pPRF evaluation of \( x \) using the functional secret key \( fK \). A valid witness holder can recover the same pseudorandom value using the functional evaluation key \( eK \) by computing \( G[\Pi[pk_1, E, \epsilon, \alpha], crs](x, w) \) where \( G \) is the special circuit (defined in Figure 6, section 2) that recursively computes the evaluation phase of RE to achieve \( E(x, w) \).

A pPRF can be constructed from one-way functions \([\text{GGM86, BW13, BGI14}]\) and a sub-exponentially secure sub-linear compact RE scheme in CRS model can be instantiated from a sub-exponentially secure weakly sub-linear compact public key functional encryption (PKFE) scheme and assuming sub-exponential hardness of learning with error (LWE) assumption \([\text{BNPW16, LPST16b}]\). We achieve the following result.

**Theorem 1.** (Informal) Assuming LWE with sub-exponential hardness and the existence of sub-exponentially secure one-way functions, if there exists a weakly sub-linear compact public key functional encryption (PKFE) scheme with sub-exponential security, then there exists a secure witness PRF scheme.

We note the following advantages of our scheme over the witness PRF scheme of Zhandry \([\text{Zha16}]\):

- Zhandry used subset-sum encoding to build witness PRF and subset-sum encoding is constructed from multilinear maps. Our scheme is established from a puncturable pseudorandom function and a randomized encoding scheme in CRS model which does not use multilinear maps. Therefore our scheme is more reliable in light of the recent attacks on multilinear maps. Security of our witness PRF is based on sub-exponentially secure one-way functions, sub-exponentially hard of LWE assumption and sub-exponentially secure PKFE. Note that PKFE are well studied and secure tools compared to existing multilinear maps which are noisy, approximation to ideal multilinear maps and vulnerable to many attacks.

- The security proof of the witness PRF of \([\text{Zha16}]\) relies on a newly suggested multilinear subset-sum Diffie-Hellman assumption which is instance dependent non-standard assumption with hardness proved in the generic multilinear maps model. In contrast, we achieve
security by showing indistinguishability between hybrid sequences. Our proof is instance
independent and does not rely on any such non-standard assumptions.

- The multilinearity level increases at least linearly with the number of gates used in the
relation circuit of the underlying NP language in the construction of Zha16 and the
efficiency of their scheme also decreases with the size of the relation circuit. On the other
hand, our scheme uses a randomized encoding scheme that essentially executes decryption
procedure of a PKFE scheme for \((n + 1)\) times [LPST16b] where \(n\) denotes the total size of
the instance and witness. Our construction provides more efficient witness PRF evaluation
than the scheme of Zhandry.

As an application of witness PRF, we present an offline witness encryption (OWE) scheme
encouraged by the construction of Abusalah et al. [AFP16]. They used a standard public key
encryption (PKE) scheme to encrypt a message \(m\) together with an instance \(x\) of an NP lan-
guage \(L\) twice with two different randomness to produce two different ciphertexts \(c_1\) and \(c_2\).
Besides, they generate a(n) SSS-NIZK proof \(\pi\) of the statement that \(c_1, c_2\) encrypt the same
message-instance pair. The resulting ciphertext of their OWE is \((x, c_1, c_2, \pi)\). They have in-
stantiated the encryption of their OWE using an ElGamal encryption scheme for the PKE
and established a(n) SSS-NIZK proof \(\pi\) of the statement that “\(\text{two ElGamal ciphertexts } c_1, c_2\)
encrypt the same message” via Gorth-Sahai proofs (GS-proofs) [GS08]. Here 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 equa-
tions over some groups. The ElGamal ciphertexts can be represented in a way to get a set
of pairing product equations that supports the above statement and a(n) SSS-NIZK proof can
be ensured using the GS-proofs for those equations. Therefore, for practical use of the OWE
scheme of [AFP16], we need to carefully choose the PKE scheme so that a(n) SSS-NIZK proof
can be achieved through the GS-proofs. If some other PKE scheme is used for more efficiency
or security then one may face problem in generating such a proof for the abovementioned state-
ment as there exists no efficient SSS-NIZK proof system for any general relation (according to
our knowledge).

We try to fix this limitation of the OWE scheme of [AFP16] by replacing SSS-NIZK with
any extractable witness PRF which can efficiently handle any relation. Our OWE is built with
a plain PKE scheme and a secure witness PRF for encryption. In the setup phase, we generate
two pairs of secret-public keys \((SK_1, PK_1)\), \((SK_2, PK_2)\) from the key generation algorithm
of the PKE and a pair of functional secret key, evaluation key \((fk, ek)\) for the witness PRF which
corresponds to the NP statement that \(\text{two given ciphertexts obtained from the PKE scheme}
encrypt the same message}\). The public parameters for encryption consist of \(PK_1, PK_2\) and \(ek\).
Our OWE encryption first computes two ciphertexts \(c_1, c_2\) encrypting the same instance-message
pair \((x, m)\) under the two public keys \(PK_1, PK_2\), then utilizing the evaluation key \(ek\) executes
the evaluation process of witness PRF to produce a pseudorandom string \(y\) for the statement
that \(c_1, c_2\) encrypt the same message. The OWE ciphertext components for an NP language \(L\)
are \(c_1, c_2, x\) and \(y\).

For decryption, we need to compute a circuit \(C\) in the setup phase of OWE. The circuit \(C\)
on input \((c_1, c_2, x, y)\) and a witness \(w\) for \(x \in L\) works as follows:

- Use the functional secret key \(fk\) for the statement that \(c_1, c_2\) encrypt the same message to
get a value \(y'\) and check if \(y' = y\).
- Check whether \(w\) is a valid witness for the statement \(x\).
- If both the checks pass, then decrypt \(c_1\) using \(SK_1\) and obtain \((x', m')\).
Output $m'$ if $x = x'$.

As in the case of evaluation procedure of our witness PRF scheme, here also we pick the RE scheme in CRS model of [LPST16b] to get a randomized encoding of an input less Turing machine $\Pi[pk_1, C, \epsilon, \alpha]$ and use the special circuit $G[\overline{\Pi[pk_1, C, \epsilon, \alpha]}, crs]$ that corresponds to the circuit $C$ as the public parameter for decryption. The decryption algorithm of our OWE scheme computes the circuit $G$ on an input $(c, w)$ to recover the original message $m$ if $w$ is a valid witness for $x \in L$, where $c = (c_1, c_2, x, y)$. In the OWE scheme of [AFP16], they managed decryption with an obfuscated circuit and claimed that the decryption phase is less efficient than that of the reusable WE scheme of [Zha16]. We employ the randomized encoding technique of [LPST16b] to achieve an efficient decryption for our OWE scheme where the decryption time is polynomial in the size of the circuit $C$ and the total size of ciphertext and witness.

To prove our OWE scheme is secure, we need to consider the extractable security of the underlying witness PRF scheme. Unfortunately, we do not know how to construct a polynomial time extractor which can extract a witness using a challenge statement $x^*$, a witness PRF value $F(fk, x^*)$ and some auxiliary information. Extractable security requires that an adversary cannot distinguish $F(fk, x^*)$ from a random value unless a valid witness to the statement $x^*$ is known to him. In this application of witness PRF for OWE, we note that an adversary does not have a witness for the statement "$c_1, c_2$ encrypt the same message" as the randomness used at the time of encryption which is a part of the witness is computationally hidden in the ciphertexts of the PKE. In this context, we note that the witness PRF of [Zha16] was assumed to be extractable under the extracting subset-sum Diffie-Hellman assumption, rather fabricating an extractor for the witness PRF. Assuming our witness PRF is extractable we arrive at the following result.

**Theorem 2.** (Informal) Assuming existence of sub-exponentially secure one-way functions, a secure public-key encryption (PKE), an extractable witness PRF and a sub-exponential simulation secure randomized encoding (RE) scheme in CRS model for Turing machines, there exists a secure offline witness encryption (OWE) scheme.

We can also transform our OWE into an offline functional witness encryption (OFWE) scheme where a decryption outputs a function of message and witness instead of only message. The encryption algorithm of our OFWE takes a function $f$ as an additional input and encrypts the pair $(x, m')$ using the PKE scheme to produce two ciphertexts $c_1, c_2$ under two public keys $PK_1, PK_2$ where $m' = (f, m)$. For decryption, we use the same circuit $C$ except that we get $(x', m')$ by decrypting $c_1$ using $SK_1$ and then output $f(m, w)$ if $x = x'$ and $w$ is a valid witness for $x \in L$. The security demands that an adversary should not distinguish between the encryptions of $(x, (f_0, m_0))$ and $(x, (f_1, m_1))$ if $f_0(m_0, w) = f_1(m_1, w)$ for all valid witness $w$ for $x \in L$. Our OFWE is secure under the same assumptions described in Theorem 2.

We further convert our single relation witness PRF scheme into a multi-relation witness PRF. In multi-relation witness PRF (mwpRF), a size bound of the relation circuits supported by the scheme is pre-specified depending on which we generate a secret functional key $fk$ that is used for every relation. We use a randomly chosen pRF secret key for $fk$ as in our single relation scheme. The generation of evaluation key $ek_R$ corresponding to a relation $R$ is similar to that in our single relation witness PRF except the fact that the circuit $E$ is hardcoded with the relation $R$ in the key generation phase. We achieve the following result.

**Theorem 3.** (Informal) Assuming LWE with sub-exponential hardness and the existence of sub-exponentially secure one-way functions, if there exists a weakly sub-linear compact public key functional encryption (PKFE) scheme with sub-exponential security, then there exists a secure multi-relation witness PRF scheme.
Our mwPRF inherits the same advantages as of our single relation witness PRF over the scheme of [Zha16]. Additionally, the generation of evaluation key of [Zha16] 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.

Related works. The concept of witness encryption was introduced by Garg et al. [GGSW13]. They gave two candidate constructions of witness encryption for NP-complete exact-cover problem. One is based on a multilinear group family and other uses a graded encoding system. The security of their schemes depends on the decision multilinear no-exact-cover assumption and the decision graded encoding no-exact-cover assumption.

In [GKP13], Goldwasser et al. have shown how to obtain extractable witness encryption scheme using the construction of [GGSW13]. They developed attribute-based encryption (ABE) schemes for any polynomial time Turing machines and Random Access Machines (RAM) employing an extractable witness encryption, a succinct argument of knowledge and an existentially unforgeable signature as the ingredients. Additionally, they have constructed a (single-key and succinct) functional encryption scheme coupling their ABE-scheme for Turing machines with a fully homomorphic encryption scheme.

Boyle, Chung and Pass [BCP14] initiated the study of extractability obfuscation. They constructed an extractability obfuscator for all non-uniform polynomial-time Turing machines. A new notion of functional witness encryption was introduced where decryption gives a function of a message when a valid witness for the instance is known to the recipient of the ciphertext. In this work, it is shown that functional witness encryption is, in fact, equivalent to extractability obfuscation.

Witness encryption with soundness security was introduced by Garg et al. [GGSW13]. Bellare and Hoang [BH15] found that the soundness security of witness encryption scheme does not suffice for the security of the applications in [GGSW13]. The gap can be filled by the new security notion called adaptive soundness security. In this security model, given a security parameter $\lambda$, an adversary produces an instance $x$, a challenge pair of messages $(m_0, m_1)$ and a state $St$. After receiving a ciphertext of message $m_b$ from the challenger where $b$ is chosen randomly from $\{0, 1\}$, the adversary has to guess for $b$ provided the given instance does not belong to the corresponding language. In this work [BH15], Bellare and Hoang established a way to achieve the adaptive soundness security of witness encryption from indistinguishability obfuscation.

In [GGHW17], Garg et al. came up with implausibility results on extractable witness encryption with auxiliary input and general-purpose $diO$ (differing-inputs obfuscation) by assuming the existence of a special-purpose obfuscation. In particular, a specific circuit $C^*$ with special auxiliary input $aux^*$ cannot be obfuscated in a way that hides some specific information. However, the existence of such a special-purpose obfuscation is a falsifiable assumption which they did not able to show how to break for candidate obfuscation schemes.

Aritra and Hnada [AH14] constructed a witness encryption scheme based on multilinear maps. They took the problem of existence of Hamilton cycle in a huge graph. The security proof is based on a generic colored matrix model as defined in the work of candidate indistinguishability obfuscation [GGH+16].

