How to Avoid Obfuscation Using Witness PRFs

Mark Zhandry
Stanford University, USA
mzhandry@stanford.edu

Abstract

We propose a new cryptographic primitive called *witness pseudorandom functions* (witness PRFs). Witness PRFs are related to witness encryption, but appear strictly stronger: we show that witness PRFs can be used for applications such as multi-party key exchange without trsuted setup, polynomially-many hardcore bits for any one-way function, and several others that were previously only possible using obfuscation. Current candidate obfuscators are far from practical and typically rely on unnatural hardness assumptions about multilinear maps. We give a construction of witness PRFs from multilinear maps that is simpler and much more efficient than current obfuscation candidates, thus bringing several applications of obfuscation closer to practice. Our construction relies on new but very natural hardness assumptions about the underlying maps that appear to be resistant to a recent line of attacks.

Keywords: Witness PRFs, multilinear maps, multiparty key exchange

1 Introduction

The goal of program obfuscation in cryptography is to scramble a program with the intention of hiding embedded secrets. Recently, Garg et al. [GGH+13b] gave the first candidate construction of a program obfuscator, which has sparked a flurry of research showing many exciting uses of obfuscation. Such uses include functional encryption [GGH+13b], short signatures and deniable encryption [SW14], multiparty key exchange and traitor tracing [BZ14], and much more [HSW14, GGHR14, BCP14, ABG+13, PPS13, KNY14].

While these results are exciting, instantiating these schemes with current candidate obfuscators [GGH+13b, BR14, BGK+14, PST14, AGIS14, GLSW14, SZ14, Zim14] has several drawbacks:

- First, all current obfuscators make use of a core obfuscator for shallow circuits. The size and running time of the obfuscated circuit outputted by the core obfuscator grows exponentially in the depth of the input circuit, thus only permitting very low-depth programs. Getting obfuscation for all circuits currently requires an expensive boosting step involving obfuscating the decryption algorithm for a fully homomorphic encryption scheme. While in many applications it is possible to use asymptotically low-depth primitives, the hidden constants are typically not small. As multiplicative constants in the depth translate to constants in the exponent after applying the core obfuscator, using low-depth primitives gives programs that are polynomial in size and running time, but the polynomial is often astronomical.
- The security of most current obfuscatiors is based on tautological complexity assumptions that essentially match the schemes, resulting in arguably unnatural assumptions. The exception is

the recent work of Gentry, Lewko, Sahai, and Waters [GLSW14], who prove the security of their scheme relative to the multilinear subgroup elimination (MSE) assumption, a simple natural hardness assumption on multilinear maps. While [GLSW14] makes significant progress in improving the security of obfuscation, there are some drawbacks. First, the MSE assumption is a so-called "source-group" assumption, and our confidence in these assumptions has been shaken due to a recent line of attackss [CHL+14, GHMS14, BWZ14b, CLT14]. In particular, there is only one possible candidate multilinear map, namely the very recent map of [CLT15], on which to instantiate the scheme, and the security of these new maps has not been sufficiently vetted. Another drawback of [GLSW14] is that their proof requires complexity leveraging, which appears inherent to basing security of obfuscation on a single assumption [GGSW13]. To use complexity leveraging, the security parameter must be set quite large, compounding the efficiency issues above. Thus, on any maps that support the MSE assumption, the most efficient obfuscation implementations cannot be based on the MSE assumption and will likely instead rely on assumptions matching the scheme.

1.1 Motivating Example: Non-interactive Key Exchange Without Setup

We motivate the following discussion using a specific application of obfuscation: multiparty non-interactive key exchange (NIKE). In such a protocol protocol, n users each generate a secret and public value and simultaneously publishes their public values to a public bulletin board. All of the users then read off the values from the bulletin board and are each able to derive the same shared key with no further interaction. Non-interaction is crucial to obtaining a *re-usable* protocol: $N \gg n$ users can each publish their public value, and then at a later point *any* subset of n of them can establish a shared secret key *without any additional interaction*. In contrast, in an interactive scheme, the protocol needs to be carried out once *for every* subset of users that wishes to derive a key.

Boneh and Silverberg [BS02] show how to build multiparty NIKE from (symmetric) multilinear maps as follows. Recall that a symmetric n-linear map consists of a source group $\mathbb{G}$ with generator g of order p, a target group $\mathbb{G}_T$ with generator g_T of order p, and a multilinear "pairing" operation $e : \mathbb{G}^n \to \mathbb{G}_T$ with the property that $e(g^{a_1}, g^{a_2}, \ldots, g^{a_n}) = g_T^{a_1 a_2 \ldots a_n}$. Ideally any operation except the group and pairing operations should be computationally infeasible. Using an n-linear map, the Boneh-Silverberg protocol for $n+1$ users is as follows: user i chooses a random $a_i \in \mathbb{Z}_p$, and publishes $h_i = g^{a_i}$. The shared secret is $K = g_T^{a_1 a_2 \ldots a_{n+1}}$. User i can compute K as $e(h_1, h_2, \ldots, h_{i-1}, h_{i+1}, \ldots, h_{n+1})^{a_i}$ by pairing the other n public values, and then exponentiating by her secret value. However, an eavesdropper that only sees the h_i would have to pair all $n+1$ of the public values to obtain K, but the pairing operation only supports pairing n elements together.

Garg, Gentry, and Halevi [GGH13a] give the first candidate multilinear map construction, thus giving the first multiparty NIKE protocol. However, in their construction, generating g and e require secrets, knowledge of which completely breaks any security of the maps. The protocol is therefore only non-interactive in a *trusted* setup model, where setup must be performed by a central authority, and the authority will also be able to learn the shared key. Moreover, since g is needed for users to compute their public value, the setup must take place *before* the protocol is carried out. The need for a trusted central authority is a serious limitation of the protocol.

Multiparty NIKE without setup. Boneh and Zhandry [BZ14] show how to use obfuscation to remove the setup phase entirely. In their protocol, each party generates a seed s_i of length λ for

a pseudorandom generator G with output size 2λ, and publishes the corresponding output x_i. In addition, a designated master party (say, party 1) chooses a random key fk for a PRF F, and builds the following program P:

- On input $y_1, \ldots, y_n, s, i$, check that $\mathsf{G}(s) = y_i$.
- If the check fails, output $\perp$.
- Otherwise, output $\mathsf{F}(\mathsf{fk}, y_1, \ldots, y_n)$.

The master party then publishes an obfuscation of P along with their public x_i. Each party i can now compute $K = \mathsf{F}(\mathsf{fk}, x_1, \ldots, x_n)$ by feeding $x_1, \ldots, x_n, s_i, i$ into the obfuscation of P. Thus, all parties establish the same shared key K. An eavesdropper meanwhile only gets to see the obfuscation of P and the x_i, and tries to determine K. He can do so in one of two ways:

- Run the obfuscation of P on inputs of his choice, hoping that one of the outputs is K.
- Inspect the obfuscation of P to try to learn K.

The one-wayness of G means the first approach is not viable. Boneh and Zhandry show that when using an indistinguishability obfuscator and "puncturable" PRF, the value of K is still hidden, even if the adversary inspects the obfuscated code for P. The proof works roughly as follows: first, all of the public values x_i are replaced with truly random strings. The security of G shows that this change is undetectable. Then, since G is expanding, with high probability, none of the x_i have pre-images under G. This means there is no input to the program P that causes it to pass the check and output $K = \mathsf{F}(\mathsf{fk}, x_1, \ldots, x_n)$. Then, using indistinguishability obfuscation and the puncturing property of F, it is possible to show the adversary learns no information about K.

Implementing the Boneh Zhandry protocol. There are two ways to instantiate the obfuscator in the protocol above using multilinear maps:

- Directly on a "core obfuscator" for shallow circuits. The multilinearity required for the underlying map will be approximately 2^d for input circuit of depth d. This presents a serious implementation barrier, as parameters in current multilinear maps grow polynomially with the multilinearity. In an asymptotic sense, using circuits of logarithmic depth will result in polynomial-sized programs. In the case of the Boneh-Zhandry protocol, the bottleneck is clearly the PRF. While there exist puncturable PRFs that are computable in log-depth (for example, it is folklore that the Naor-Reignold PRF is puncturable), the constant term is moderate. Thus, if the depth of the PRF is, say $c \log(2n\lambda)$ ($2n\lambda$ being roughly the input size to the PRF), the resulting program requires multilinearity at least $(2n\lambda)^c$, a polynomial. However, for even moderate c, this polynomial becomes very large.
- By boosting the "core obfuscator" to a general obfuscator for all circuits. Depending on the conversion used, this at best requires obfuscating a low-depth PRF [App13] with the core obfuscator any way, and at worst obfuscating the decryption function of a fully homomorphic encryption scheme [GGH+13b]. Therefore, this approach seems unlikely to yield significant improvements.

Thus, we pay a very steep price for eliminating the setup. Using multilinearity as a proxy for efficiency, we see that the multilinearity for an n-user protocol increases from $n-1$ to $(2n\lambda)^c$ for a moderate constant c.

1.2 Abstracting the Needed Functionality

Observe that we do not need to hide the entire program P in the application above: for example, the entire computation up until the evaluation of the PRF can be leaked. Thus, we do not necessarily need the full power of obfuscation. In fact, obfuscation is used in a very particular way:

- First, the input is separated into two parts. The first part, the "instance", consists of the $y_1, \ldots, y_n$. The second part, the "witness" or "token", consists of s, i.
- The program has the following structure: check some relation between the instance and witness and then apply a PRF to the instance (but not the witness) if the check passes.
- The security we desire is that if the instance has no witness, no information about the output of the PRF at that point is revealed.

Thus, obfuscation is acting as an access control to the PRF, only allowing evaluation at a point if the user can supply a valid token.