One of the main limitations of [GGSW13] is that the candidate had no proof of security (other than essentially assuming the scheme is secure). In [GLW14], Gentry, Lewko and Waters introduced positional witness encryption which provides a proof reduction of a witness encryption scheme via a sequence of $2^n$ hybrid experiments where $n$ is witness length of the NP-statement.
Liu, Kakvi and Warinschi [LKWI5] constructed a witness encryption scheme based on multilinear maps where they achieved extractable security without obfuscation. In particular, they presented the scheme for a special subset-sum problem. To encrypt any instance of an NP language, they needed to reduce it from conjunctive normal form-satisfiability problem (CNF-SAT) to the special subset-sum problem.

Derler and Slamanig [DS] uses SPHFs to build a practical witness encryption scheme for algebraic languages defined over bilinear groups. Their witness encryption scheme is compatible with the statements used in Groth-Sahai proofs.

**Organization.** The rest of the paper is arranged as follows. In section 2, we provide definitions of some cryptographic tools that are related to our work. Next section 3 contains our construction of witness PRFs and the security requirements. We present our offline witness encryption scheme in section 4. In section 5, we show that a multi-relation witness PRF can be instantiated from the witness PRF developed in section 3. Finally, we conclude in section 6.

2 Preliminaries

We use the notations in Table 1 throughout this paper.

<table>
<thead>
<tr>
<th>$a \leftarrow A$</th>
<th>$a$ is an output of the procedure $A$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a \leftarrow X$</td>
<td>$a$ is chosen uniformly at random from set $X$.</td>
</tr>
<tr>
<td>negligible function</td>
<td>$\mu : \mathbb{N} \rightarrow \mathbb{R}$ is a negligible function if $\mu(n) \leq \frac{1}{n^c}$ holds for every polynomial $p(\cdot)$ and all sufficiently large $n \in \mathbb{N}$.</td>
</tr>
<tr>
<td>$\lambda$, $S(\cdot)$-indistinguishability</td>
<td>Two ensembles ${X_\lambda}$ and ${Y_\lambda}$ are $\lambda$, $S(\cdot)$-indistinguishable means $</td>
</tr>
<tr>
<td>$\delta$-sub-exponential indistinguishability</td>
<td>Two ensembles ${X_\lambda}$ and ${Y_\lambda}$ are $\delta$-sub-exponential indistinguishable means $</td>
</tr>
<tr>
<td>Expt($1^\lambda$, 0) $\approx_{\delta}$ Expt($1^\lambda$, 1)</td>
<td>For any polynomial size distinguisher $D$, the advantage $\Delta =</td>
</tr>
</tbody>
</table>

**Table 1: Notations**

2.1 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 [GGMS06] 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 1.** (Pseudorandom function). A pseudorandom function (PRF) is a function $F : \{0,1\}^\lambda \times \mathcal{X} \rightarrow \mathcal{Y}$ with polynomial runtime satisfying

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

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

8
2.2 Puncturable Pseudorandom Function

Sahai and Waters [SW14] 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 2.** (Puncturable pseudorandom function). A puncturable pseudorandom function (pPRF) consists of a tuple of algorithms pPRF = (Gen, Eval, Punc) over the domain \( X \) and range \( Y \) 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 \( \lambda \) 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 \( x \in X \), outputs the PRF value \( y \in Y \).
- \( K\{S\} \leftarrow \text{pPRF.Punc}(K, S) \): It is a deterministic algorithm that takes a secret key \( K \) and a polynomial-size set \( S \subset X \) as input and outputs a punctured key \( K\{S\} \). If \( S \) contains a single element, say \( x \), then we simply write \( K\{S\} \) as \( K\{x\} \).

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

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

**Definition 3.** (Pseudorandomness at punctured points). We say that a puncturable pseudorandom function pPRF = (Gen, Eval, Punc) preserves pseudorandomness at punctured points if

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

for every PPT adversary \( A \) and any polynomial-size subset \( S \) of \( X \), where \( K \leftarrow \text{pPRF.Gen}(1^\lambda) \), \( K\{S\} \leftarrow \text{pPRF.Punc}(K, S) \), \( U \) denotes the uniform distribution over \( Y \) and \( \mu \) is a negligible function in \( \lambda \). The pPRF is said to be \( \delta \)-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 [BW13, BGI14] that puncturable PRFs can be constructed from one-way functions using the GGM tree-based construction of PRFs [GGM86] where the size of the punctured key grows polynomially with the number of elements in the set \( S \). The GGM construction [GGM86] 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.** [GGM86, Lev87] 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.** [GGM86, BW13, BGI14] Assuming the existence of one-way functions, there exists an efficiently computable puncturable pseudorandom function for any desired poly-size input length.
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 $c_b^b \leftarrow \text{PKE.Enc}(\text{PK}, m_b^b; r)$ to $\mathcal{A}$.

4. The adversary $\mathcal{A}$ observes $c_b^b$ and $st$ and outputs a guess $b'$ for $b$.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig1.png}
\caption{$\text{Expt}_{\mathcal{A}}^{\text{PKE}}(1^\lambda, b)$: The security game of a CPA-secure public-key encryption}
\end{figure}

### 2.3 Public-Key Encryption

#### Definition 4. (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 $\lambda$ as input. It outputs a public key PK and a secret key SK. A user who obtains a key pair $(\text{PK}, \text{SK})$ 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.

- $\text{PKE.Dec}(\text{SK}, c) \in \mathcal{M} \cup \{\bot\}$: The recipient of a ciphertext $c$ runs this algorithm using the secret key SK and gets either a message $m \in \mathcal{M}$ or $\bot$ where $\bot$ indicates a failure of the algorithm.

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

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

#### Definition 5. (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 1 where $b \in \{0, 1\}$ and $\mu$ is a negligible function of $\lambda$. The PKE is said to be $\delta$-selectively secure for some specific negligible function $\delta(\cdot)$ if the above indistinguishability gap $\mu(\lambda)$ is less than $\delta(\lambda)\Omega(1)$.

### 2.4 Witness PRF

Informally, a witness PRF scheme [Zha16] 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 6. (Witness PRF). A witness PRF ($w$PRF) for an NP language $L$ with the witness relation $R : \chi \times \mathcal{W} \rightarrow \{0, 1\}$ consists of three algorithms $w$PRF = $(\text{Gen}, F, \text{Eval})$ and works as follows:
1. The adversary $\mathcal{A}$ chooses a single challenge query on an instance $x^* \in \chi \setminus L$ to the challenger.

2. The challenger generates $(f_k, e_k) \leftarrow w\text{PRF}.\text{Gen}(1^\lambda, R)$ and gives $e_k$ to the adversary $\mathcal{A}$.

3. The challenger computes $y_0 \leftarrow w\text{PRF}.F(f_k, x^*)$, selects $y_1 \leftarrow \mathcal{Y}$ and sends $y_0, y_1$ to $\mathcal{A}$ for a randomly chosen $b \in \{0, 1\}$.

4. The adversary $\mathcal{A}$ makes polynomially many queries on instance $x_i \in \chi$, $i = 1, 2, \ldots, u$, to which the challenger responses with $w\text{PRF}.F(f_k, x_i)$ if $x_i \neq x^*$; otherwise terminates the game.

5. Then $\mathcal{A}$ outputs a guess $b'$ for $b$.

---

**Figure 2:** $\text{Expt}_{\mathcal{A}}^{w\text{PRF}}(1^\lambda, b)$: The security game of a selectively-secure witness PRF

- $(f_k, e_k) \leftarrow w\text{PRF}.\text{Gen}(1^\lambda, R)$: A trusted authority\(^1\) takes as input the security parameter $\lambda$ and a relation circuit $R : \chi \times \mathcal{W} \rightarrow \{0, 1\}$ and randomly generates a secret function key $f_k$ and a public evaluation key $e_k$. A user receiving $(f_k, e_k)$ through a secure channel, keeps $f_k$ as a secret key and publishes $e_k$. We note that $R(x, w) = 1$ if and only if $w$ is a valid witness for $x \in L$.

- $y \leftarrow w\text{PRF}.F(f_k, x)$: Using a function key $f_k$ and an input $x \in \chi$, the user runs this algorithm which deterministically outputs some $y \in \mathcal{Y}$.

- $w\text{PRF}.\text{Eval}(e_k, x, w) \in \mathcal{Y} \cup \{\bot\}$: An witness holder runs this algorithm using an evaluation key $e_k$, an input $x \in \chi$ and a witness $w \in \mathcal{W}$ and deterministically recovers either $y \in \mathcal{Y}$ or $\bot$.

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

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

**Definition 7.** (Selectively secure witness PRF). We say that a witness PRF scheme $w\text{PRF} = (\text{Gen}, F, \text{Eval})$ for an NP language $L$, a relation $R : \chi \times \mathcal{W} \rightarrow \{0, 1\}$, a set $\mathcal{Y}$, is selectively secure if

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

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 2 where $b \in \{0, 1\}$ and $\mu$ is a negligible function of $\lambda$. The $w\text{PRF}$ is said to be $\delta$-selectively secure for some specific negligible function $\delta(\cdot)$ if the above indistinguishability gap $\mu(\lambda)$ is smaller than $\delta(\lambda)^{\Omega(1)}$.

**Definition 8.** (Extractable witness PRFs). A witness PRF scheme $w\text{PRF} = (\text{Gen}, F, \text{Eval})$ 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

$$\left| \Pr \left[ \mathcal{A}^{w\text{PRF}.F(f_k, \cdot)(e_k, x^*, \text{Aux}, y^*) = 1 : (f_k, e_k) \leftarrow w\text{PRF}.\text{Gen}(\lambda, R) \right] - \Pr \left[ \mathcal{D}^{w\text{PRF}.F(f_k, \cdot)(e_k), y^* \leftarrow w\text{PRF}.F(f_k, x^*)} \right] \right|$$

\(^1\)We note that a user may itself run this algorithm to get the secret function key $f_k$ and the evaluation key $e_k$ which is made public.
\[
\Pr \left[ A^wPRF.F(fk, x^*, Aux, y^*) = 1 : (fk, ek) \leftarrow wPRF.Gen(\lambda, R), (x^*, Aux) \overset{\$}{\leftarrow} D_{wPRF.F(fk, \cdot)}(ek), y^* \overset{\$}{\leftarrow} \mathcal{Y} \right] \geq \frac{1}{2} + \frac{1}{p(\lambda)} \quad (2)
\]
for every PPT adversary \( A \) and infinitely many \( \lambda \), then there exists a PPT extractor \( E \) and a polynomial \( q(\cdot) \) such that
\[
\Pr \left[ w^* \overset{\$}{\leftarrow} E(fk, x^*, Aux, y^*, \{x_i\}, r) : R(x^*, w^*) = 1, \{x_i\} \right] \geq \frac{1}{\eta(\lambda)} \quad (3)
\]
for infinitely many \( \lambda \).

2.5 Multi-Relation Witness PRF

The notion of multi-relation witness PRFs was introduced by Zhandry [Zha16] to work with multiple relations but with the same secret function key.

**Definition 9.** (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 \( mwPRF=(Gen, F, Eval) \) and works as follows:

- \( fk \leftarrow mwPRF.Gen(\lambda, s) \): It is a randomized algorithm run by a user which takes as input a security parameter \( \lambda \) and a bound \( s \) on the size of supported relations and produces a secret function key \( fk \).
- \( y \leftarrow mwPRF.F(fk, x) \): 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 \( \mathcal{Y} \).
- \( ek_R \leftarrow mwPRF.KeyGen(fk, R) \): 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 \( R \).
- \( mwPRF.Eval(ek_R, x, w) \in \mathcal{Y} \cup \{\perp\} \): 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
\[
mwPRF.Eval(ek_R, x, w) = \begin{cases} mwPRF.F(fk, x) & \text{if } R(x, w) = 1 \\ \perp & \text{if } R(x, w) = 0 \end{cases}
\]

**Definition 10.** (Selectively secure multi-relation witness PRF). We say that a multi-relation witness PRF scheme \( mwPRF=(Gen, F, Eval) \) 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
\[
|\Pr[\text{Expt}_{A}^{mwPRF}(1^\lambda, 0) = 1] - \Pr[\text{Expt}_{A}^{mwPRF}(1^\lambda, 1) = 1]| \leq \mu(\lambda)
\]
for any \( \lambda \in \mathbb{N} \) and every PPT adversary \( A \) in the experiments \( \text{Expt}_{A}^{mwPRF}(1^\lambda, b) \) defined in Figure 3 where \( b \in \{0, 1\} \) and \( \mu \) is a negligible function of \( \lambda \). The \( mwPRF \) is said to be \( \delta \)-selectively secure for some specific negligible function \( \delta(\cdot) \) if the above indistinguishability gap \( \mu(\lambda) \) is smaller than \( \delta(\lambda)^{O(1)} \).

We can similarly define extractable multi-relation witness PRFs as described in Definition 8 for single relation witness PRFs.

---

\(^2\)We note that 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 $A$ chooses a single challenge query on an instance $x^* \in \mathcal{X}$ to the challenger.

2. The challenger generates $\mathbf{f}_k \leftarrow \text{mwPRF.Gen}(1^\lambda, s)$, and delivers $s$ to the adversary $A$.

3. The challenger computes $y_0 \leftarrow \text{mwPRF.F}(\mathbf{f}_k, x^*)$ and $y_1 \leftarrow \mathcal{Y}$ and sends $y_b$ to $A$ for a randomly chosen $b \in \{0, 1\}$.

4. The adversary $A$ makes polynomially many evaluation key queries $R_i \in \mathcal{R}$ for $i = 1, 2, \cdots, l$, to which the challenger responds with $\mathbf{e}_k(R_i) \leftarrow \text{mwPRF.KeyGen}(\mathbf{f}_k, R_i)$ if $x^*$ has no witness corresponding to the relation $R_i$; otherwise stops the game.

5. Next, $A$ makes polynomially many queries on instance $\{x_i^j\}_{j=1}^{u(i)}$ for $i = 1, 2, \cdots, l$, where $u(i)$ is a polynomial in $\lambda$. The challenger responds with $\text{mwPRF.F}(\mathbf{f}_k, x_i^j)$ 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, $A$ outputs a guess $b'$ for $b$.

Figure 3: $\text{Expt}^{\text{mwPRF}}_A(1^\lambda, b)$: The security game of a selectively-secure multi-relation witness PRF scheme

### 2.6 Witness Encryption

Witness encryption was introduced by Garg et al. [GGSW13] to encrypt a message with an NP statement and decryption is successful with a valid witness to the statement.

**Definition 11.** (Witness encryption). A witness encryption (WE) scheme for an NP language $L$ with the witness relation $R : \chi \times W \rightarrow \{0, 1\}$ consists of two algorithms $\text{WE} = (\text{Enc}, \text{Dec})$ satisfying the following:

- $c \leftarrow \text{WE.Enc}(1^\lambda, x, m)$: An encrypter takes as input a security parameter $\lambda$, an instance $x \in \chi$ and a message $m \in \mathcal{M}$ and outputs a ciphertext $c$.

- $\text{WE.Dec}(c, w) \in \mathcal{M} \cup \{\bot\}$: A witness holder takes a ciphertext $c$ and a witness $w \in W$ as input and outputs a message $m \in \mathcal{M}$ or $\bot$.

**Correctness:** For any $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$, $(x, w) \in \chi \times W$ 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 12.** (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 $R : \chi \times W \rightarrow \{0, 1\}$, if

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

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

### 2.7 Offline Witness Encryption

Witness encryption scheme with an offline phase [AFP16] 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.
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 $(st, c_b, pp_e, pp_d)$.

Figure 4: Expt$^\text{OWE}_A(1^\lambda, b)$: The security game of a selectively-secure offline witness encryption

**Definition 13.** (Offline witness encryption). An offline witness encryption (OWE) scheme for an NP language $L$ with witness relation $R : \chi \times W \rightarrow \{0, 1\}$ is a tuple of algorithms OWE = (Setup, Enc, 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 $\lambda$ 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 $\lambda$, an instance $x \in \chi$, a message $m \in \mathcal{M}$ and encryption parameter $pp_e$. It computes a ciphertext $c$ and broadcasts it over a public channel.

- OWE.Dec$(c, w, pp_d) \in \mathcal{M} \cup \{\bot\}$: 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 $\bot$.

**Correctness:** For any $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$, $(x, w) \in \chi \times W$ such that $x \in L$, $R(x, w) = 1$, we have that
\[
\Pr[\text{OWE.Dec}(c, w, pp_d) = m : (pp_e, pp_d) \leftarrow \text{OWE.Setup}(1^\lambda, R)] = 1.
\]

**Definition 14.** (Selectively secure offline witness encryption). We say that an offline witness encryption scheme OWE = (Setup, Enc, Dec) for an NP language $L$ and a relation $R : \chi \times W \rightarrow \{0, 1\}$, is selectively secure if
\[
\left| \Pr[\text{Expt}_A^\text{OWE}(1^\lambda, 0) = 1] - \Pr[\text{Expt}_A^\text{OWE}(1^\lambda, 1) = 1] \right| \leq \mu(\lambda)
\]
for any $\lambda \in \mathbb{N}$ and every PPT adversary $\mathcal{A}$ in the experiments Expt$^\text{OWE}_A(1^\lambda, b)$ defined in Figure 4, where $b \in \{0, 1\}$ and $\mu$ is a negligible function of $\lambda$. The OWE is said to be $\delta$-selectively secure for some specific negligible function $\delta(\cdot)$ if the above indistinguishability gap $\mu(\lambda)$ is less than $\delta(\lambda)^{\Omega(1)}$.

### 2.8 Offline Functional Witness Encryption

The notion of functional witness encryption scheme was given by Boyel et al. [BGI14] who established the equivalence between extractable obfuscation and functional witness encryption with extractable security. Abusalah et al. [AFP16] introduced functional witness encryption with a setup algorithm and named it as offline functional witness encryption.

**Definition 15.** (Offline functional witness encryption). An offline functional witness encryption (OFWE) scheme for an NP language $L$ with witness relation $R : \chi \times W \rightarrow \{0, 1\}$ and a class of functions $\{f_\lambda\}_{\lambda \in \mathbb{N}}$ is a tuple of algorithms OFWE = (Setup, Enc, Dec) with the following requirement:
1. The adversary \( A \) chooses \( x \in X \), \((f_0,m_0),(f_1,m_1) \in f_3 \times M \) 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 \((x,m_0,m_1,\alpha) \) to the challenger where \( \alpha \) is a state containing some auxiliary information.

2. The challenger generates \((pp_e,pp_d) \leftarrow OFWE.Setup(1^\lambda,R)\) and sends this to \( A \).

3. The challenger selects \( b \in \{0,1\} \) and sends \((c_0,pp_e,pp_d) \leftarrow OFWE.Enc(1^\lambda,x,(f_0,m_0),pp_e) \) to \( A \).

4. The adversary \( A \) guesses a bit \( b' \) for \( b \) by observing \((\alpha,c_0,pp_e,pp_d)\).

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

\[
\Pr \left[ OFWE.Dec(c,w,pp_d) = f(m,w) : (pp_e,pp_d) \leftarrow OFWE.Setup(1^\lambda,R), c \leftarrow OFWE.Enc(1^\lambda,x,(f,m),pp_e) \right] = 1.
\]

**Definition 16.** (Selectively secure offline functional witness encryption). We say that an offline functional witness encryption OFWE = (Setup, Enc, Dec) for an NP language \( L \) along with a deterministic function \( \Pi \) and an input \( x \) determines a function \( \Pi(x) \) and reveals no information beyond \( \Pi(x) \). Recently, Lin et al. [LPST16b] 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 17.** (Randomized encoding schemes in CRS model). A randomized encoding scheme RE = (Setup, Enc, Eval) in CRS model for a class of Turing machines \( \{M_\lambda\} \) where Setup and Enc are randomized algorithms and Eval is a deterministic algorithm, performs as follows:

**Figure 5:** \( \text{Expt}_{OFWE}^A(1^\lambda,b) \): The security game of a selectively-secure offline functional witness encryption

- \((pp_e,pp_d) \leftarrow OFWE.Setup(1^\lambda,R)\): A trusted third party runs this algorithm taking input a security parameter \( \lambda \) and publishes a public parameter \( pp_e \) for encryption and a public parameter \( pp_d \) for decryption.

- \( c \leftarrow OFWE.Enc(1^\lambda,x,(f,m),pp_e) \): The encryption algorithm takes as input a security parameter \( \lambda \), an instance \( x \in \chi \), a function \( f \in f_\lambda \), a message \( m \in M \) and encryption parameter \( pp_e \). It outputs a ciphertext \( c \) over a public channel. The domain of the function class is \( M \times W \).

- \( OFWE.Dec(c,w,pp_d) \in M \cup \{\bot\} \): A witness holder on receiving a ciphertext \( c \) runs this algorithm using a witness \( w \) and decryption parameter \( pp_d \) and recovers either \((f(m,w),w) \) or \( \bot \).

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

\[
\Pr \left[ OFWE.Dec(c,w,pp_d) = f(m,w) : (pp_e,pp_d) \leftarrow OFWE.Setup(1^\lambda,R), c \leftarrow OFWE.Enc(1^\lambda,x,(f,m),pp_e) \right] = 1.
\]

**Definition 16.** (Selectively secure offline functional witness encryption). We say that an offline functional witness encryption OFWE = (Setup, Enc, Dec) for an NP language \( L \) with relation \( R : \chi \times W \rightarrow \{0,1\} \) and a function class \( \{\Pi_\lambda\}_{\lambda \in \mathbb{N}} \), is selectively secure if

\[
\left| \Pr[\text{Expt}_A^{OFWE}(1^\lambda,0) = 1] - \Pr[\text{Expt}_A^{OFWE}(1^\lambda,1) = 1] \right| \leq \mu(\lambda)
\]

for any \( \lambda \in \mathbb{N} \) and every PPT adversary \( A \) in the experiments \( \text{Expt}_A^{OFWE}(1^\lambda,b) \) defined in Figure 5 where \( b \in \{0,1\} \) and \( \mu \) is a negligible function of \( \lambda \). The OFWE is said to be \( \delta \)-selectively secure for some specific negligible function \( \delta(\cdot) \) if the above indistinguishability gap \( \mu(\lambda) \) is less than \( \delta(\lambda)^{\Omega(1)} \).

### 2.9 Randomized Encoding Scheme in CRS Model

Randomized encoding was introduced by Ishai and Kushilevitz [IK00] to encode a complex deterministic function \( \Pi \) along with an input \( x \) through an encoding algorithm whose output distribution \( \Pi(x) \) can efficiently compute \( \Pi(x) \) and reveals no information beyond \( \Pi(x) \). Recently, Lin et al. [LPST16b] 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 17.** (Randomized encoding schemes in CRS model). A randomized encoding scheme \( \text{RE} = (\text{Setup, Enc, Eval}) \) in CRS model for a class of Turing machines \( \{M_\lambda\} \) where Setup and Enc are randomized algorithms and Eval is a deterministic algorithm, performs as follows:
• \((crs, pk) \leftarrow \text{RE.Setup}(1^\lambda, 1^m, 1^n, 1^T, 1^l)\): A trusted third party takes as input a security parameter \(\lambda\), 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 \(pk\).

• \(\hat{\Pi}_x \leftarrow \text{RE.Enc}(pk, \Pi, x)\): 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 \(\hat{\Pi}_x\).

• \(y \leftarrow \text{RE.Eval}(\hat{\Pi}_x, crs)\): An evaluator makes use of an encoding \(\hat{\Pi}_x\) 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

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

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

**Definition 18.** \((\lambda_0, S(\cdot))-simulation security of randomized encoding in CRS model). We say that a randomized encoding scheme \(\text{RE}\) 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 \(\text{Sim}\) 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

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

and

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

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

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

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

and

\[
\text{Time}_{\text{RE.Eval}}(\hat{\Pi}_x, crs) = \text{poly}(\lambda, ||\Pi||, |x|, T)
\]

for every security parameter \(\lambda\), Turing machine \(\Pi\), input \(x\), time-bound \(T\) and every encoding \(\hat{\Pi}_x \leftarrow \text{RE.Enc}(pk, \Pi, x)\) where \((crs, pk) \leftarrow \text{RE.Setup}(1^\lambda, \cdot)\). 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 \(\{\mathcal{M}_\lambda\}\) if the efficiency requirement for \(\text{RE.Enc}\) 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 \(\{\mathcal{M}_\lambda\}\) if the efficiency requirement for \(\text{RE.Enc}\) is defined as

\[
\text{Time}_{\text{RE.Enc}}(1^\lambda, \Pi, x, T) = l \cdot \text{poly}(\lambda, ||\Pi||, |x|, \log T)
\]
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 [LPST16b]. The compactness (respectively, sub-linear compactness) of RE in CRS model depends on the compactness (respectively, sub-linear compactness) of the underlying PKFE. In [BNPW16], a weakly sub-linear compact PKFE for $P/\text{poly}$ (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.** [BNPW16] 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.** [LPST16b] 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 $P/\text{poly}$ (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 1.** 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 [LPST16b]. The succinctness of a RE scheme for Turing machines depends on the succinctness of PKFE for $P/\text{poly}$. Using only the sub-exponential hardness of LWE, there exists a succinct PKFE with sub-exponential security for $\text{NC}^1$ [GKP+13a]. Also, there exist transformations [GKP+13a, ABSV14] from symmetric key encryption with decryption circuit in $\text{NC}^1$ together with succinct PKFE for $\text{NC}^1$ to a succinct PKFE for $P/\text{poly}$. We note that if the symmetric key encryption and the succinct PKFE for $\text{NC}^1$ are both sub-exponentially secure then the resulting succinct PKFE for $P/\text{poly}$ is also sub-exponentially secure.