1.3 Our Results

In this work, define a new primitive called *witness pseudorandom functions* (witness PRFs) that captures the functionality and security properties needed above. Informally, a witness PRF for an NP language L is a PRF F such that anyone with a valid witness that $x \in L$ can compute $\mathsf{F}(x)$ without the secret key, but for all $x \notin L$, $\mathsf{F}(x)$ is computationally hidden without knowledge of the secret key. More precisely, a witness PRF consists of the following three algorithms:

- $\mathsf{Gen}(L, n)$ takes as input (a description of) an NP language L and instance length n (and implicitly a security parameter), and outputs a secret function key fk and public evaluation key ek.
- $\mathsf{F}(\mathsf{fk}, x)$ takes as input the function key fk, an instance $x \in \{0,1\}^n$, and produces an output y.
- $\mathsf{Eval}(\mathsf{ek}, x, w)$ takes the evaluation key ek, and instance x, and a witness w that $x \in L$, and outputs $\mathsf{F}(\mathsf{fk}, x)$ if w is a valid witness, $\perp$ otherwise.

For security, we require that for any $x \in \{0,1\}^n \setminus L$, the value $\mathsf{F}(\mathsf{fk}, x)$ is pseudorandom even given ek. In Section 3, we also consider many variants of this definition. For example, an interactive variant allows the adversary to make polynomially many PRF queries to $\mathsf{F}(\mathsf{fk}, \cdot)$, and still requires that $\mathsf{F}(\mathsf{fk}, x)$ is indistinguishable from random (conditioned, of course, on x not being one of the PRF queries). We also define an extractable variant that allows $x \in L$, but if the adversary can distinguish $\mathsf{F}(\mathsf{fk}, x)$ from random, then the adversary must "know" a witness that $x \in L$.

Witness PRFs are closely related to the concept of smooth projective hash functions (a comparison is given in Section 1.9), and can be seen as a generalization of constrained PRFs [BW13, KPTZ13, BGI14] to arbitrary NP languages[1].

We first show how to replace obfuscation with witness PRFs for certain applications, including a no-setup multiparty key exchange protocol. We then show how to build witness PRFs from multilinear

[1] This is not strictly true, as constrained PRFs generate the secret function key independent of any language and multiple evaluation keys can be generated for multiple languages. Witness PRFs, on the other hand, only permit one evaluation key, and the language for the key must be known when the function key is generated. In Section 7.1, we discuss how to obtain *multi-relation* witness PRFs which get around these issues.

maps. Our witness PRFs are more efficient than current obfuscation candidates, and rely on very natural, though new, assumptions about the underlying maps. We stress that all of our applications can be instantiated using obfuscation, and the applications are therefore not "new." However, instantiating the applications with witness PRFs result in significant efficiency improvements compared to obfuscation. Moreover, in light of recent attacks on multilinear maps [CHL+14, GHMS14, BWZ14b, CLT14], the only assumptions on which we can based security for current obfuscation implementations is the tautological assumption that the schemes are secure. While our witness PRFs also have a similar drawback, our new assumptions are simpler and more natural than the assumptions required for obfuscation, and rely on more robust "target group" assumptions (more details below). Therefore, our work can be seen as (1) improving the minimal assumption under which several applications are possible and (2) providing significant efficiency improvements for those applications.

Below, we list our main results:

- We show how to realize the following primitives from witness PRFs
  - **Multiparty non-interactive key exchange (NIKE) without trusted setup** (Section 4.2). We give a construction closely related to the Boneh-Zhandry [BZ14] protocol, where the obfuscator is replaced with a witness PRF, and prove that security still holds.
  - **Poly-many hardcore bits** (Section 4.3). Bellare, Stepanovs, and Tessaro[BST14] construct a hardcore function of arbitrary output size for any one-way function. They require differing inputs obfuscation [BGI+01, BCP14, ABG+13], which is a form of knowledge assumption for obfuscators. We show how to replace the obfuscator with a witness PRF that satisfies our extractable notion of security.
  - **Reusable Witness Encryption** (Section 4.4). In witness encryption, messages are encrypted to instances x of some NP language L, and any user that knows a witness that $x \in L$ can decrypt the ciphertext. Security says that if $x \notin L$, the ciphertext reveals no information about the plaintext. Garg, Gentry, Sahai, and Waters [GGSW13] define and build the first witness encryption scheme from multilinear maps. Later, Garg et al. [GGH+13b] show that indistinguishability obfuscation implies witness encryption. We show that witness PRFs are actually sufficient, showing that witness PRFs are essentially a generalization of witness encryption.

    We also define a notion of re-usability for witness encryption, and give a construction from witness PRFs. Our re-usable witness encryption scheme has very short ciphiertexts: namely proportional to the security parameter and independent of the size of the relation. Combining with the witness encryption-to-attribute-based encryption conversion of Garg et al. [GGSW13], this allows us to build attribute-based encryption (ABE) for circuits with similarly short ciphertexts (namely independent of the size of the access policy). No other ABE construction with such succinct ciphertexts is known without using obfuscation; it is not known how to construct such an ABE scheme from the (non-reusable) witness encryption scheme of [GGSW13].
  - **Rudich Secret Sharing for mNP** (Section 4.5). Rudich secret sharing is a generalization of secret sharing to the case where the sets of "qualified" users correspond to instances of a monotone NP (mNP) language L. In other words, n users are each given a share of a secret s. Any set $S \subseteq [n]$ of users corresponds to an instance $x \in \{0,1\}^n$, and if the

users in S know a witness that $x \in L$, they can collectively reconstruct the secret using their shares. However, if $x \notin L$, the secret remains hidden. Monotonicity implies that adding users to a qualified set S does not affect the ability of S to compute the secret. Komargodski, Naor, and Yogev [KNY14] give the first construction for all of mNP using witness encryption[2]. We give a related protocol using witness PRFs that is reusable, which results in much shorter shares than in [KNY14].

- **Fully distributed broadcast encryption** (Section 4.6). In broadcast encryption, n users each have a user-specific secret key, and anyone can encrypt a message to an arbitrary subset $S \subseteq [n]$ of users. Each user in S can decrypt using their individual secret, but users outside of S, even if they all collude, learn nothing about the message. The measures of interest for broadcast encryption are the sizes of the ciphertext, user secret keys, and public broadcast key as a function of the number of users n. Boneh and Zhandry [BZ14] observe that multiparty NIKE protocols with small messages give rise to broadcast encryption with constant-size ciphertexts and secret keys, but with large public keys. The resulting scheme has the novel property of being distributed, where users generate their own secret keys. In Boneh and Zhandry's notion of distributed broadcast encryption, the large public keys are inherent because there is a component of the public key corresponding to each user. We put forward a new notion of *fully distributed* broadcast encryption which does not suffer from this issue, and give a construction from our extractable notion of witness PRFs where secret keys, public keys, and ciphertexts are all poly-logarithmic in n. Our scheme even obtains the strong notion of adaptive security[3]. We note that our construction could have been instantiated using (extractable) witness encryption, but witness PRFs give a protocol with better parameters.

- Next, we show how to build witness PRFs from multilinear maps. We first define an intermediate notion of a subset-sum encoding, and construct such encodings from multilinear maps. Our construction is very simple, and we argue security based on new assumptions on multilinear maps. While our assumptions basically match the security of the subset-sum encodings, the assumptions are very simple and natural due to the simplicity of our scheme. Our full construction is given in Section 5.

  In Section 6, we then show how to build witness PRFs from subset-sum encodings. The resulting construction is much more efficient that what is currently possible with obfuscation. In particular, we can build witness PRFs for arbitrary relations directly without the costly boosting step required for obfuscation. The multilinearity required for the underlying multilinear maps is roughly equal to the size of the circuit defining the relation, rather than exponential in the depth, as in current obfuscators. While implementing our construction is still impractical for all except the most basic relations, future research in improving the efficiency of multilinear maps will bring our construction closer to practice.

- Finally, in Section 7.1, we discuss how to obtain a multi-language variant of witness PRFs, where multiple evaluation keys ek_{L_i} corresponding to multiple language L_i can be produced. A witness for x relative to any of the L_i can be used to evaluate the PRF on x, and if $x \notin L_i$ for any i, then the value of the PRF on x is pseudorandom. We do not need such multi-language

[2] Originally, [KNY14] used obfuscation, but in a later update showed that witness encryption was sufficient.

[3] Of course, obtaining adaptive security from an interactive assumption is not that interesting. However, our construction relies only on a *non*-interactive variant. Therefore, obtaining adaptive security is non-trivial.

witness PRFs for any of our applications, but we believe they are an interesting object, and may be useful in other situations.

1.4 Secure Subset-Sum Encodings

As a first step to building witness PRFs, we construct a primitive called a subset-sum encoding. Roughly, such an encoding corresponds to a (multi-)set S of n integers, and consists of a secret encoding function which maps integers t into encodings $\hat{t}$. Additionally, there is a public evaluation function which takes as input a subset $T \subseteq S$, and can compute the encoding $\hat{t}$ of the sum of the elements in T: $t = \sum_{i \in T} i$. For security, we ask that for any t that does not correspond to a subset-sum of elements of S, the encoding $\hat{t}$ is indistinguishable from a random element.

We provide a simple candidate subset-sum encoding from asymmetric cryptographic multilinear maps. We use an asymmetric variant of multilinear maps, though it is straightforward to adapt our protocol to the symmetric setting. Recall that in an asymmetric n-linear map, instead of a single source group $\mathbb{G}$, there are now n source groups $\mathbb{G}_1, \ldots, \mathbb{G}_n$ with generators $g_1, \ldots, g_n$, and the pairing operation only allows for one element from each group. That is, $e : \mathbb{G}_1 \times \cdots \times \mathbb{G}_n \to \mathbb{G}_T$ where

$$e(g_1^{a_1}, g_2^{a_2}, \ldots, g_n^{a_n}) = g_T^{a_1 a_2 \ldots a_n} \ [4].$$

To generate a subset-sum encoding for a collection $S = \{v_1, \ldots, v_n\}$ of n integers, choose a random $\alpha \xleftarrow{R} \mathbb{Z}_p$, and compute $V_i = g_i^{\alpha^{v_i}}$ for $i = 1, \ldots, n$. Publish V_i for each i. α is kept secret.

The encoding of a target integer t is $\hat{t} = g_T^{\alpha^t}$. Given the secret α it is easy to compute $\hat{t}$[5]. Moreover, if $t = \sum_{i \in T} i$ for some subset $T \subseteq S$, then given the public values V_i, it is also easy to compute $\hat{t}$ using the multilinear operation: define $V_{i,1} = V_i$ and $V_{i,0} = g_i$ so that $V_{i,b} = g_i^{\alpha^{b v_i}}$. Then set b_i to be the indicator function for $i \in T$ (so that $t = \sum_{i \in [n]} b_i v_i$) and compute

$$\hat{t} = e(V_{1,b_1}, \ldots, V_{n,b_n}) = e(g_1^{\alpha^{b_1 v_1}}, \ldots, g_n^{\alpha^{b_n v_n}}) = g_T^{\prod_{i \in [n]} \alpha^{b_i v_i}} = g_T^{\left(\alpha^{\sum_{i \in [n]} b_i v_i}\right)} = g_T^{\alpha^t}$$

However, if t cannot be represented as a subset-sum of elements in S, then the multilinear map operations do not allow for computing $\hat{t}$: there is no way to pair or multiply the V_i and g_i together so that the result is $\hat{t}$. We conjecture that in this case, $\hat{t}$ is hard to compute. This gives rise to a new complexity assumption on multilinear maps: we say that the *multilinear subset-sum Diffie-Hellman assumption* holds for a multilinear map if, for any set of integers $S = \{v_1, \ldots, v_n\}$ and any target t that cannot be represented as a subset-sum of elements in S, that $g_T^{\alpha^t}$ is indistinguishable from a random group element, even given the elements $\{g_i^{\alpha^{v_i}}\}_{i \in [n]}$[6]. In Section 7, we show that this assumption holds in a generic model of multilinear maps. We leave for future work the problem of proving security in the more refined generic model of Gentry et al. [GHMS14], which captures the

[4] This is the asymmetric variant of the multilinear map notion proposed by Boneh and Silverberg [BS02]. Current multilinear map candidates actually support a much richer set of operations, but our construction does not require this additional structure.

[5] Current multilinear map candidates do not allow all users to perform exponentiation by arbitrary elements of $\mathbb{Z}_p$, which makes computing V_i and $\hat{t}$ potentially problematic. However, whomever sets up the subset-sum encoding will also set up the multilinear map, and will thus have a trapdoor that *does* allow computing V_i and $\hat{t}$. Therefore, the secret key should also include this trapdoor along with α.

[6] We can also use an even stronger assumption that also allows the adversary to adaptively ask for values $g_T^{\alpha^{t'}}$ for $t' \neq t$. This will result in a stronger security guarantee for the subset-sum encodings and our derived witness PRFs.

recent line of "zero-izing" attacks. However, while we do not prove security in the zero-izing model, we stress that these attacks do not appear to apply to our assumptions.

Our assumption can the seen as an "uber-assumption" family, containing exponentially-many assumptions, one per SubsetSum instance (S, t). For example, setting S to be $\{1, 2, 3\}$ and t to be -1, our assumption states that $g_T^{\alpha^{-1}}$ is indistinguishable from random, given the elements $\{g_1^{\alpha^1}, g_2^{\alpha^2}, g_3^{\alpha^3}\}$. The assumptions in this family are similar in flavor to several existing assumptions on bilinear and multilinear maps, such as the Diffie-Hellman inversion and Diffie-Hellman Exponent assumptions.

Notice that the element that must be distinguished from random, namely $g_T^{\alpha^t}$, is in the target group $\mathbb{G}_T$. Therefore, our assumption is a *target-group* assumption, which appears more plausible on currently multilinear map candidates than *source-group* assumptions involving only elements in the groups $\mathbb{G}_1, \ldots, \mathbb{G}_n$. Indeed, recent attacks have broken many source-group assumptions [CHL+14, GHMS14, BWZ14b, CLT14], and arguably all of the "nice" such assumptions. For all current obfuscators, the assumption that the scheme is secure is a source-group assumption; while the recent line of attacks does not appear to break current obfuscators, the attacks do decrease our confidence in their security. Target-group assumptions, on the other hand, have so-far resisted attack.

Application to witness encryption. Recall that in a witness encryption scheme as defined by Garg, Gentry, Sahai, and Waters [GGSW13], a message m is encrypted to an instance x, which may or may not be in some NP language L. Given a witness w that $x \in L$, it is possible to decrypt the ciphertext and recover m. However, if $x \notin L$, m should be computationally hidden.

Our subset-sum encodings immediately give us witness encryption for the language L of SubsetSum instances. Let (S, t) be a SubsetSum instance. To encrypt a message m to (S, t), generate a subset-sum encoding for set S. Then, using the secret encoding algorithm, compute $\hat{t}$. The ciphertext is the public evaluation function, together with $c = \hat{t} \oplus m$. To decrypt using a witness subset $T \subseteq S$, use the public evaluation procedure on T to obtain $\hat{t}$, and then XOR with c to obtain m. If $(S, T) \notin$ SubsetSum, then the security of our subset-sum encoding implies that $\hat{t}$, and hence m, is hidden from the adversary.

Since SubsetSum is NP-complete, we can use NP reductions to obtain witness encryption for any NP language L. Our scheme may be more efficient than [GGSW13] for languages L that have simpler reductions to SubsetSum than to the ExactCover problem used by [GGSW13]. For example, the language L_{LWE} of learning-with-errors instances admits a very simple algebraic reductions to SubsetSum. Also, while our assumptions are new, they are no more or less plausible than the assumptions used in [GGSW13].

We can also obtain a special case of Rudich secret sharing. Given a SubsetSum instance (S, t), compute the elements $V_i, \hat{t}$ as above, and compute $c = \hat{t} \oplus s$ where s is the secret. Hand out share (V_i, c) to user i. Notice that a set U of users can learn s if they know a subset $T \subseteq U$ such that $\sum_{j \in T} j = t$. If no such subset exists, then our subset-sum Diffie-Hellman assumption implies that s is hidden from the group U of users.

1.5 Witness PRFs for NP

As defined above, witness PRFs are PRFs that can be evaluated on any input x for which the user knows a witness w that $x \in L$. For any $x \notin L$, the value of the PRF remains computationally hidden. Notice that subset-sum encodings *almost* give us witness PRFs for the SubsetSum problem.

Indeed, the setup algorithm for a subset-sum encoding only depends on the subset S of integers, and not the target value t. Thus, a subset-sum encoding for a set S gives us a witness PRF for the language L_S of all integers t that are subset-sums of the integers in S.

To turn a subset-sum encoding into a witness PRF for an arbitrary language, we give a reduction from any NP language L to SubsetSum with the following property: the set S is independent of the instance x itself, but is instead determined entirely by the NP relation defining L (and the instance length). The instance x instead only affects the target t. Therefore, to build a witness PRF for any fixed NP relation R, run our reduction algorithm to obtain a set S_R, and then build a subset-sum encoding for S_R.

A notable feature of our resulting witness PRF is that its efficiency is comparable to the efficiency of existing witness encryption schemes for general relations R. In particular, the level of multilinearity required and the number of group elements in the evaluation key are equal to the size of the set S_R, which is roughly equal to the number of gates in R. In the original witness encryption scheme of Garg, Gentry, Sahai, and Waters [GGSW13], the level of multilinearity required and the number of group elements in the ciphertexts roughly correspond to the ExactCover instance size, which similarly grows linearly with R. Therefore, we get the added functionality of witness PRFs essentially "for free" in terms of efficiency.

1.6 Replacing Obfuscation with Witness PRFs

To demonstrate how to use witness PRFs in place of obfuscation, we return to the problem of multiparty non-interactive key exchange without setup.

We now explain how witness PRFs actually suffice for this application. As in the Boneh-Zhandry protocol, each user chooses a random seed s_i for the PRG G, and publishes the output x_i. Simultaneously, we define an NP language L consisting of all tuples $(x_1, \ldots, x_n)$ where at least one of the x_i has a pre-image under G. Instead of obfuscating a program, the master party can simply produce a witness PRF F for the language L, and publishes the corresponding evaluation key ek. All users then set the shared key to be $\mathsf{F}(\mathsf{fk}, x_1, \ldots, x_n)$, which all the honest parties can compute using ek since they know a witness.

To argue security, as in the Boneh-Zhandry protocol we replace the x_i with random elements, and rely on the security of G to show that this change is undetectable. Then with overwhelming probability none of the x_i have pre-images under G. This means that with overwhelming probability $(x_1, \ldots, x_n)$ is no longer in L. Therefore, the security of the witness PRF shows that the value $K = \mathsf{F}(\mathsf{fk}, x_1, \ldots, x_n)$ is computationally indistinguishable from a random string, as desired.

Notice that the master party does not know the instance $(x_1, \ldots, x_n)$ until *after* all parties have published their values; in particular, he does not know the instance when setting up the witness PRF. This is crucial to obtaining a non-interactive scheme. Witness encryption, on the other hand, requires knowing the instance when generating the ciphertext, and therefore appears insufficient for non-interactive key exchange.