A sub-exponentially secure weakly sub-linear compact PKFE is achieved in [BNPW16] using SXIO and a public-key encryption scheme with sub-exponential security. Indistinguishability obfuscation ($iO$) is a technique to make a class of circuits $\{C_\lambda\}$ unintelligible in the sense that for any circuit $C \in C_\lambda$, $C$ and $iO(1^\lambda, C)$ have the same output on all possible inputs and $iO(1^\lambda, C_0)$ is indistinguishable from $iO(1^\lambda, C_1)$ for any two circuits $C_0, C_1 \in C_\lambda$ such that $C_0(x) = C_1(x)$, where $\lambda$ is the security parameter. We want the size of $iO(1^\lambda, C)$ 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 $C \in C_\lambda$ 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 [BKS15] 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 $\gamma$ 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 [BNPW16].

**Remark 2.** In [LPST16b], an $iO$ is instantiated from a sub-exponentially secure and sub-linearly compact RE scheme in CRS model and a sub-exponentially secure pseudorandom generator (PRG). They followed the technique of GGM construction [GGM86] of building a PRF.
from a PRG using a tree. To get an $iO$, the PRG in the GGM construction is replaced with a sub-exponentially secure sub-linear compact RE in CRS model. Let $(C_{\lambda})_{\lambda \in \mathbb{N}}$ be a circuit class with maximum size $S_{\lambda}$, input size $n$, output size $l$ and the running time bound $T$. The obfuscation procedure for a circuit $C \in C_{\lambda}$ works as follows:

- We generate $(crs, pk) \leftarrow \text{RE.Setup}(1^\lambda, 1^n, 1^T, 1^l)$, for $i \in \{0, 1, \ldots, n\}$, where $crs$ is a common reference string and $pk$ is an encoding key. Let $crs = \{crs_i\}_{i=0}^n, pk = \{pk_i\}_{j=1}^n$.

- We construct an input less Turing machine $\Pi[pk_{i+1}, C, z, \alpha_{z_{i+1}}]$, where hardcoded entities are $\bar{pk}_{i+1}, C, z = z_1 z_2 \ldots z_i \in \{0, 1\}^i$ and a string $\alpha_{z_i} \in \{0, 1\}^{2p(\lambda, i)}$ ($p$ being a polynomial depending on the input $\lambda, i$) for all $i \in \{0, 1, \ldots, n-1\}$. When $i = 0$, $z$ is the null string $\epsilon$ and $\alpha_{z_0}$ is a random string $\alpha \leftarrow \{0, 1\}^{2p(\lambda, 0)}$. The Turing machine $\Pi[pk_1, C, \epsilon, \alpha]$ computes randomized encodings of $\Pi[pk_2, C, 0, \alpha_0]$ and $\Pi[pk_2, C, 1, \alpha_1]$ where $(\alpha_0, \alpha_1) \leftarrow \text{PRG}(\alpha)$ with $|\alpha_0| = |\alpha_1| = 2p(\lambda, 1)$, PRG being a sub-exponentially secure pseudorandom generator. To be more specific, the Turing machine $\Pi[pk_1, C, \epsilon, \alpha]$ first generates $(\alpha_0, \alpha_1) \leftarrow \text{PRG}(\alpha)$ and uses the randomness $\alpha_0$ to compute encoding $\bar{\Pi}[pk_2, C, 0, \alpha_0] \leftarrow \text{RE.Enc}(pk_1, \Pi[pk_2, C, 0, \alpha_0], \epsilon)$ and the randomness $\alpha_1$ to compute the encoding $\bar{\Pi}[pk_2, C, 1, \alpha_1] \leftarrow \text{RE.Enc}(pk_1, \Pi[pk_2, C, 1, \alpha_1], \epsilon)$. More generally, the Turing machine $\Pi[pk_{i+1}, C, z, \alpha_{z_i}]$ computes randomized encodings $\bar{\Pi}[pk_{i+2}, C, z, \alpha_{z_i+1}] \leftarrow \text{RE.Enc}(pk_{i+1}, \Pi[pk_{i+2}, C, z, \alpha_{z_i}], \epsilon)$ and $\bar{\Pi}[pk_{i+2}, C, z, z_{i+1}, \alpha_{z_i+1}] \leftarrow \text{RE.Enc}(pk_{i+1}, \Pi[pk_{i+2}, C, z, \alpha_{z_i+1}], \epsilon)$, where $(\alpha_{z_i}, \alpha_{z_i+1}) \leftarrow \text{PRG}(\alpha_{z_{i+1}})$ for $i \in \{1, 2, \ldots, n-1\}$. When $i = n$, the machine $\Pi[pk_{i+1}, C, z, \alpha_{z_n}]$ outputs $C(z)$. We denote the class of all such Turing machines associated with the class of circuits $(C_{\lambda})$ as $(M_{\lambda})$.

- We compute an encoding $\bar{\Pi}[pk_1, C, \epsilon, \alpha] \leftarrow \text{RE.Enc}(pk_0, \Pi[pk_1, C, \epsilon, \alpha], \epsilon)$. Next, we construct the special circuit $\mathcal{G}([\Pi[pk_1, C, \epsilon, \alpha], crs])$ as described in Figure 6 which takes input an $n$ bit string $z = z_1 z_2 \ldots z_n$. For each $i \in \{0, 1, \ldots, n-1\}$, the circuit recursively computes $\text{RE.Eval}(\bar{\Pi}[pk_{i+1}, C, z_1 z_2 \ldots z_i, \alpha_{z_i}], crs_i)$ which by correctness of RE, is equal to the output of the Turing machine $\Pi[pk_{i+1}, C, z_1 z_2 \ldots z_i, \alpha_{z_i}]$, i.e. two randomized encodings $\bar{\Pi}[pk_{i+2}, C, z_1 z_2 \ldots z_0, \alpha_{z_0}]$ and $\bar{\Pi}[pk_{i+2}, C, z_1 z_2 \ldots z_1, \alpha_{z_1}]$ (as in line 3 of Figure 6). Finally, the circuit returns $\text{RE.Eval}(\bar{\Pi}[pk_{n+1}, C, z, \alpha_{z_n}], crs_n)$ which actually is equal to $C(z)$. The obfuscation of the circuit $C$ is $iO(1^\lambda, C) = \mathcal{G}([\Pi[pk_1, C, \epsilon, \alpha], crs])$.

- To evaluate the circuit $C$ for an input $z$, we compute $\mathcal{G}([\Pi[pk_1, C, \epsilon, \alpha], crs](z)$.

Lin et al. [LPST16b] proved that for any pair of functionally equivalent circuits $C_0, C_1 \in C_{\lambda}$, the joint distribution $\mathcal{G}([\Pi[pk_1, C_0, \epsilon, \alpha], crs])$ is indistinguishable from $\mathcal{G}([\Pi[pk_1, C_1, \epsilon, \alpha], crs])$. In particular, they have shown using the method of induction that for any label $i \in \{0, 1, \ldots, n\}$, $z \in \{0, 1\}^i$ the joint distributions $\mathcal{G}([\Pi[pk_{i+1}, C_0, z, \alpha_{z}], crs_i, pk_i])$ and $\mathcal{G}([\Pi[pk_{i+1}, C_1, z, \alpha_{z}], crs_i, pk_i])$ are indistinguishable. The indistinguishability was achieved by the simulation security of the RE scheme.

**Theorem 8.** [LPST16b] 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 Turning machines.

\[\text{For every } \lambda \in \mathbb{N}, i \leq 2^\lambda, \rho(\lambda, i) = \rho(\lambda, i-1) + (2d\lambda)^{1/4} \text{ and } \rho(\lambda, -1) = \lambda \text{ where } \epsilon \text{ is a constant associated with the sub-exponential security of PRG, } d > 0 \text{ is any constant strictly greater than } \epsilon \text{ and the constant } c \text{ represents the security loss associated with the indistinguishability security of RE (section 4, [LPST16b])}.\]
Hardwired: $\Pi [pk]$, C, $\epsilon$, $\alpha$, crs
Input: an input $z = (z_1 z_2 \ldots z_n)$

1. $\bar{\Pi} \leftarrow \Pi [pk]$, C, $\epsilon$, $\alpha$, $i \leftarrow 0$
2. while $i < n$ do
3. $\bar{\Pi} \leftarrow \Pi [pk]$, C, $z_1 z_2 \ldots z_{i+1}$, $\alpha^{i+1}$) $\leftarrow$ RE.Eval($\bar{\Pi}$, crs)
4. $\bar{\Pi} \leftarrow \Pi [pk]$, C, $z_1 z_2 \ldots z_{i+1}$, $\alpha^{i+1}$
5. end do
6. return RE.Eval($\bar{\Pi}$, crs)

Figure 6: The Special Circuit $G[\Pi [pk]$, C, $\epsilon$, $\alpha$, crs$]

Hardwired: a pPRF key $K$.
Input: an instance $x \in X = \{0, 1\}^k$ and a witness $w \in W = \{0, 1\}^{n-k}$.
Padding: the circuit is padded to size $pad = pad(s, n, \lambda)$, determined in the analysis.

1. if $R(x, w) = 1$ then
2. $y \leftarrow$ pPRF.Eval($K$, x).
3. else $y \leftarrow \perp$
4. end if
5. return $y$

Figure 7: Evaluation Circuit $E = EC[K]$

We stress that RE.Enc($pk$, $\Pi [pk]$, C, $\epsilon$, $\alpha$, $\epsilon$) is actually a ciphertext obtained from the encryption algorithm of underlying PKFE that uses ($\Pi [pk]$, C, $\epsilon$, $\alpha$, $\epsilon$, $0^{\lambda+1}$) as the plaintext. The size of the special circuit $G$ is bounded by $\text{poly}(\lambda, |C|, T)$ and runtime of $G$ on input $z$ is bounded by $\text{poly}(\lambda, |z|, |C|, T)$. We will use the notation $G[\Pi [pk]$, C, $\epsilon$, $\alpha$, 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 (wPRF) that uses a puncturable pseudorandom function pPRF = (Gen, Eval, Punc) with domain $X = \{0, 1\}^k$ and range $Y$ and a randomized encoding scheme RE = (Setup, Enc, Eval) which is a bounded input sub-linearly compact randomized encoding scheme in CRS model. Our scheme wPRF = (Gen, F, Eval) for an NP language $L$ with relation circuit $R : X \times W \rightarrow \{0, 1\}$, $X = \{0, 1\}^k$, $W = \{0, 1\}^{n-k}$ and $|R| \leq s$, is given by the following algorithms.

- $(fk, ek) \leftarrow wPRF.Gen(1^\lambda, R)$: 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 $\lambda$ is a security parameter.
  - Choose a pPRF key $K \leftarrow pPRF.Gen(1^\lambda)$ where $K \in \{0, 1\}^\lambda$.
  - Construct the circuit $E = EC[K] \in \{E_\lambda\}$ as defined in Figure 7. 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 (crs$_i$, $pk_i$) $\leftarrow$ RE.Setup($1^\lambda$, $1^S$, $1^n$, $1^T$, $1^l$) for $i \in \{0, 1, \ldots, n\}$ where crs$_i$ is a common reference string and $pk_i$ is an encoding key. We define $crs = \{crs_i\}_{i=0}^n$ and
\( \overrightarrow{pk} = \{ pk_j \}_{j=1}^n \).