Efficiency comparison. Let $p(\lambda)$ be the circuit size for computing G. It is straightforward to implement a relation for L with circuits of size $8n\lambda + O(\lambda) + p(\lambda)$ (the bottleneck is the muxing operation to select one of the inputs to check). Using fast PRGs, we can take $p(\lambda) = O(\lambda)$. Thus, our witness PRF uses multilinear maps with linearity $8n\lambda + O(\lambda)$. While this is somewhat worse than the multilinearity $n - 1$ required for the direct protocol with trusted setup, it is several orders

of magnitude better than the $(2n\lambda)^c$ linearity required for the obfuscation-based construction. We note that, using knowledge variants of obfuscation as in [ABG+13], it is possible to reduce the multilinearity required for the obfuscation construction to $(\lambda \log n)^{c'}$ for a larger constant c'. Using our knowledge variant of witness PRFs, we can similarly reduce the multilinearity of our protocol to $O(\lambda \log n)$. In either case (using knowledge assumptions or not), our witness PRFs are currently (by far) the most efficient multiparty key exchange protocols that do not require a trusted setup.

The reasons for the efficiency gains are two-fold:

- Our witness PRF construction grows polynomially with circuit size, rather than exponentially in the depth as in current obfuscators. Thus we will get immediate improvements for all except the shallowest circuits.
- For the applications discussed in this work, the original constructions required obfuscating a PRF. This translates to using the underlying multilinear map operations to simulate the evaluation of the PRF, which is quite costly. In contrast, our witness PRFs use the multilinear map elements themselves as the PRF outputs, eliminating the need for a separate PRF computation. Thus only the relation checking needs to be carried out with multilinear operations. For cases such as key exchange where the PRF evaluation is the bottleneck, this results in significant additional efficiency gains.

1.7 More Detail on Results

We now describe in slightly more detail how witness PRFs can be used for applications in addition to multiparty NIKE.

Poly-many hardcore bits for any one-way function. Bellare, Stepanovs, and Tessaro [BST14] show, given any one-way function f, how to construct a function g such that $g(x)$ for a random x is pseudorandom even given $f(x)$, and the function g has arbitrary-length outputs. The idea is that g is an obfuscation of the program that first takes x as input, computes $y = f(x)$, and outputs $K = \mathsf{F}(y)$ where F is a puncturable PRF. Intuitively, in order to learn anything about K from the obfuscation, it should be necessary to know an input x that causes g to output K. Such an input is a pre-image x' of y, which can then be used to break the one-wayness of f. However, such an input always exists (since $y = f(x)$), and this appears to preclude the use of *indistinguishability* obfuscation to argue security[7]. Instead, security relies on a "knowledge" variant of obfuscation, called differing-inputs obfuscation [BCP14, ABG+13].

To use witness PRFs, define the language $L_f = \{y : y = f(x)\}$, where witnesses that $y \in L_f$ are simply the pre-images of y. Then define $g(x)$ as the output of a witness PRF on $y = f(x)$. That is, run $(\mathsf{fk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(L_f, n)$, discard fk, and define $g(x) = \mathsf{Eval}(\mathsf{ek}, f(x), x) = \mathsf{F}(\mathsf{fk}, f(x))$. To prove that g is hardcore, we rely on a knowledge version of witness PRFs, which requires that even for instances in the language, the output of the function is pseudorandom, so long as the adversary does not "know" a witness. The one-wayness of f implies that the adversary who sees only $y = f(x)$ does not "know" a pre-image (a witness) for $y \in L_f$. This means that $g(x) = \mathsf{F}(\mathsf{fk}, y)$ is pseudorandom.

[7] However, in the case where f is injective, they show that indistinguishability obfuscation *does* suffice.

Re-usable witness encryption. We have already seen how witness PRFs give witness encryption. We will actually show that witness PRFs give a re-usable variant, which cuts down significantly on ciphertext size.

The idea is to run $(\mathsf{fk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(L, n)$ once, and publish ek as a public parameter. Then, to encrypt a message m to an instance x, simply produce the ciphertext $\mathsf{F}(\mathsf{fk}, x) \oplus m$. Of course, this scheme can only securely encrypt once to each instance, and requires the secret function key to encrypt. To overcome these limitations, we define a new language $L' = \{(x, y) : x \in L \text{ or } y = \mathsf{G}(s) \text{ for some } s\}$ where $\mathsf{G} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ is an expanding pseudorandom generator. Now ek is generated for the language L'. To encrypt to an instance x, choose a random $s \in \{0,1\}^\lambda$, let $y = \mathsf{G}(s)$. Now notice that s is a witness that $(x, y) \in L'$. Therefore, the encrypter can evaluate F on (x, y) as $K = \mathsf{Eval}(\mathsf{ek}, (x, y), s) = \mathsf{F}(\mathsf{fk}, (x, y))$, and output the ciphertext $(y, K \oplus m)$.

For decryption, notice that a witness w that $x \in L$ is also a witness that $(x, y) \in L'$. Therefore, anyone with a witness w can compute $K = \mathsf{Eval}(\mathsf{ek}, (x, y), w)$, and from this recover m.

For security, suppose $x \notin L$. First, we move to a setting where y is chosen uniformly at random in $\{0,1\}^{2\lambda}$, which is indistinguishable from a valid ciphertext by the security of G. Then, we observe that y is, with overwhelming probability, not in the image space of G. This means that $(x, y) \notin L'$. Witness PRF security then implies that K is pseudorandom, and therefore the plaintext is hidden.

Notice that in our protocol, the length of the ciphertext, not including the instance, is simply $|y| + |m|$, where $|y| = 2\lambda$. This is essentially as good as symmetric key encryption of m, and is notably independent of the instance size and relation size.

We can therefore use our re-usable witness encryption scheme with the witness encryption-to-attribute-based encryption conversion of Garg et al. [GGSW13] to obtain a new ABE scheme. The resulting protocol has very compact ciphertexts: always $2\lambda + |m|$, regardless of attribute or policy size. It is straightforward to obtain similar results from obfuscation; to our knowledge, this is the first ABE scheme with constant-sized ciphertexts that does *not* require obfuscation. No construction with short ciphertexts from weaker primitives such as witness encryption is known.

Distributed broadcast encryption. Our attribute-based encryption scheme can immediately be used to obtain broadcast encryption with short ciphertexts, though short ciphertext protocols have been known for some time [BGW05]. Our multiparty NIKE protocol can be combined with a conversion of Boneh and Zhandry [BZ14] to obtain a protocol with short ciphertexts that is *distributed*, where each user choses their own secret key. Such a protocol was previously only possible with obfuscation. The scheme also has short user secret keys, but public broadcast keys are large.

The large broadcast key is inherent in Boneh and Zhandry's notion of distributed broadcast encryption, since the public key contains a component for each user. We therefore propose a notion of *fully distributed* broadcast encryption, where the components are gradually absorbed into the broadcast key without (necessarily) increasing its length.

We show how to obtain this fully distributed notion using our knowledge variant of witness PRFs and the techniques of Ananth et al. [ABG+13]. Our protocol works roughly as follows. Each user i chooses a random input s_i to a one-way function f, and publishes the corresponding output $x_i = f(s_i)$. The public broadcast key (roughly) consists of the Merkle tree hash h of all the x_i, and the secret key for user i consists of s_i, x_i, and the "opening" of h to x_i at position i. This opening is effectively a "proof" that one of the hashed values is x_i. We show h can be constructed iteratively, one user at a time, so that each user can compute the openings for themselves. Moreover, the properties of Merkle hashes ensure that the size of the opening is at most logarithmic in the number

of elements hashed (that is, the number of users).

To encrypt to a set S of users, the encrypter constructs a language L_S of Merkle hashes in such a way that, for each user $i \in S$, that user's secret key will be a witness that $h \in L_S$, but users outside S will not know a witness. Then, the encrypter encrypts the broadcast message using a witness encryption scheme for language L_S to the instance h. Users in S can decrypt because they know a witness, and those outside S cannot. The size of the relation defining L_S is logarithmic in the number of users, so the size of the ciphertext is (poly)logarithmic. The result scheme is distributed, and has ciphertexts, secret keys, and broadcast keys that are poly-logarithmic in the number of users. We show that by using our *reusable* witness encryption based on witness PRFs, additional log factors can be shaved off. Unfortunately, witnesses that $h \in L_S$ always exist (and more importantly, false "proofs" exist), and so we need to rely on our knowledge variant of witness PRFs (or similarly, a knowledge variant of witness encryption). However, our protocol attains the strong notion of adaptive security.

1.8 Directions for Future Work

Our work raises several intriguing open questions:

- We give several applications of witness PRFs that previously required the full power of obfuscation. For what other applications of obfuscation do witness PRFs suffice?

- Witness PRFs do not appear sufficient for many applications of obfuscation, including some that seem well-suited for witness PRFs on the surface. For example, obfuscation plays a similar role of gatekeeper to a PRF in the traitor tracing scheme of Boneh and Zhandry [BZ14]. However, in there scheme, the underlying relation must actually be kept secret for security to hold. In our notion of witness PRFs, the relation is not a secret, and our construction explicitly requires the relation to be public. A natural goal is to devise a stronger notion of witness PRFs that would suffice for these applications (say, by hiding some information about the relation) but yet has efficiency similar to that of witness PRFs and witness encryption.

- While our assumptions are natural, they are *instance dependent*, meaning that the assumption depends on the challenge instance. This means our scheme relies on an exponential number of assumptions, one per instance. An important goal is therefore to construct witness PRFs from simple instance *in*dependent assumptions. We note that the since witness PRFs imply witness encryption, the arguments of Garg et al. [GGSW13] indicate that such a construction would likely involve complexity leveraging.

  Indistinguishability obfuscation can be used to build witness PRFs, and indistinguishability obfuscation can in turn be based on simple assumptions following the work of Gentry et al. [GLSW14]. However, such an approach defeats the efficiency gains of building witness PRFs directly. A natural starting point to look for a construction would be the witness encryption scheme of Gentry, Lewko, and Waters [GLW14], which is also based on instance independent assumptions.

1.9 Other Related Work

Obfuscation. Barak et al. [BGI+01] begin the formal study of program obfuscation by giving several formalizations of program obfuscation, including virtual black box (VBB) obfuscation,

indistinguishability obfuscation (iO), and differing inputs obfuscation (diO). They show that VBB obfuscation is impossible to achieve for general programs, though VBB obfuscation has since been achieved for very specific functionalities [CRV10]. Garg et al. [GGH+13b] give the first candidate construction of a general purpose indistinguishability obfuscator, which has been followed by several constructions with improved security analyses [BR14, BGK+14, PST14, GLSW14, SZ14] or improved efficiency [AGIS14, SZ14, Zim14]. Boyle, Chung, and Pass [BCP14] and Ananth et al. [ABG+13] independently conjecture that current candidate indistinguishability obfuscators might actually differing inputs obfuscations (also referred to as extractability obfuscators in [BCP14]).

Removing Obfuscation. Very recently, a few works have shown how to remove obfuscation from certain applications. Garg, Gentry, Halevi, and Zhandry [GGHZ14] build the first many-key functional encryption schemes that do not rely on obfuscation, though their construction is obfuscation-inspired. Boneh et al. [BLR+14] build a near-practical order revealing encryption scheme; the only other known construction requires obfuscation and is therefore far from practical.

Smooth Projective Hash Functions. Cramer and Shoup [CS02] define the notion of *smooth projective hash functions* (SPHFs), a concept similar to that of witness PRFs. Concurrently and independently of our work, Chen and Zhang [CZ14] define the notion of *publicly evaluable PRFs* (PEPRFs), which are again similar in concept to witness PRFs. The main differences between SPHFs and PEPRFs and our witness PRFs are the following:

- Existing constructions of SPHFs and PEPRFs are only for certain classes of languages, such as certain group-theoretic languages. Witness PRFs on the other hand, can handle arbitrary NP languages, and such flexibility is required for the applications in this work.
- Security definitions for SPHFs and PEPRFs are somewhat different than for witness PRFs. SPHFs, for example, do not allow multiple function queries, but require statistical security. For PEPRFs, the challenge is chosen randomly, whereas witness PRFs allow adversarial challenges.
- SPHFs and PEPRFs can be built efficiently from standard assumptions. In contrast, our witness PRFs require new assumptions on multilinear maps.

Witness Encryption. Garg, Gentry, Sahai, and Waters [GGSW13] define witness encryption and give the first candidate construction for the NP-Complete ExactCover problem, whose security is based on the *multilinear no-exact-cover problem*, which they define. Goldwasser et al. [GKP+13] define a stronger notion, called extractable witness encryption, which stipulates that anyone who can distinguish the encryption of two messages relative to an instance x must actually be able to produce a witness for x. Our extractable notion for witness PRFs can be seen as a generalization of extractable witness encryption. Subsequently, Garg, Gentry, Halevi, and Wichs [GGHW14] cast doubt on the plausibility of the most general forms of extractable witness encryption (and thus extractable witness PRFs), though their results do not apply to most potential applications of the primitives.

Multiparty Key Exchange. The first key exchange protocol for $n = 2$ users is the celebrated Diffie-Hellman protocol. Joux [Jou04] shows how to use pairings to extend this to $n = 3$ users, and Boneh and Silverberg [BS02] show that multilinear maps give rise to n-user key exchange for

any n. The first multilinear maps were constructed by Garg, Gentry, and Halevi [GGH13a] and by Coron, Lepoint, and Tibouchi [CLT13], giving the first n-user key exchange for $n > 3$. However, the parameters for the map must be published *before* the protocol is executed, meaning the scheme is only non-interactive in the common reference string model. Moreover, whomever creates the parameters would know secrets that would allow for breaking the protocol, which means the reference string must be set up by a trusted authority. Using obfuscation, Boneh and Zhandry [BZ14] give the first n user key exchange protocol for $n > 3$ that does not require a trusted setup, and in fact requires no setup at all.

Hard-core bits. The Goldreich-Levin theorem [GL89] shows how to build a single hard-core bit for any one-way function. While this result can be extended to logarithmically-many bits, and polynomially-many hard-core bits have been constructed for *specific* one-way functions [CGH01], producing polynomially-many hard-core bits for *arbitrary* one-way functions remained an open problem. The obfuscation-based hard-core function of Bellare, Stepanovs, and Tessaro [BST14] is the first and only construction prior to this work.

Broadcast Encryption. There has been an enormous body of work on broadcast encryption, and we only mention a few specific works. Boneh, Gentry, and Waters [BGW05] use bilinear maps to give the first broadcast scheme with short ciphertexts and secret keys, though public broadcast keys grew linearly with the number of users. Some subsequent schemes based on bilinear maps were able to achieve adaptive security [GW09], but the public parameters always grew linearly with the number of recipients. Boneh and Zhandry [BZ14] give a broadcast scheme from indistinguishability obfuscation which achieves similarly short ciphertexts and secret keys. Their broadcast scheme has the novel property of being distributed, where every user chooses their own secret key. However, their public keys are obfuscated programs, and are quite large (namely, linear in the number of users), and security is proved in a weaker *static* model. Ananth et al. [ABG+13] show how to shrink the public key (while keeping secret keys and ciphertexts roughly the same size) at the expense of losing the distributed property. Boneh, Waters, and Zhandry [BWZ14a] give several broadcast schemes whose concrete parameter sizes are much better directly from multilinear maps, and very recently Zhandry [Zha14] gives a variant that is adaptively secure. However, these schemes are also not distributed.

Secret Sharing. The first secret sharing schemes due to Blakely [Bla79] and Shamir [Sha79] are for the *threshold* access structure, where any set of users of size at least some threshold t can recover the secret, and no set of size less than t can learn anything about the secret. In an unpublished work, Yao shows how to perform (computational) secret sharing where the allowable sets are decided by a polynomial-sized monotone circuit. Rudich raises the possibility of performing secret sharing where allowable sets are decided by a non-deterministic circuit. The first such scheme was built by Komargodski, Naor and Yogev [KNY14], and uses witness encryption.

2 Preliminaries

2.1 Subset-Sum

Let $\mathbf{A} \in \mathbb{Z}^{m \times n}$ be an integer matrix, and $\mathbf{t} \in \mathbb{Z}^m$ be an integer vector. The *subset-sum* search problem is to find an $\mathbf{w} \in \{0,1\}^n$ such that $\mathbf{t} = \mathbf{A} \cdot \mathbf{w}$. The decision problem is to decide if such an $\mathbf{w}$ exists.

We define several quantities related to a subset-sum instance. Given a matrix $\mathbf{A} \in \mathbb{Z}^{m \times n}$, let $\mathsf{SubSums}(\mathbf{A})$ be the set of all subset-sums of columns of $\mathbf{A}$. That is, $\mathsf{SubSums}(\mathbf{A}) = \{\mathbf{A} \cdot \mathbf{w} : \mathbf{w} \in \{0,1\}^n\}$. Define $\mathsf{Span}(\mathbf{A})$ as the convex hull of $\mathsf{SubSums}(\mathbf{A})$. Equivalently, $\mathsf{Span}(\mathbf{A}) = \{\mathbf{A} \cdot \mathbf{w} : \mathbf{w} \in [0,1]^n\}$. We define the integer range of $\mathbf{A}$, or $\mathsf{IntRange}(\mathbf{A})$, as $\mathsf{Span}(\mathbf{A}) \bigcap \mathbb{Z}^m$. We note that given an instance $(\mathbf{A}, \mathbf{t})$ of the subset-sum problem, it is efficiently decidable whether $\mathbf{t} \in \mathsf{IntRange}(\mathbf{A})$. Moreover, $\mathbf{t} \notin \mathsf{IntRange}(\mathbf{A})$ implies that $(\mathbf{A}, \mathbf{t})$ is unsatisfiable. The only "interesting" instances of the subset sum problem therefore have $\mathbf{t} \in \mathsf{IntRange}(\mathbf{A})$. From this point forward, we only consider $(\mathbf{A}, \mathbf{t})$ a valid subset sum instance if $\mathbf{t} \in \mathsf{IntRange}(\mathbf{A})$.

2.2 Multilinear Maps

An asymmetric multilinear map [BS02] is defined by an algorithm Setup which takes as input a security parameter λ, a multilinearity n, and a minimum group order p_{min}[8]. It outputs (the description of) $n+1$ groups $\mathbb{G}_1, \ldots, \mathbb{G}_n, \mathbb{G}_T$ of prime order $p \geq \max(2^\lambda, p_{min})$, corresponding generators $g_1, \ldots, g_n, g_T$, and a map $e : \mathbb{G}_1 \times \cdots \times \mathbb{G}_n \to \mathbb{G}_T$ satisfying

$$e(g_1^{a_1}, \ldots, g_n^{a_n}) = g_T^{a_1 \ldots a_n}$$

Cryptographic multilinear maps are multilinear maps where certain computations not expressly allowed by the map are computationally difficult. For example, it should at a minimum be computationally infeasible to compute $a \in \mathbb{Z}_p$ given g_i^a for a random a. An example of the type of computational assumption we make in this work is that the following problem is hard: given $g_i^{ab^i}$ for $i \in [n]$, distinguish g_T^a from a random element of $\mathbb{G}_T$.

Another requirement we make on multilinear maps is that a random element of $\mathbb{G}_T$ is statistically indistinguishable from a uniform random bit string.