- Compute the randomized encoding \( \overrightarrow{\Pi[pk_1, E, \epsilon, \alpha]} \leftarrow \text{RE.Enc}(pk_0, \overrightarrow{\Pi[pk_1, E, \epsilon, \alpha]}, \epsilon) \) where \( \epsilon \) is a null string, \( \alpha \) is a random binary string and \( \Pi[pk_1, E, \epsilon, \alpha] \) is a Turing machine defined in Remark 2.

- Build the special circuit \( G[\overrightarrow{\Pi[pk_1, E, \epsilon, \alpha]}, \overrightarrow{crs}] \) as described in Figure 6.

- Set \( fk = K \), \( ek = G[\overrightarrow{\Pi[pk_1, E, \epsilon, \alpha]}, \overrightarrow{crs}] \) 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.

\[ y \leftarrow wPRF.F(fk, x) \] This algorithm is run by the user who has a secret function key \( fk \) and outputs a \( wPRF \) value \( y \leftarrow pPRF.Eval(K, x) \in \mathcal{Y} \) for an instance \( x \in \mathcal{X} \) using the secret function key \( fk \) as a \( pPRF \) key \( K \).

\[ wPRF.Eval(ek, x, w) \] An evaluator takes a witness \( w \in \mathcal{W} \) for \( x \in L \) and uses the public evaluation key \( ek = G[\overrightarrow{\Pi[pk_1, E, \epsilon, \alpha]}, \overrightarrow{crs}] \) to get back the \( wPRF \) value as \( G[\overrightarrow{\Pi[pk_1, E, \epsilon, \alpha]}, \overrightarrow{crs}](z) \) where \( z = (x, w) \in \{0, 1\}^n \).

**Correctness.** The output of \( wPRF.F \) for an instance \( x \) is a \( pPRF \) evaluation value \( y \leftarrow pPRF.Eval(K, x) \in \mathcal{Y} \) on \( x \) using the secret key \( K \in \{0, 1\}^{\lambda} \). On the other hand, for \( wPRF.Eval \) an witness-holder computes \( G[\overrightarrow{\Pi[pk_1, E, \epsilon, \alpha]}, \overrightarrow{crs}](z) \) where \( z = (x, w) \). By the correctness of randomized encoding scheme as discussed in Remark 2 we have \( G[\overrightarrow{\Pi[pk_1, E, \epsilon, \alpha]}, \overrightarrow{crs}](z) = E(x, w) \). The circuit \( E \) (Figure 7) on input \( x, w \), first checks whether \( R(x, w) = 1 \) holds. If this condition is satisfied, then it outputs \( y \leftarrow pPRF.Eval(K, x) \in \mathcal{Y} \) using the hardcoded key \( K \). Therefore a valid witness-holder of \( x \in L \) can recompute the \( wPRF \) value \( y \in \mathcal{Y} \) associated with \( x \) using the witness \( w \) and the evaluation key \( ek = G[\overrightarrow{\Pi[pk_1, E, \epsilon, \alpha]}, \overrightarrow{crs}] \). Note that, if \( w \) is not valid witness for \( x \in L \) then the output of \( G[\overrightarrow{\Pi[pk_1, E, \epsilon, \alpha]}, \overrightarrow{crs}](z) = E(x, w) \) is the distinguished symbol \( \bot \). Therefore, our witness PRF follows the correctness property stated in Equation 1 Definition 6.

**Padding Parameter.** The proof of security relies on the indistinguishability of randomized encodings of the machines \( \Pi[pk_1, E, \epsilon, \alpha] \) and \( \Pi[pk_1, E', \epsilon, \alpha] \) (where \( E \) and \( E' \) are defined in Figure 7 and 8). 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 simple \( pPRF \) 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 \( wPRF.F \) and \( wPRF.Eval \). The size of \( \mathcal{X} \) is \( 2^k \) and \( wPRF.F \) includes a PRF evaluation over the domain \( \mathcal{X} \). Therefore, size of \( wPRF.F \) is bounded by \( \text{poly}(\lambda, k) \). We note that, \( wPRF.Eval \) only runs the circuit \( G[\overrightarrow{\Pi[pk_1, E, \epsilon, \alpha]}, \overrightarrow{crs}] \) over an input of size \( n \). The running time of \( G[\overrightarrow{\Pi[pk_1, E, \epsilon, \alpha]}, \overrightarrow{crs}] \) is \( \text{poly}(\lambda, n, |E|, T) = \text{poly}(\lambda, n, k, s, T) \) and the size of \( G[\overrightarrow{\Pi[pk_1, E, \epsilon, \alpha]}, \overrightarrow{crs}] \) is \( \text{poly}(\lambda, |E|, T) = \text{poly}(\lambda, k, s, T) \). In particular, the running time and size of \( wPRF.Eval \) 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. Hence, \( T < T_R + \text{poly}(\lambda, k, s, T_R) \).

**Theorem 9.** Assuming LWE with sub-exponential hardness and the existence of \( \delta \)-sub-exponentially secure one-way functions, if there exists a weakly sub-linear compact public key functional encryption scheme for \( \mathbb{P}/\text{poly} \) with \( \delta \)-sub-exponential security, then there exists a \( \delta \)-secure witness PRF scheme.
Linear compact public key functional encryption scheme for $P$

Proof. The existence of sub-exponentially hard LWE and $\delta$-sub-exponentially secure weakly sub-linear public key functional encryption scheme for $P/poly$ imply a $\delta$-sub-exponential simulation secure (Definition 18) randomized encoding scheme for Turing machines in CRS model (Theorem 7). We get a simulation secure (Definition 18) randomized encoding scheme for Turing machines in CRS model (Theorem 7). We get a simulation secure (Definition 18) randomized encoding scheme for Turing machines in CRS model (Theorem 7). We get a simulation secure (Definition 18) randomized encoding scheme for Turing machines in CRS model (Theorem 7). We get a simulation secure (Definition 18) randomized encoding scheme for Turing machines in CRS model (Theorem 7).

Claim 1. Assume existence of $\delta$-sub-exponentially secure one-way functions. Our construction $1$ of $wPRF = (\text{Gen}, F, \text{Eval})$ is $\delta$-selectively secure witness PRF if the pPRF integrated in our scheme is $\delta$-secure puncturable PRF and the RE scheme employed in our scheme is a bounded input sub-linearly compact randomized encoding scheme in CRS model with $\delta$-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 $A$, the distinguishing advantage between the two experiments $\text{Expt}^{wPRF}_A(1^{\lambda}, 0)$ and $\text{Expt}^{wPRF}_A(1^{\lambda}, 1)$ given in Figure 2 is negligible. We consider the following hybrid games.

<table>
<thead>
<tr>
<th>Hybrid Game</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hybd$_0$</td>
<td>This is the standard security experiment $\text{Expt}^{wPRF}_A(1^{\lambda}, 0)$ described in Figure 9.</td>
</tr>
<tr>
<td>Hybd$_1$</td>
<td>In this hybrid game we change $K \leftarrow \text{pPRF.Gen}(1^{\lambda})$ into a punctured key $K{x^<em>} \leftarrow \text{pPRF.Punc}(K, {x^</em>})$ and $ek = G[\Pi[p_{k_i}^1, E^<em>, \epsilon, \alpha], \overline{\epsilon_3}]$ instead of $G[\Pi[p_{k_i}^1, E, \epsilon, \alpha], \overline{\epsilon_3}]$ where $E^</em> = \text{EC}[K{x^<em>}]$ is the circuit as defined in Figure 8 and $y^</em> \leftarrow \text{pPRF.Eval}(K, x^<em>) \in \mathcal{Y}$. We note that the functionality and running time of both the circuits $E$ and $E^</em>$ are the same. Also, the size of the two machines $\Pi[p_{k_i}^1, E, \epsilon, \alpha]$ and $\Pi[p_{k_i}^1, E^<em>, \epsilon, \alpha]$ is the same due to padding. Therefore, the joint distribution ${\Pi[p_{k_i+1}^1, E, z, \alpha_i^z], \overline{\epsilon_3}, p_{k_i}}$ is indistinguishable from ${\Pi[p_{k_i+1}^1, E^</em>, z, \alpha_i^z], \overline{\epsilon_3}, p_{k_i}}$ for every label $i \in {0, 1, \ldots, n}$ and $z \in {0, 1}^t$ (as discussed in Remark 2). Hence by simulation security of the RE scheme, we have $\text{Hybd}<em>0 \approx</em>{\delta} \text{Hybd}_1$ (notation explained in Table 1), i.e., these two hybrid games are computationally indistinguishable.</td>
</tr>
<tr>
<td>Hybd$_2$</td>
<td>This hybrid game is the same as previous one except that here we take $y^<em>$ as a uniformly random element from $\mathcal{Y}$ instead of setting $y^</em> \leftarrow \text{pPRF.Eval}(K, x^*) \in \mathcal{Y}$. From the pseudorandomness at punctured points (Definition 3) of the pPRF we have, $\mu(\lambda) \geq</td>
</tr>
</tbody>
</table>

Figure 8: Evaluation Circuit $E^* = \text{EC}[K\{x^*\}]$
1. The adversary $A$ submits a challenge statement $x^* \in \mathcal{X} \setminus L$.

2. The challenger generates $(\mathcal{K}, \mathcal{E}) \leftarrow \text{wPRF.Gen}(1^\lambda, R)$ as follows and sends $\mathcal{E}$ to $A$:
   2.1 Choose $K \leftarrow \text{pPRF.Gen}(1^\lambda)$ and set $\mathcal{E} = K$.
   2.2 Construct the circuit $E = EC[K]$ as defined in Figure 9.
   2.3 Generate $(\text{crs}, \mathcal{P}) \leftarrow \text{RE.Setup}(1^\lambda, 1^n, 1^r, 1^\ell)$ for $i \in \{0, 1, \ldots, n\}$ where $S, n, T, l$ are the same as in Construction 1 and set $\mathcal{E}_R = \text{crs}_i = \{\text{crs}_i\}^{n}_{i=0}$ and $\mathcal{P}_i = \{\mathcal{P}_i\}^{n}_{i=0}$.
   2.4 Build the special circuit $\mathcal{G}[\mathcal{P}, E, \epsilon, \alpha, \mathcal{E}]$ as described in Figure 9 where $\mathcal{G}[\mathcal{P}, E, \epsilon, \alpha] \leftarrow \text{RE.Enc}(\mathcal{P}, E, \epsilon, \alpha, \epsilon)$ and $\mathcal{G}[\mathcal{P}, E, \epsilon, \alpha]$ is a Turing machine defined in Remark 2.
   2.5 Set $\mathcal{E} = \mathcal{G}[\mathcal{P}, E, \epsilon, \alpha, \mathcal{E}]$.

3. The challenger computes $y^* \leftarrow \text{wPRF.F}(\mathcal{K}, x^*) \in \mathcal{Y}$ and sends it to $A$.

4. The adversary $A$ can make polynomial number of queries for $\text{wPRF.F}$ on some $x \in \mathcal{X} \setminus \{x^*\}$ to the challenger and receives $\text{wPRF.F}(\mathcal{K}, x)$.

5. The adversary $A$ outputs a bit $b'$.

Figure 9: Hybd$_0$ associated with our wPRF for infinitely many $\lambda$ and a negligible function $\mu$ where $U$ denotes uniform distribution over the domain $\mathcal{Y}$ of PRF.Eval. Since the PRF is $\delta$-secure, we have $\mu(\lambda) \leq \delta(\lambda)^{2(1)}$. Thus it holds that Hybd$_1 \approx_\delta$ Hybd$_2$.  

Hybd$_3$ In this hybrid game, again we consider $\mathcal{E} = \mathcal{G}[\mathcal{P}, E, \epsilon, \alpha, \mathcal{E}]$ corresponding to the circuit $E = EC[K]$ 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 Hybd$_2 \approx_\delta$ Hybd$_3$. Note that Hybd$_3$ is actually the regular experiment Expt$_{\text{wPRF}}^{\lambda}(1^\lambda, 1)$. Hence, by the above sequence of hybrid arguments, Expt$_{\text{wPRF}}^{\lambda}(1^\lambda, 0)$ is indistinguishable from Expt$_{\text{wPRF}}^{\lambda}(1^\lambda, 1)$ and we write Expt$_{\text{wPRF}}^{\lambda}(1^\lambda, 0) \approx_\delta$ Expt$_{\text{wPRF}}^{\lambda}(1^\lambda, 1)$.