Approximate Multilinear Maps. Current candidate multilinear maps [GGH13a, CLT13] are only *approximate* and do not satisfy the ideal model outlined above. In particular, the maps are noisy, resulting in several implications. First, representations of group elements are not unique. Current map candidates provide an extraction procedure that takes a representation of a group element in the the target group $\mathbb{G}_T$ and outputs a canonical representation. This allows multiple users with different representations of the same element to arrive at the same value. The extraction procedure satisfies the requirement that, when applied to a random element of the target group, the result is statistically close to a uniform random bit string.

A more significant limitation is that noise grows with the number of multiplications and pairing operations. If the noise term grows too large, then there will be errors in the sense that the extraction procedure above will fail to output the canonical representation. In our application, the number of

[8] It is easy to adapt multilinear map constructions [GGH13a, CLT13] to allow setting a minimum group order.

multiplications is equal to the multilinearity, which current candidates natively support without needing to adjust the parameter settings[9].

Lastly, and most importantly for our use, current map candidates do not allow regular users to compute g_i^α for any $\alpha \in \mathbb{Z}_p$ of the user's choice. Instead, the user computes a "level-0 encoding" of a random (unknown) $\alpha \in \mathbb{Z}_p$, and then pairs the "level-0 encoding" with g_i, which amounts computing the exponentiation g_i^α. To compute terms like $g_i^{\alpha^k}$ would require repeating this operation k times, resulting in a large blowup in the error. Thus, for large k, computing terms like $g_i^{\alpha^k}$ is infeasible for regular users. However, whomever sets up the map knows secret parameters about the map and *can* compute g_i^α for any $\alpha \in \mathbb{Z}_p$ without blowing up the error. Thus, the user who sets up the map can pick α, compute α^k in $\mathbb{Z}_p$, and then compute $g_i^{\alpha^k}$ using the map secrets. This will be critical for our construction.

3 Witness PRFs

Informally, a witness PRF is a generalization of constrained PRFs [BW13, KPTZ13, BGI14] to arbitrary NP relations. That is, for an NP language L, a user can evaluate the function F at an instance x only if $x \in L$ *and* the user can provide a witness w that $x \in L$. More formally, a witness PRF is the following:

Definition 3.1. A witness PRF is a triple of algorithms $(\mathsf{Gen}, \mathsf{F}, \mathsf{Eval})$ such that:

- Gen is a randomized algorithm that takes as input a security parameter λ and a circuit $R : \mathcal{X} \times \mathcal{W} \to \{0,1\}$[10], and produces a secret function key fk and a public evaluation key ek.
- F is a deterministic algorithm that takes as input the function key fk and an input $x \in \mathcal{X}$, and produces some output $y \in \mathcal{Y}$ for some set $\mathcal{Y}$.
- Eval is a deterministic algorithm that takes as input the evaluation key ek and input $x \in \mathcal{X}$, and a witness $w \in \mathcal{W}$, and produces an output $y \in \mathcal{Y}$ or $\perp$.
- For correctness, we require $\mathsf{Eval}(\mathsf{ek}, x, w) = \begin{cases} \mathsf{F}(\mathsf{fk}, x) & \text{if } R(x,w) = 1 \\ \perp & \text{if } R(x,w) = 0 \end{cases}$ for all $x \in \mathcal{X}, w \in \mathcal{W}$.

We note one significant way in which our notion of witness PRFs is *weaker* than constrained PRFs: our notion only allows a single evaluation key ek for a relation R that must be chosen at setup time. In contrast, constrained PRFs allow arbitrarily-many ek for different circuits, and the circuits can be chosen after setup. This limitation will be inherent to our construction: the function defined by $\mathsf{F}(\mathsf{fk}, \cdot)$ will depend on the relation R. Nonetheless, this definition will be sufficient for our applications. In section 7.1, we define a multi-relation variant, discuss a possible approach to building such enhanced primitives.

[9] In fact, the parameters can be set more aggressively since our application does not need to support re-randomization. Re-randomizing elements adds significant noise in current encodings, and the native parameter settings support this noise growth.

[10] By accepting relations as circuits, our notion of witness PRFs only handles instances of a fixed size. It is also possible to consider witness PRFs for instances of arbitrary size, in which case R would be a Turing machine.

3.1 Security

The simplest and most natural security notion we consider is a direct generalization of the security notion for constrained PRFs, which we call adaptive instance interactive security. Consider the following experiment $\texttt{EXP}^R_{\mathcal{A}}(b, \lambda)$ between an adversary $\mathcal{A}$ and challenger, parameterized by a relation $R : \mathcal{X} \times \mathcal{W} \to \{0, 1\}$, a bit b and security parameter λ.

- Run $(\mathsf{fk}, \mathsf{ek}) \xleftarrow{R} \mathsf{Gen}(\lambda, R)$ and give ek to $\mathcal{A}$.
- $\mathcal{A}$ can adaptively make queries on instances $x_i \in \mathcal{X}$, to which the challenger response with $\mathsf{F}(\mathsf{fk}, x_i)$.
- $\mathcal{A}$ can make a single challenge query on an instance $x^* \in \mathcal{X}$. The challenger computes $y_0 \leftarrow \mathsf{F}(\mathsf{fk}, x^*)$ and $y_1 \xleftarrow{R} \mathcal{Y}$, and responds with y_b.
- After making additional F queries, $\mathcal{A}$ produces a bit b'. The challenger checks that $x^* \notin \{x_i\}$, and that there is no witness $w \in \mathcal{W}$ such that $R(x, w) = 1$ (in other words, $x \notin L$)[11]. If either check fails, the challenger outputs a random bit. Otherwise, it outputs b'.

Define W_b as the event the challenger outputs 1 in experiment b. Let $\texttt{WPRF.Adv}^R_{\mathcal{A}}(\lambda) = |\Pr[W_0] - \Pr[W_1]|$.

Definition 3.2. $\mathsf{WPRF} = (\mathsf{Gen}, \mathsf{F}, \mathsf{Eval})$ is *adaptive instance interactively secure* for a relation R if, for all PPT adversaries $\mathcal{A}$, there is a negligible function negl such that $\texttt{WPRF.Adv}^R_{\mathcal{A}}(\lambda) < \mathsf{negl}(\lambda)$.

We can also define a weaker notion of *static instance* security where $\mathcal{A}$ commits to x^* before seeing ek or making any F queries. Independently, we can also define *non-interactive* security where the adversary is not allowed any F queries.

Fine-grained security notions. While adaptive instance interactive security and its weaker variants will suffice for many applications, it is in some ways stronger than necessary. For example, for several applications, the instance is chosen by the reduction algorithm, not the adversary. Therefore, we aim to give more fine-grained notions of security, similar to the obfuscation-based notions of [BST14]. Such notions might be more plausible than the general purpose notion above, yet suffice for applications.

To that end, we define an *adaptive* R-instance sampler for WPRF as a PPT algorithm $\mathcal{D}$ that does the following. $\mathcal{D}$ is given ek derived as $(\mathsf{fk}, \mathsf{ek}) \xleftarrow{R} \mathsf{Gen}(\lambda, R)$, and is then allowed to make a polynomial number of queries on instances x_i, receiving $\mathsf{F}(\mathsf{fk}, x_i)$ in response. Finally, $\mathcal{D}$ produces an instance $x^* \notin \{x_i\}$ and potentially some auxiliary information Aux. We say that $\mathcal{D}$ is *static* if $\mathcal{D}$ does not depend on ek and does not make any F queries (but may still depend on λ). Finally, $\mathcal{D}$ is *semi-static* if it does not make any F queries, but may depend on ek.

We now define an experiment $\texttt{EXP}^R_{\mathcal{D},\mathcal{A}}(b, \lambda)$ between a challenger and algorithm $\mathcal{A}$, parameterized by relation R, an adaptive, semi-static, or static R-instance sampler $\mathcal{D}$, and bit b:

- $(\mathsf{fk}, \mathsf{ek}) \xleftarrow{R} \mathsf{Gen}(\lambda, R)$, $(x^*, \mathsf{Aux}) \xleftarrow{R} \mathcal{D}^{\mathsf{F}(\mathsf{fk}, \cdot)}(\mathsf{ek})$, $y_0 \leftarrow \mathsf{F}(\mathsf{fk}, x^*), y_1 \xleftarrow{R} \mathcal{Y}$. Give $\mathsf{ek}, x^*, \mathsf{Aux}, y_b$ to $\mathcal{A}$
- $\mathcal{A}$ is allowed to make F queries on instances $x_i \in \mathcal{X}, x_i \neq x^*$, to which the challenger responds with $\mathsf{F}(\mathsf{sk}, x_i)$.

[11] This check in general cannot be implemented in polynomial time, meaning our challenger is not efficient.

- $\mathcal{A}$ eventually outputs a guess b'. If there is a witness $w \in \mathcal{W}$ such that $R(x^*, w) = 1$ (in other words, if $x^* \in L$), then the challenger outputs a random bit. Otherwise, it outputs b'

We define W_b to be the event of outputting 1 in experiment b, and define the advantage of $\mathcal{A}$ to be $\mathtt{WPRF.Adv}^{\mathbf{A}}_{\mathcal{D},\mathcal{A}}(\lambda) = |\Pr[W_0] - \Pr[W_1]|$

We now define our main notion of security for witness PRFs:

Definition 3.3. $\mathsf{WPRF} = (\mathsf{Gen}, \mathsf{F}, \mathsf{Eval})$ is *interactively secure* for relation R and R-instance sampler $\mathcal{D}$ if, for all PPT adversaries $\mathcal{A}$, there is a negligible function negl such $\mathtt{WPRF.Adv}^{R}_{\mathcal{D},\mathcal{A}}(\lambda) < \mathsf{negl}(\lambda)$.