This completes the proof of Theorem 9.

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 $R : \mathcal{X} \times \mathcal{W} \rightarrow \{0, 1\}$ based on an extractable witness PRF (wPRF) and a randomized encoding (RE) scheme in CRS model. The main ingredients and notations used in this proposed construction are the following:

(i) A public-key encryption $\text{PKE} = (\text{Gen}, \text{Enc}, \text{Dec})$ with CPA-security.

(ii) An extractable witness PRF $w\text{PRF} = (\text{Gen}, \text{Eval})$ for the NP language $L' = \{(c_1, c_2, \mathcal{P}_1, \mathcal{P}_2) : \exists (x, m, r_1, r_2) \text{ such that } c_1 = \text{PKE.Enc}(\mathcal{P}_1, (x, m); r_i) \text{ for } i = 1, 2\}$ with the relation $R'(\mathcal{X}) \times \mathcal{W}^e \rightarrow \{0, 1\}$. Therefore, $R'(\mathcal{X}) = \{(c_1, \mathcal{P}_1, \mathcal{P}_2), (x, m, r_1, r_2)\} = 1$ if $c_1$ and $c_2$ are both encryptions of the same message $(x, m)$ using public keys $\mathcal{P}_1$, $\mathcal{P}_2$ and randomness $r_1$, $r_2$ respectively; otherwise 0. Here we assume that message, ciphertext of the PKE and the wPRF value can be represented as bit-strings.

(iii) A sub-linearly compact bounded input randomized encoding scheme $\text{RE} = (\text{Setup}, \text{Enc}, \text{Eval})$ in CRS model with $\delta$-sub-exponential simulation security for Turing machines. We describe below our offline witness encryption scheme $\text{OWE} = (\text{Setup}, \text{Enc}, \text{Dec})$ for an NP language $L$ with relation circuit $R : \mathcal{X} \times \mathcal{W} \rightarrow \{0, 1\}$. 

22
• \( (pp_e, pp_d) \leftarrow \text{OWE.Setup}(1^\lambda, R) \): This is run by a trusted authority to generate public parameters for both encryption and decryption where \( R \) is a relation circuit and \( \lambda \) 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 \( (fk, ek) \leftarrow \text{wPRF.Gen}(1^\lambda, R') \) for the relation circuit \( R' \) defined in (ii).
  - Construct the circuit \( C_1 = \text{MOC}[SK_1, fk] \in \{C_\lambda\} \) as defined in Figure 10. 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 \( (crs_i, pk_i) \leftarrow \text{RE.Setup}(1^\lambda, 1^S, 1^n, 1^T, 1^i) \) for \( i \in \{0, 1, \ldots, n\} \) where \( crs_i \) is a common reference string and \( pk_i \) is an encoding key. Set \( \overline{crs} = \{crs\}_{i=0}^n \) and \( \overline{pk} = \{pk_i\}_{j=1}^n \).
  - Compute the randomized encoding \( \overline{\Pi[pk_1, C_1, \epsilon, \alpha]} \leftarrow \text{RE.Enc}(\overline{pk_0}, \overline{\Pi[pk_1, C_1, \epsilon, \alpha]}, \epsilon) \) where \( \epsilon \) is a null string, \( \alpha \) is a random binary string and \( \Pi[pk_1, C_1, \epsilon, \alpha] \) is a Turing machine defined in Remark 2.
  - Construct the special circuit \( G[\overline{\Pi[pk_1, C_1, \epsilon, \alpha]}, \overline{crs}] \) as described in Figure 6.
  - Set \( pp_e = (PK_1, PK_2, ek) \), \( pp_d = G[\overline{\Pi[pk_1, C_1, \epsilon, \alpha]}, \overline{crs}] \).
  - Output \( (pp_e, pp_d) \).

• \( c \leftarrow \text{OWE.Enc}(1^\lambda, x, m, pp_e) \): An encryptor 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:
  - Choose two random strings \( r_1, r_2 \leftarrow \{0, 1\}^{l_{\text{PKE}}(\lambda)} \) where \( l_{\text{PKE}} \) is a polynomial in \( \lambda \).
  - Compute two ciphertexts \( c_i = \text{PKE.Enc}(PK_i, (x, m); r_i) \) for \( i = 1, 2 \).
  - Generate a \( w\text{PRF} \) evaluation of the statement \( (c_1, c_2, PK_1, PK_2) \) with witness \( (x, m, r_1, r_2) \) as \( y \leftarrow w\text{PRF.Eval}(ek, (c_1, c_2, PK_1, PK_2), (x, m, r_1, r_2)) \).
  - 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 \( pp_d \) to decrypt the ciphertext to learn the message by outputting \( G[\overline{\Pi[pk_1, C_1, \epsilon, \alpha]}, \overline{crs}](z) \) where \( z = (c, w) \).

**Correctness.** The ciphertext \( c \) has four components where first two components \( c_1, c_2 \) are the ciphertexts of the same message \( (x, m) \), the third one is a statement \( x \in L \) and the last component \( y \) is a \( w\text{PRF} \) 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 10) as the correctness of \( w\text{PRF} \) scheme \( (\text{Equation 1}, \text{Definition 6}) \) 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 \( \text{RE} \) as described in Remark 2 we have \( G[\overline{\Pi[pk_1, C_1, \epsilon, \alpha]}, \overline{crs}](z) = C_1(z) = m \) where \( z = (c, w) \). Hence Equation 4 of Definition 13 holds for this construction which establishes the correctness of our OWE scheme.

**Efficiency.** The encryption algorithm \( \text{OWE.Enc} \) computes two public-key encryption on a message of size \( (|x| + |m|) \) and one \( w\text{PRF} \) evaluation of an input of the form \( (c_1, c_2, PK_1, PK_2) \) with
Assuming the existence of sub-exponentially secure one-way functions, our construction 2 of OWE = (Setup, Enc, Dec) is $\delta$-selectively secure offline witness encryption if the underlying PKE utilized in our OWE is a $\delta$-selectively secure public-key encryption (Definition 5) and the RE employed in our OWE is a bounded input $\delta$-sub-exponential simulation secure (Definition 18) sub-linear compact randomized encoding scheme (Definition 21) in CRS model for the class of Turing machines $\{M_{\lambda}\}$ associated with the class of circuits $\{C_{\lambda}\}$.

**Theorem 10.** Assuming the existence of sub-exponentially secure one-way functions, our construction 2 of OWE = (Setup, Enc, Dec) is $\delta$-selectively secure offline witness encryption if the underlying PKE utilized in our OWE is a $\delta$-selectively secure public-key encryption (Definition 5) and the RE employed in our OWE is a bounded input $\delta$-sub-exponential simulation secure (Definition 18) sub-linear compact randomized encoding scheme (Definition 21) in CRS model for the class of Turing machines $\{M_{\lambda}\}$ associated with the class of circuits $\{C_{\lambda}\}$.

**Proof.** We show that the distinguishing advantage between two experiments $\text{Exp}_{A}^{\text{OWE}}(0^{\lambda}, 0)$ and $\text{Exp}_{A}^{\text{OWE}}(1^{\lambda}, 1)$ (see Figure 1) for any non-uniform PPT adversary $A$ 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 $m_{1}$.

**Hybd$_{0}$** The first game is the standard selective security experiment $\text{Exp}_{A}^{\text{OWE}}(0^{\lambda}, 0)$ where the adversary $A$ is given the ciphertext corresponding to the message $m_{0}$. We describe it in Figure 11.

**Hybd$_{1}$** In this hybrid game, we pick $y$ uniformly at random from $Y$ instead of setting $y \leftarrow \text{wPRF.Eval}(ek, (c_{1}, c_{2}, PK_{1}, PK_{2}), (x, m_{0}, r_{1}, r_{2}))$. All other contents of the ciphertext $c$ remain the same as in the previous game. Since the random values $r_{1}$ and $r_{2}$ which are the part of witnesses corresponding to the ciphertexts $c_{1}$ and $c_{2}$ respectively are not known to the adversary, by the extractable nature of wPRF (Definition 8) and the CPA-security of the underlying PKE scheme (Definition 5), these two hybrid games are computationally indistinguishable. We prove this in the following claim.

**Claim 2.** Assuming the PKE is a $\delta$-selectively secure public-key encryption and the wPRF
1. The adversary chooses \((x, m_0, m_1, st) \leftarrow A(1^\lambda)\) and sends it to the challenger. Here \(x \not\in L\), \(|m_0| = |m_1|\) 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 \(A\):
   2.1 Generate \((SK_i, PK_i) \leftarrow \text{PKE.Gen}(1^\lambda)\) for \(i = 1, 2\) and \((fk, ek) \leftarrow \text{uPRF.Gen}(1^\lambda, R')\) 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 circuit \(C_1 = \text{MOC}[SK_1, fk]\) (see Figure 10)
   2.4 Generate \((crs, pk_i) \leftarrow \text{RE.Setup}(1^{\lambda i}, 1^{\lambda i}, 1^T, 1^t)\) for \(i \in \{0, 1, \ldots, n\}\) where \(S, n, T, l\) are the same as in Construction 2 and set \(\tilde{c}_{ij} = \{crs\}_{i=0}^n \cup \{pk_i\}_{j=1}^n\)
   2.5 Construct the special circuit \(G[\Pi[pk_1, C_1, c, \alpha, \tilde{c}_B]]\) as described in Figure 6 where \(\Pi[pk_1, C_1, c, \alpha, \tilde{c}_B]\) is a Turing machine defined in Remark 2
   2.6 Set \(pp_d = G[\Pi[pk_1, C_1, c, \alpha, \tilde{c}_B]]\)

3. The challenger produces the ciphertext \(c \leftarrow \text{OWE.Enc}(1^\lambda, x, m_0, pp_e)\) as follows and submits it to \(A\):
   3.1 Choose \(r_1, r_2 \in \{0, 1\}^{\text{poly}(\lambda)}\)
   3.2 Compute \(c_1 \leftarrow \text{PKE.Enc}(PK_1, (x, m_0); r_1)\), \(c_2 \leftarrow \text{PKE.Enc}(PK_2, (x, m_0); r_2)\)
   3.3 Evaluate \(y \leftarrow \text{uPRF.Eval}(ek, (c_1, c_2, PK_1, PK_2), (x, m_0, r_1, r_2))\)
   3.4 Set \(c = (c_1, c_2, x, y)\)

4. The adversary observing \((st, c, pp_e, pp_d)\), outputs a bit \(b' \leftarrow A(st, c, pp_e, pp_d)\).

Figure 11: \text{Hybd}_0(\lambda)\) associated with our OWE is extractable, \text{Hybd}_0 and \text{Hybd}_1 are \(\delta\)-indistinguishable.

**Proof.** Let us prove this by contradiction. Suppose, the OWE-adversary \(A\) can distinguish between \text{Hybd}_0 and \text{Hybd}_1. Hence, there exist a polynomial \(p(\cdot)\) such that for infinitely many \(\lambda,
\[|\Pr[\text{Hybd}_0(\lambda) = 1] - \Pr[\text{Hybd}_1(\lambda) = 1]| \geq \frac{1}{2} + \frac{1}{p(\lambda)}\]

We note that in \text{Hybd}_0, \(y\) is computed as \(w\text{PRF.Eval}(ek, (c_1, c_2, PK_1, PK_2), (x, m_0, r_1, r_2))\) which is in fact equal to \(w\text{PRF.F}(fk, (c_1, c_2, PK_1, PK_2)) \in \mathcal{Y}\) (by the correctness of \(w\text{PRF}\)) and in \text{Hybd}_1, \(y\) is chosen uniformly from \(\mathcal{Y}\). Therefore the distinguishing advantage of \(A\) is the same advantage as in Equation 2 (Definition 8) and hence (from Equation 3) the extractable security of \(w\text{PRF}\) implies that there exists a PPT extractor \(E\) and a polynomial \(q(\cdot)\) such that
\[|\Pr[w' \leftarrow E(ek, (c_1, c_2, PK_1, PK_2), \text{Aux}, y, \{x_i\}, r) : R'((c_1, c_2, PK_1, PK_2), w') = 1]| \geq \frac{1}{q(\lambda)}\]

where \(\{x_i\}\) are the same \(w\text{PRF}\) queries made by \(A\) and \(r\) is the random coin used by \(A\). We can construct a PKE-adversary \(B\) against the CPA security (Definition 5) of PKE scheme for the key \(PK_2\) as described in Figure 12.