We can also define non-interactive security where we do not allow $\mathcal{A}$ to make any F queries.

We can recast adaptive instance interactive security in this framework:

Fact 3.4. WPRF *is* adaptive witness interactively secure *for a relation R if it is interactively secure for R and all adaptive R-instance samplers $\mathcal{D}$.*

Extractable Witness PRFs. For some applications, we will need an extractable notion of witness PRF, which roughly states that $\mathsf{F}(x)$ is pseudorandom even for instances $x \in L$, unless the adversary "knows" a witness w for x.

Formally, we modify $\mathtt{EXP}^{R}_{\mathcal{D},\mathcal{A}}(b, \lambda)$ to get a new extracting experiment $\mathtt{EXP}^{e,R}_{\mathcal{D},\mathcal{A}}(b, \lambda)$ where we remove the check that $x \notin L$, and define $\mathtt{WPRF.Adv}^{e,R}_{\mathcal{D},\mathcal{A}}(\lambda)$ as the advantage of $(\mathcal{D}, \mathcal{A})$ in this new game. We also define a second experiment for an extractor $\mathcal{E}$:

- $(\mathsf{fk}, \mathsf{ek}) \xleftarrow{R} \mathsf{Gen}(\lambda, R)$, $(x^*, \mathsf{Aux}) \xleftarrow{R} \mathcal{D}^{\mathsf{F}(\mathsf{fk},\cdot)}(\mathsf{ek})$, $y^* \leftarrow \mathsf{F}(\mathsf{fk}, x^*)$, $b' \xleftarrow{R} \mathcal{A}^{\mathsf{F}(\mathsf{fk},\cdot)}(\mathsf{ek}, x^*, \mathsf{Aux}, y^*)$
- Let $\{(x_i, y_i)\}$ be the F queries and responses made by $\mathcal{A}$ and r the random coins used by $\mathcal{A}$. Run $w^* \xleftarrow{R} \mathcal{E}(\mathsf{ek}, x^*, \mathsf{Aux}, y^*, \{x_i\}, r)$. Output $R(x^*, w^*)$.

Define the advantage $\mathtt{EWPRF.Adv}^{R}_{\mathcal{D},\mathcal{E}}(\lambda)$ as the probability the challenger outputs 1.

Definition 3.5. $(\mathsf{Gen}, \mathsf{F}, \mathsf{Eval})$ is *extractable interactively secure* for a relation R and R-instance sampler $\mathcal{D}$ if, for all PPT adversaries $\mathcal{A}$ such that $\mathtt{WPRF.Adv}^{R}_{\mathcal{D},\mathcal{A}}(\lambda) > 1/q_{\mathcal{A}}(\lambda)$ for some polynomial $q_{\mathcal{A}}$, there is an efficient extractor $\mathcal{E}$ and polynomial $q_{\mathcal{E}}$ such that $\mathtt{EWPRF.Adv}^{R}_{\mathcal{D},\mathcal{E}}(\lambda) > 1/q_{\mathcal{E}}(\lambda)$.

Similarly, we define the relaxation to non-interactive security as we did for standard witness PRFs, where $\mathcal{A}$ is not allowed any F queries.

Remark 3.6. Notice that we've restricted the extractor to only making the same queries made by $\mathcal{A}$. This is potentially a stronger requirement than necessary for an extractable witness PRF. However, this restriction will become important in our constructions. For example, consider constructing an extractable witness PRF WPRF for a relation R by first building an extractable witness PRF WPRF' for an NP-Complete relation R', and then performing an NP reduction. To prove extractable security for WPRF, begin with an adversary $\mathcal{A}$ for WPRF. Use $\mathcal{A}$ and the NP reduction to construct an adversary $\mathcal{A}'$ for WPRF'. The existence of $\mathcal{A}'$ implies an extractor $\mathcal{E}'$ for WPRF'. The goal is to use this extractor to build an extractor $\mathcal{E}$ for WPRF. The problem is that, in the reduction, a legal query to WPRF' might not correspond to a legal query for WPRF. Thus if $\mathcal{E}'$ is allowed to make arbitrary queries, then there is no way for $\mathcal{E}$ to simulate them. However, if $\mathcal{E}'$ can only make queries made by $\mathcal{A}'$, this is no longer a problem since the queries made by $\mathcal{A}'$ will correspond to queries made by $\mathcal{A}$, which *are* legal WPRF queries.

We can now give a general extractability definition for witness PRFs:

Definition 3.7. A witness PRF is extractable static witness interactively secure for relation R if it is extractable interactively secure for R and any static R-instance sampler $\mathcal{D}$.

Remark 3.8. It is also possible to define semi-static or adaptive variants of the above. However, these variants are not attainable for many relations R. For example, consider a relation R where it is easy to sample instances in the language along with witnesses, but given only the instance, finding a witness is hard (an example of such a language is the language of outputs of a one-way function, where witnesses are the corresponding inputs). Then consider the following semi-static instance sampler: sample $x^* \in L$ along with witness w, and use w and ek to compute $y^* = \mathsf{Eval}(\mathsf{ek}, x^*, w) = \mathsf{F}(\mathsf{fk}, x^*)$. Output x^* as the instance and y^* as the auxiliary information. Clearly, given y^*, it is easy to distinguish $\mathsf{F}(\mathsf{fk}, x^*)$ from random. However, this is insufficient for extracting a witness w for x^*.

Remark 3.9. We will eventually show that the extractable witness PRFs imply extractable witness encryption. The recent work of Garg, Gentry, Halevi, and Wichs [GGHW14] shows that extractable witness encryption is problematic, casting some doubt on the plausability of of building extractable witness PRFs. The same doubt is cast on our extractable multilienar map assumptions and our intermediate notion of an extractable subset-sum encoding to be defined later. However, we stress that the results of [GGHW14] only apply to specific auxilliary inputs Aux, which turn out to be the obfuscations of certain programs. However, in many applications $\mathcal{D}$ will be determined by the reduction algorithm (that is, not the adversary) and Aux will be very simple or even non-existant. Therefore, the results of [GGHW14] will often not apply. While it may be impossible to build extractable witness PRFs for all $\mathcal{D}$, it seems plausible to build extractable witness PRFs for the specific applications we investigate.

Remark 3.10. We note the counter-intuitive property that extractable witness PRFs to not imply standard witness PRFs. Consider an R-instance sampler that outputs an instance $x \notin L$ with probability $1/2$, and outputs an instance $x \in L$ with an easy-to-compute witness with probability $1/2$. Extractability trvially holds, since it is possible to extract a witness with probability $1/2$. However, non-extracting security may not hold, as the cases where $x \in L$ are eliminated by the challenger. However, for all our applications that us non-extracting witness PRFs, extracting witness PRFs also suffice.

4 Applications

In this section, we show that for several applications of obfuscation, the obfuscator can be replaced with witness PRFs. We break the applications into several categories:

- *Inherent sampler* constructions are those whose security is proven relative to a fixed instance sampler determined entirely by the construction. Of our constructions, these are the most plausible, since they will not be subject to the impossibility results in the literature [GGHW14]. Our constructions are CCA-secure encryption, non-interactive key exchange, and hardcore functions for any one-way function.

- *Parameterized sampler* constructions are those where the security definition of the primitive depends on an instance sampler $\mathcal{D}$, and security holds relative to $\mathcal{D}$ if the underlying witness PRF is secure for some other instance sampler $\mathcal{D}'$ derived from $\mathcal{D}$. Such constructions

include (reusable) witness encryption, and (resuable) secret sharing for monotone NP. These constructions are likely to be secure for some samplers, but may not be secure for all samplers.

- *Restricted sampler class* constructions are those where the instance sampler depends on the adversary $\mathcal{A}$, meaning the witness PRF must be secure for a large class of samplers. However, we show that the sampler is still restricted, meaning security must only hold relative to a restricted set of samplers. Because of the restriction on instance samplers, it is still plausible that the construction is secure even considering impossibility results. Our fully-dynamic broadcast encryption scheme falls into this category.

4.1 CCA-secure Public Key Encryption

We demonstrate that witness PRFs give a simple construction of CCA-secure public key encryption that is similar to the obfuscation-based construction of Sahai and Waters [SW14]. Given the similarities of witness PRFs to smooth projective hash functions (SPHFs) [CS02], and that the original motivation for SPHFs was CCA-secure public key encryption, this result is not surprising. Instead, we present the construciton as a warm-up for the more interesting applications that follow.

Construction 4.1. Let $\mathsf{WPRF} = (\mathsf{WPRF.Gen}, \mathsf{F}, \mathsf{Eval})$ be a witness PRF, and let $\mathsf{G} : \mathcal{S} \to \mathcal{Z}$ be a pseudorandom generator with $|\mathcal{S}|/|\mathcal{Z}| < \mathsf{negl}$. Build the following key encapsulation mechanism $(\mathsf{Enc.Gen}, \mathsf{Enc}, \mathsf{Dec})$:

- $\mathsf{Enc.Gen}(\lambda)$: Let $R(z, s) = 1$ if and only if $\mathsf{G}(s) = z$. In other words, R defines the language L of strings $z \in \mathcal{Z}$ that are images of G, and witnesses are the corresponding pre-images. Run $(\mathsf{fk}, \mathsf{ek}) \xleftarrow{R} \mathsf{WPRF.Gen}(\lambda, R)$. Set fk to be the secret key and ek to be the public key.
- $\mathsf{Enc}(\mathsf{ek})$: sample $s \xleftarrow{R} \mathcal{S}$ and set $z \leftarrow \mathsf{G}(s)$. Output z as the header and $k \leftarrow \mathsf{Eval}(\mathsf{ek}, z, s) \in \mathcal{Y}$ as the message encryption key.
- $\mathsf{Dec}(\mathsf{fk}, z)$: run $k \leftarrow \mathsf{F}(\mathsf{fk}, z)$.