If \(c'_0 \leftarrow \text{PKE.Enc}(PK_2, (x, m_0); r_2)\), then \(y = w\text{PRF.Eval}(ek, (c_1, c_2, PK_1, PK_2), (x, m_0, r_1, r_2)) = w\text{PRF.F}(fk, (c_1, c_2, PK_1, PK_2))\) and hence the PKE-adversary \(B\) simulates \text{Hybd}_0.

If \(c'_2 \leftarrow \text{PKE.Enc}(PK_2, (x, m_1); r_1)\), then \(w\text{PRF.Eval}(ek, (c_1, c_2, PK_1, PK_2), (x, m_0, r_1, r_2)) \neq w\text{PRF.F}(fk, (c_1, c_2, PK_1, PK_2))\) with high probability and \(y = w\text{PRF.F}(fk, (c_1, c_2, PK_1, PK_2))\) acts like a random \(y\) from \(\mathcal{Y}\) which implies that \(B\) simulates \text{Hybd}_1.

By hypothesis, the adversary \(A\) can distinguish between the hybrids \text{Hybd}_0 and \text{Hybd}_1, and hence there exists an extractor \(E\) that, on input \(ek, (c_1, c_2, PK_1, PK_2), \text{Aux}, y, \{x_i\}, r\), is able to find a witness \(w' = (x, m_0, r_1, r_2)\), where \(\text{Aux}\) contains \(st_A\), the \(w\text{PRF}\) queries \(\{x_i\}\) and \(r\) indicates the random coin used by \(A\). Therefore, the OWE-adversary \(A\) can
The PKE-challenger runs $\text{PKE.Gen}(\lambda) \rightarrow (\text{SK}_2, \text{PK}_2)$ and makes $\text{Pk}_2$ public.

The PKE-adversary $B$ submits the challenge messages $m'_0, m'_1$ to the PKE-challenger as follows with some auxiliary information $st$ and $|m'_0| = |m'_1|$:  
1. Invoke OWE-adversary $A$ to obtain $(x, m_0, m_1, st, A) \leftarrow A(\lambda)$ where $x \notin L$ and $|m_0| = |m_1|$
2. Generate $(\text{SK}_1, \text{PK}_1) \leftarrow \text{PKE.Gen}(\lambda)$
3. Compute $(\text{fk}, \text{ek}) \leftarrow \text{uPRF.Gen}(\lambda, R')$ for the relation $R'$ defined in Construction 2
4. Choose $r_1 \leftarrow \{0,1\}^{\text{PKE}(\lambda)}$
5. Compute $c_1 = \text{PKE.Enc(}\text{PK}_1, (x, m_0); r_1)$
6. Set $m'_0 = (x, m_0), m'_1 = (x, m_1)$ and $st = (\text{SK}_1, \text{PK}_1, \text{fk}, \text{ek}, c_1, r_1, x, m_0, m_1, st, A)$

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 $B$, where $r_2 \leftarrow \{0,1\}^{\text{PKE}(\lambda)}$.

The PKE-adversary $B$ simulates $A$ to output a guess for $b$ by observing $(st, c'_b)$ as follows:  
1. Set $c_2 = c'_b$
2. Compute $y = \text{uPRF.F}(\text{fk}, (c_1, c_2, \text{PK}_1, \text{PK}_2))$
3. Construct the message output circuit $C_1 = \text{MOC}[\text{SK}_1, \text{fk}]$ (see Figure 10)
4. Generate $(\text{crs}_i, \text{pk}_i) \leftarrow \text{RE.Setup}(\lambda, 1^{|\text{crs}_i|}, 1^{|\text{pk}_i|}, 1^{|\text{pk}_i|})$ for $i \in \{0,1,\ldots,n\}$ where $S, n, T, l$ are the same as in Construction 2 and set $\overline{c_i} = \langle \text{crs}_{i,0}, \text{pk}_{i,0} \rangle$ and $\overline{pk}_i = \{\text{pk}_{i,j}\}_{j=1}^n$
5. Construct the special circuit $\mathcal{G} \overline{[\text{pk}_1, C_1, \epsilon, \alpha]} \leftarrow \text{RE.Enc(}\overline{\text{pk}_1}, \overline{\text{pk}_1}, C_1, \epsilon, \alpha, \epsilon)$ and $\overline{\text{pk}_1} \leftarrow \text{RE.Enc(}\overline{\text{pk}_1}, C_1, \epsilon, \alpha)$ a Turing machine defined in Remark 2
6. Set $c = (c_1, c_2, x, y), pp_e = (\text{PK}_1, \text{PK}_2, \text{ek}), pp_d = \mathcal{G} \overline{[\text{pk}_1, C_1, \epsilon, \alpha]} \leftarrow \text{RE.Enc(}\overline{\text{pk}_1}, C_1, \epsilon, \alpha)$ and send $(st, c, pp_e, pp_d)$ to the OWE-adversary $A$
7. Output a guess $b' \leftarrow A(st, c, pp_e, pp_d)$ for $b$

Figure 12: The PKE-adversary $B$ simulating $\text{Hybd}_1$

learn the bit $b$ chosen by the PKE-challenger while executing the CPA security game and sends it to the PKE-adversary $B$ which helps in breaking the CPA security of the PKE scheme for the key $\text{PK}_2$. Consequently, we have for infinitely many $\lambda$, 

$$\frac{1}{q(\lambda)} \leq |\text{Pr}[w' \leftarrow \mathcal{E}(\text{ek}, (c_1, c_2, \text{PK}_1, \text{PK}_2), \text{Aux}, y, \{x_i\}, r) : R'(c_1, c_2, \text{PK}_1, \text{PK}_2, w') = 1]|$$

$$= |\text{Pr}[\text{Exp}_{\text{B}}^{\text{PKE}}(\lambda, 0) = 1] - \text{Pr}[\text{Exp}_{\text{B}}^{\text{PKE}}(\lambda, 1) = 1]|$$

But the CPA security of the underlying PKE scheme implies that this can happen only with a negligible probability. Hence we reach a contradiction to the fact that the success probability of the extractor must be non-negligible. Therefore, we have $\text{Hybd}_0 \approx_\delta \text{Hybd}_1$. This completes the proof of Claim 2.

**Hybd$_2$** This hybrid game is exactly same as $\text{Hybd}_1$ except that we set $c_2 \leftarrow \text{PKE.Enc}(\text{PK}_2, (x, m_1); r_2)$ instead of $c_2 \leftarrow \text{PKE.Enc}(\text{PK}_2, (x, m_0); r_2)$. These two hybrids are computationally indistinguishable by the CPA security of the underlying PKE scheme for the key $\text{PK}_2$ as shown in Claim 3.

**Claim 3.** Assuming the PKE is a $\delta$-selectively secure public-key encryption, $\text{Hybd}_1$ and $\text{Hybd}_2$ are $\delta$-indistinguishable.

**Proof.** We prove this by contradiction. Let us assume that there exists a polynomial $p(\cdot)$ such that for infinitely many $\lambda$ 

$$|\text{Pr}[\text{Hybd}_1(\lambda) = 1] - \text{Pr}[\text{Hybd}_2(\lambda) = 1]| \geq \frac{1}{2} + \frac{1}{p(\lambda)}.$$

We construct a PPT adversary $B$ against the CPA security (Definition 5) of the PKE scheme for the key $\text{PK}_2$ as described in Figure 12 with only change in line 4.2 where we
choose a uniformly random $y$ from $\mathcal{Y}$ instead of computing $y = \text{PRF.F}(\text{fk}, (c_1, c_2, \text{PK}_1, \text{PK}_2))$.

Note that, in both $\text{Hyb}_1$ and $\text{Hyb}_2$ the value of $\text{wPRF.Eval(ek, (c_1, c_2, \text{PK}_1, \text{PK}_2), \cdot)}$ is set as a randomly chosen $y$ from $\mathcal{Y}$. Consequently, $\mathcal{B}$ simulates $\text{Hyb}_1$ if $c'_y \leftarrow \text{PKE.Enc(\text{PK}_2, (x, m_0); r_2)}$, and $\text{Hyb}_2$ if $c'_y \leftarrow \text{PKE.Enc(\text{PK}_2, (x, m_1); r_2)}$. Therefore, the distinguishing advantage of the OWE-adversary $\mathcal{A}$ between $\text{Hyb}_1$ and $\text{Hyb}_2$ is the same as the advantage of the PKE-adversary $\mathcal{B}$ in the CPA security game for the key $\text{PK}_2$. More formally, there exists a polynomial $q(\cdot)$ such that for infinitely many $\lambda$, we have

$$
\frac{1}{q(\lambda)} \leq |\text{Pr}[\text{Hyb}_1(\lambda) = 1] - \text{Pr}[\text{Hyb}_2(\lambda) = 1]| = |\text{Pr}[\text{Expt}^{\text{PKE}}_B(1^\lambda, 0) = 1] - \text{Pr}[\text{Expt}^{\text{PKE}}_B(1^\lambda, 1) = 1]|$$

This shows that we arrive at a contradiction to the CPA security of the PKE scheme. Hence it holds that $\text{Hyb}_1 \approx_3 \text{Hyb}_2$. This completes the proof of Claim 3.

**Hyb$_3$** This hybrid game is the same as the previous game except that we take $\text{pp}_d$ as the circuit $\mathcal{G}[\Pi[pk_1, C_2, \epsilon, \alpha], \text{crs}]$ instead of setting $\text{pp}_d \leftarrow \mathcal{G}[\Pi[pk_1, C_1, \epsilon, \alpha], \text{crs}]$. We show that the adversary $\mathcal{A}$’s distinguishing advantage between $\text{Hyb}_2$ and $\text{Hyb}_3$ is negligible in the following claim.

**Claim 4.** Assuming the RE is a $\delta$-sub-exponential simulation secure sub-linear randomized encoding scheme in CRS model for the class of Turing machines $\{M_\lambda\}$ associated with the class of circuits $\{C_i\}$, $\text{Hyb}_2$ and $\text{Hyb}_3$ are $\delta$-indistinguishable.

**Proof.** We need to show that the joint distributions $(\Pi[pk_{i+1}, C_1, z, \alpha'_z, \text{cr}_i, pk_i])$ and $(\Pi[pk_{i+1}, C_2, z, \alpha'_z, \text{cr}_i, pk_i])$ for every label $i \in \{0, 1, \ldots, n\}$ and $z \in \{0, 1\}^l$, are indistinguishable. It will imply that the two hybrids $\text{Hyb}_2$, $\text{Hyb}_3$ 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 2.

We define an RE-adversary $\mathcal{B}$ against the indistinguishability secure RE scheme in Figure 13. We note RE is $\delta$-indistinguishability secure implies that, if the two ensembles \{\Pi_1(x_1), |\Pi_1, |x_1, T_1 : (\Pi_1, x_1, T_1) \overset{\$}{\leftarrow} X_{1,\lambda}\} and \{\Pi_2(x_2), |\Pi_2, |x_2, T_2 : (\Pi_2, x_2, T_2) \overset{\$}{\leftarrow} X_{2,\lambda}\} are $\delta$-indistinguishable then the two distributions \{RE.Enc(pk, \Pi_1, x_1): (\Pi_1, x_1, T_1) \overset{\$}{\leftarrow} X_{1,\lambda}\} and \{RE.Enc(pk, \Pi_2, x_2): (\Pi_2, x_2, T_2) \overset{\$}{\leftarrow} X_{2,\lambda}\} are also $\delta$-indistinguishable, where $\Pi_j \in M_\lambda$ and $T_j$ denotes the runtime of $\Pi_j$ on input $x_j$ for $j = 1, 2$.

Therefore, if $\text{pp}_d = \mathcal{G}[\Pi[pk_1, C_1, \epsilon, \alpha], \text{crs}]$ then $\mathcal{B}$ simulates $\text{Hyb}_2$ and if $\text{pp}_d = \mathcal{G}[\Pi[pk_1, C_2, \epsilon, \alpha], \text{crs}]$ then $\mathcal{B}$ simulates $\text{Hyb}_3$. 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 10), thus $C_1$ and $C_2$ both output $\bot$.

**Case 2.** $(x \neq \bar{x}, c_1 = \bar{c}_1$ and $c_2 = \bar{c}_2)$: Correctness of PKE scheme implies PKE.Dec(SK, $c_j = (\bar{x}, \bar{m}_j)$ in line 3 of $C_j$ (Figure 10) and both the circuits returns $\bot$ 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 PKE.Dec(SK, $c_1) = \text{PKE.Dec(SK}_2, c_2)$. Therefore, the behavior of both circuits
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 \(B\), where \(\bar{x} \notin L\), \(|\bar{m}_0| = |\bar{m}_1|\) and \(st\) contains some auxiliary information.

2. The RE-adversary \(B\) generates public parameters \((pp_e, pp_d)\) as follows and sends it to \(A\):
   2.1 Generate \((SK_i, PK_i) \leftarrow \text{PKE.Gen}(1^\lambda)\) for \(i = 1, 2\) and \((f_k, ek) \leftarrow \text{PRF.Gen}(1^\lambda, R')\) where relationship \(R'\) is the same as in Construction 2
   2.2 Set \(pp_e = (PK_1, PK_2, ek)\)
   2.3 Construct the message output circuits \(C_j = \text{MOC}[SK_j, f_k]\) for \(j = 1, 2\) (see Figure 10)
   2.4 Generate \((\text{crs}, pk_i) \leftarrow \text{RE.Setup}(1^\lambda, 1^\sigma, 1^n, 1^l, 1^t)\) for \(i \in \{0, 1, \ldots, n\}\) where \(S, n, T, l\) are the same as in Construction 2 and set \(\bar{c} = (\text{crs})_{i=0}^n\) and \(pk_0 = (pk_i)_{i=1}^n\)
   2.5 Submit the circuits \(C_j\) to the RE-challenger for \(j = 1, 2\)
   2.6 The RE-challenger pick a random \(c \in \{1, 2\}\) and sends \(\Pi[pk_1, C_j, \epsilon, \alpha] \leftarrow \text{RE.Enc}(pk_0, \Pi[pk_1, C_j, \epsilon, \alpha], \epsilon)\) to \(B\)
       where \(\Pi[pk_1, C_j, \epsilon, \alpha]\) is a Turing machine defined in Remark 3
   2.7 Construct the special circuit \(G[\Pi[pk_1, C_j, \epsilon, \alpha], \bar{c}])\) as described in Figure 6
   2.8 Set \(pp_d = G[\Pi[pk_1, C_1, \epsilon, \alpha], \bar{c}]\)

3. The RE-adversary \(B\) produces a OWE-ciphertext \(c\) as follows and submits it to the OWE-adversary \(A\):
   3.1 Choose \(r_1, r_2 \in \{0, 1\}^{\text{PK.}(\lambda)}\)
   3.2 Compute \(c_1 \leftarrow \text{PKE.Enc}(PK_1, (\bar{x}, \bar{m}_0); r_1)\), \(c_2 \leftarrow \text{PKE.Enc}(PK_2, (\bar{x}, \bar{m}_1); r_2)\)
   3.3 Choose \(y \leftarrow Y\)
   3.4 Set \(c = (c_1, c_2, \bar{x}, y)\) and send it to the OWE-adversary \(A\)

4. Output \(b' \leftarrow A(\bar{x}, c, \bar{c}, pp_e, pp_d)\).

---

Figure 13: The RE-adversary \(B\) simulating \(\text{Hybd}_4\)

\(C_1\) 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 \((c_1, c_2, PK_1, PK_2) \notin L'\) and by the correctness of \(w\text{PRF}\) scheme we have \(y \neq w\text{PRF.F}(f_k, (c_1, c_2, PK_1, PK_2))\). Hence, the circuits \(C_1\) and \(C_2\) return \(\bot\) due to line 2 (Figure 10).

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 4.

**Hybd_4** The only difference of this hybrid from \(\text{Hybd}_2\) is that we compute \(c_1 \leftarrow \text{PKE.Enc}(PK_1, (x, m_1); r_1)\) instead of \(c_1 \leftarrow \text{PKE.Enc}(PK_1, (x, m_0); r_1)\). Therefore, these two hybrids Hybd_3 and Hybd_4 are computationally indistinguishable by the CPA security of the underlying PKE scheme for the key \(PK_1\) as stated in the following claim.

**Claim 5.** Assuming the PKE is a \(\delta\)-selectively secure public-key encryption, Hybd_3 and Hybd_4 are \(\delta\)-indistinguishable.

The proof is analogous to that of Claim 3.

**Hybd_5** In this hybrid game we take \(pp_d\) as the circuit \(G[\Pi[pk_1, C_1, \epsilon, \alpha], \bar{c}]\) instead of \(G[\Pi[pk_1, C_2, \epsilon, \alpha], \bar{c}]\) 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 6.** Assuming the RE is a \(\delta\)-sub-exponential simulation secure sub-linear compact randomized encoding scheme in CRS model for the class of Turing machines \(\{M_\lambda\}\) associated with the class of circuits \(\{C_\lambda\}\), Hybd_4 and Hybd_5 are \(\delta\)-indistinguishable.

The proof is similar to that of Claim 4.
**Hardwired:** a PKE secret key $SK_j$, a wPRF function key $fk$.

**Input:** a ciphertext $c$ and a witness $w \in W$

1. Parse $c = (c_1, c_2, x, y)$
2. if $(wPRF.F(fk, (c_1, c_2, PK_1, PK_2)) = y)$ then
3. $(\hat{x}, (\hat{f}, \hat{m})) \leftarrow \text{PKE.Dec}(SK_j, c_j)$
4. if $(\hat{x} = x) \land (R(\hat{x}, w) = 1))$ then
5. return $f(\hat{m}, w)$
6. end if
7. end if
8. return $\bot$

Figure 14: Modified Message Output Circuit $F_j = \text{MMOC}[SK_j, fk]$, $j = 1, 2$

**Hybd$_6$** Here we again reset $y \leftarrow wPRF.Eval(ek, (c_1, c_2, PK_1, PK_2), (x, m_1, r_1, r_2))$ instead of randomly chosen $y \in Y$. Again, by a similar argument as in the Hybd$_1$, we have that the distinguishing advantage of the adversary $A$ between the hybrid games Hybd$_5$ and Hybd$_6$ is negligible. We state this in the following claim.

**Claim 7.** Assuming the PKE is a $\delta$-selectively secure public-key encryption and the wPRF is extractable, Hybd$_5$ and Hybd$_6$ are $\delta$-indistinguishable.

The proof is analogous to that of Claim 2.

Observe that Hybd$_6$ is the experiment $\text{Expt}_{\text{OWE}}(1^\lambda, 1)$. The indistinguishability between the above hybrid games implies that $\text{Expt}_{\text{OWE}}(1^\lambda, 0) \approx_\delta \text{Expt}_{\text{OWE}}(1^\lambda, 1)$ and the distinguishing advantage for the adversary $A$ is strictly less than $\mu(\lambda)$, $\mu$ is a negligible function of $\lambda$. 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 14), we will have 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 14). 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 OWEF 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.

### 5 Our Multi-Relation witness PRF

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

Input: an instance $x \in \mathcal{X} = \{0,1\}^k$ and a witness $w \in \mathcal{W} = \{0,1\}^{n-k}$.

Padding: the circuit is padded to size $\text{pad} = \text{pad}_{E_R}(s,d,n,\lambda)$, determined in the analysis.

1. if $(R(x,w) = 1)$ then
2. $y \leftarrow \text{pPRF.Eval}(K,x)$
3. else $y \leftarrow \bot$
4. end if
5. return $y$

- $\text{fk} \leftarrow \text{mwPRF.Gen}(1^\lambda,s)$: This is run by a user with input a security parameter $\lambda$ 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 $\text{fk}$ 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 $\text{mwPRF}$ 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 a evaluation key for any given relation $R \in \mathcal{R}$. It works as follows on input a secret function key $\text{fk}$ and a relation $R$:
  - Construct the circuit $E_R \in \{E_\lambda\}$ as described in Figure 15. Let the circuit $E_R$ be of size $S$ with input size $n$, output size $l$ and runtime bound $T$.
  - Generate $(\text{crs}_i,pk_i) \leftarrow \text{RE.Setup}(1^\lambda,1^S,1^n,1^T)$ for $i \in \{0,1,\ldots,n\}$ where $\text{crs}_i$ is a common reference string and $pk_i$ is an encoding key. Set $\overline{\text{crs}} = \{\text{crs}\}_{i=0}^n$ and $\overline{pk} = \{pk\}_{i=0}^n$.
  - Compute the randomized encoding $\overline{\Pi[pk_1,E_R,\epsilon,\alpha]} \leftarrow \text{RE.Enc}(pk_0,\Pi[pk_1,E_R,\epsilon,\alpha],\epsilon)$ where $\epsilon$ is a null string, $\alpha$ is a random binary string and $\Pi[pk_1,E_R,\epsilon,\alpha]$ is a Turing machine defined in Remark 2.
  - Build the special circuit $G[\Pi[pk_1,E_R,\epsilon,\alpha],[\overline{\text{crs}}]]$ as described in Figure 6 and output $\text{ek}_R = G[\Pi[pk_1,E_R,\epsilon,\alpha],[\overline{\text{crs}}]]$.

- $\text{mwPRF.Eval}(\text{ek}_R,x,w)$: An entity having a witness $w \in \mathcal{W}$ corresponding to an instance $x \in \mathcal{X}$, runs this algorithm using an evaluation key $\text{ek}_R = G[\Pi[pk_1,E_R,\epsilon,\alpha],[\overline{\text{crs}}]]$ and outputs $G[\Pi[pk_1,E_R,\epsilon,\alpha],[\overline{\text{crs}}]](z)$ where $z = (x,w)$.

Correctness. We note that our $\text{mwPRF.F}(\text{fk},x)$ is a pPRF evaluation on $x \in \mathcal{X}$ and one can only use $\text{mwPRF.Eval}$ with an evaluation key $\text{ek}_R$ 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.

\footnote{The only difference from the circuit $E$ (Figure 7) is that, $E_R$ is now hardcoded with a relation $R$ that can vary with evaluation key $\text{ek}_R$.}
Efficiency. The efficiency of our multi-relation witness PRF is the same as that of our single relation witness PRF in Construction 1.

Theorem 11. Assuming LWE with sub-exponential hardness and the existence of $\delta$-sub-exponentially secure one-way functions, if there exists a weakly sub-linear compact public key functional encryption scheme for $P/poly$ with $\delta$-sub-exponential security, then there exists a $\delta$-secure multi-relation witness PRF scheme.

We skip the proof of this theorem as it is similar to the proof of Theorem 9.

6 Conclusion

We constructed a witness PRF from a pPRF and a sub-exponentially secure sub-linear compact RE scheme in CRS model. More precisely, a sub-exponentially secure sub-linear compact RE scheme can be obtained assuming sub-exponential hardness of LWE and existence of sub-exponentially secure one-way functions, sub-exponentially secure weakly sub-linear compact PKFE [LPST16]. The existing construction of witness PRF [Zha16] is based on multi-linear maps and the security is proved in (unreliable) generic multi-linear group model. We also showed that our single relation witness PRF can be immediately converted into a multi-relation witness PRF where the security depends upon the same assumptions as in the case of our single relation witness PRF.

We use any extractable witness PRF to build an offline witness encryption (OWE) from a public-key encryption and a sub-exponentially secure sub-linear compact RE scheme in CRS model. Our OWE is inspired by the existing construction of OWE [AFP16]. We converted our OWE into a offline functional witness encryption with the same modification shown in [AFP16]. However, we unable to produce a polynomial time extractor for our witness PRF. It will be interesting to build such extractor so that extractable witness PRFs can be integrated in various cryptographic constructions.

Our witness PRF and OWE are both works with a statement-witness pair whose lengths are assumed to be bounded. As discussed in Remark 2, we have used the circuit $G[\cdot]$ as an obfuscator for bounded input circuits (or Turing machines). Lin et al. [LPST16] showed that compact RE for certain “special purpose” distributions can be used to obfuscate unbounded input Turing machines. Therefore we can use such compact RE for certain “special purpose” distributions to get witness PRF or OWE for unbounded length statement-witness pair. We note that compact RE for general distributions does not exist in the plain model [LPST16]. It would be desirable to construct witness PRFs or OWEs that support unbounded length statement-witness pair.

References


[DS] David Derler and Daniel Slamanig. Practical witness encryption for algebraic languages or how to encrypt under groth-sahai proofs.


32