Correctness is immediate. For security, we have the following:

Theorem 4.2. *If* WPRF *is interactively secure, then Construction 4.1 is a CCA secure key encapsulation mechanism. If* WPRF *is static instance non-interactively secure, then Construction 4.1 is CPA secure.*

Rather than prove Theorem 4.2, we instead prove security relative to a fine-grained security notion. Define the following static instance sampler $\mathcal{D}$: sample and output a random $z \in \mathcal{Z}$ and $\mathsf{Aux} = ()$.

Theorem 4.3. *If* G *is a secure pseudorandom generator and* WPRF *is interactively secure for relation R and R-instance sampler $\mathcal{D}$, then Construction 4.1 is a CCA secure key encapsulation mechanism. If* WPRF *is non-interactively secure, then Construction 4.1 is CPA secure.*

Proof. We prove the CCA case, the CPA case being almost identical. Let $\mathcal{B}$ be a CCA adversary with non-negligible advantage ϵ. Define **Game 0** as the standard CCA game, and define **Game 1** as the modification where the challenge header z^* is chosen uniformly at random in $\mathcal{Z}$. The security of G implies that $\mathcal{B}$ still has advantage negligibly-close to ϵ. Let **Game 2** be the game where z^* is

chosen at random, but the game outputs a random bit and aborts if z^* is in the image space of G. Since $\mathcal{Z}$ is much larger than $\mathcal{S}$, the abort condition occurs with negligible probability. Thus $\mathcal{B}$ still has advantage negligibly close to ϵ in **Game 2**. Now we construct an adversary $\mathcal{A}$ for WPRF relative to sampler $\mathcal{D}$. $\mathcal{A}$ simulates $\mathcal{B}$, answering decryption queries using its F oracle. Finally, $\mathcal{B}$ makes a challenge query, and $\mathcal{A}$ responds with its input z^*. When $\mathcal{B}$ outputs a bit b', $\mathcal{A}$ outputs the same bit. $\mathcal{A}$ has advantage equal to that of $\mathcal{B}$ in **Game 2**, which is non-negligible, thus contradicting the security of WPRF. □

We can also relax the requirement on G to be a one-way function if we assume WPRF is extractable. Let $\mathcal{D}'$ be the following static instance sampler: sample $s \xleftarrow{R} \mathcal{S}$ and output $z = f(s)$ and $\mathsf{Aux} = ()$. Then we have the following theorem:

Theorem 4.4. *If G is a secure one-way function and WPRF is extractable interactively secure for relation R and R-instance sampler $\mathcal{D}'$, then Construction 4.1 is a CCA secure key encapsulation mechanism. If WPRF is extractable non-interactively secure, then Construction 4.1 is CPA secure.*

The proof is very similar to the proof of Theorem 4.3, and we omit the details.

4.2 Non-interactive Multiparty Key Exchange

A multiparty key exchange protocol allows a group of g users to simultaneously post a message to a public bulletin board, retaining some user-dependent secret. After reading off the contents of the bulletin board, all the users establish the same shared secret key. Meanwhile, and adversary who sees the entire contents of the bulletin board should not be able to learn the group key. More precisely, a multiparty key exchange protocol consists of:

- $\mathsf{Publish}(\lambda, g)$ takes as input the security parameter and the group order, and outputs a user secret s and public value pv. pv is posted to the bulletin board.
- $\mathsf{KeyGen}(\{\mathsf{pv}_j\}_{j\in[g]}, s_i, i)$ takes as input g public values, plus the corresponding user secret s_i for the ith value. It outputs a group key $k \in \mathcal{Y}$.

For correctness, we require that all users generate the same key:

$$\mathsf{KeyGen}(\{\mathsf{pv}_j\}_{j\in[g]}, s_i, i) = \mathsf{KeyGen}(\{\mathsf{pv}_j\}_{j\in[g]}, s_{i'}, i')$$

for all $(s_j, \mathsf{pv}_j) \xleftarrow{R} \mathsf{Publish}(\lambda, g)$ and $i, i' \in [g]$. For security, we have the following:

Definition 4.5. A non-interactive multiparty key exchange protocol is statically secure if the following distributions are indistinguishable:

$$\{\mathsf{pv}_j\}_{j\in[g]}, k \text{ where } (s_j, \mathsf{pv}_j) \xleftarrow{R} \mathsf{Publish}(\lambda, g) \forall j \in [g], k \leftarrow \mathsf{KeyGen}(\{\mathsf{pv}_j\}_{j\in[g]}, s_1, 1) \text{ and}$$
$$\{\mathsf{pv}_j\}_{j\in[g]}, k \text{ where } (s_j, \mathsf{pv}_j) \xleftarrow{R} \mathsf{Publish}(\lambda, g) \forall j \in [g], k \xleftarrow{R} \mathcal{Y}$$

Notice that our syntax does not allow a trusted setup, as constructions based on multilinear maps [BS02, GGH13a, CLT13] require. Boneh and Zhandry [BZ14] give the first multiparty key exchange protocol without trusted setup, based on obfuscation. We now give a very similar protocol using witness PRFs.

Construction 4.6. Let $\mathsf{G} : \mathcal{S} \to \mathcal{Z}$ be a pseudorandom generator with $|\mathcal{S}|/|\mathcal{Z}| < \mathsf{negl}$. Let $\mathsf{WPRF} = (\mathsf{Gen}, \mathsf{F}, \mathsf{Eval})$ be a witness PRF. Let $R_g : \mathcal{Z}^g \times (\mathcal{S} \times [g]) \to \{0,1\}$ be a relation that outputs 1 on input $((z_1, \ldots, z_g), (s, i))$ if and only if $z_i = \mathsf{G}(s)$. We build the following key exchange protocol:

- $\mathsf{Publish}(\lambda, g)$: compute $(\mathsf{fk}, \mathsf{ek}) \xleftarrow{R} \mathsf{Gen}(\lambda, R_g)$. Also pick a random seed $s \xleftarrow{R} \mathcal{S}$ and compute $z \leftarrow \mathsf{G}(s)$. Keep s as the secret and publish (z, ek).
- $\mathsf{KeyGen}(\{(z_i, \mathsf{ek}_i)\}_{i \in [g]}, s)$. Each user sorts the pairs (z_i, ek_i) by z_i, and determines their index i in the ordering. Let $\mathsf{ek} = \mathsf{ek}_1$, and compute $k = \mathsf{Eval}(\mathsf{ek}, (z_1, \ldots, z_g), (s, i))$

Correctness is immediate. For security, we have the following:

Theorem 4.7. *If* WPRF *is static witness non-interactively secure, the Construction 4.6 is statically secure.*

Rather than prove Theorem 4.7, we instead prove security relative to a fine-grained security notion. Let $\mathcal{D}_g$ be the following R_g-instance sampler: choose random $z_i \xleftarrow{R} \mathcal{Z}$ for $i \in [g]$, and output $(z_1, \ldots, z_g), \mathsf{Aux} = ()$.

We have the following theorem:

Theorem 4.8. *If* WPRF *is non-interactively secure for relation R_g and R-instance sampler $\mathcal{D}_g$, and* G *is a secure PRG, then* (Publish, KeyGen) *is a statically secure NIKE protocol.*

We can also trade a stronger notion of security for the witness PRF in exchange for a weaker security requirement for G. Let $\mathcal{D}'_g$ be the following static R_g-instance sampler: choose random $s_i \xleftarrow{R} \mathcal{S}$, and set $z_i \leftarrow \mathsf{G}(s_i)$ and output $(z_1, \ldots, z_g), \mathsf{Aux} = ()$

Theorem 4.9. *If* WPRF *is extracting non-interactively secure for relation R_g and R-instance sampler $\mathcal{D}'_g$, and* G *is a secure one-way function, then* (Publish, KeyGen) *is a statically secure NIKE protocol.*

Proof. We prove Theorem 4.8, the proof of Theorem 4.9 being similar. Let $\mathcal{B}$ be an adversary for the key exchange protocol with non-negligible advantage. Then $\mathcal{B}$ sees $\{(z_i, \mathsf{ek}_i)\}_{i \in [g]}$ where $z_i \leftarrow \mathsf{G}(s_i)$ for a random $s_i \xleftarrow{R} \mathcal{S}$, as well as a key $k \in \mathcal{Y}$, and outputs a guess b' for whether $k = \mathsf{F}(\mathsf{ek}_1, \{(z_i)\}_{i \in [g]}$ or $k \xleftarrow{R} \mathcal{Y}$. Call this **Game 0**. Define **Game 1** as the modification where $z_i \xleftarrow{R} \mathcal{Z}$. The security of G implies that **Game 0** and **Game 1** are indistinguishable. Next define **Game 2** as identical to **Game 1**, except that the challenger outputs a random bit and aborts if any of the z_i are in the range of G. Since $|\mathcal{S}|/|\mathcal{Z}| < \mathsf{negl}$, this abort condition occurs with negligible probability, meaning $\mathcal{B}$ still has non-negligible advantage in **Game 2**. We construct an adversary $\mathcal{A}$ for WPRF relative to sampler $\mathcal{D}_g$ as follows: $\mathcal{A}$, on input $\mathsf{ek}, \{z_i\}_{i \in [g]}, k$ (where $\{z_i\} \xleftarrow{R} \mathcal{D}_g$), sorts the z_i, and then sets $\mathsf{ek}_1 = \mathsf{ek}$. For $i > 1$, $\mathcal{A}$ runs $(\mathsf{fk}_i, \mathsf{ek}_i) \xleftarrow{R} \mathsf{Gen}(\lambda, R_g)$. It then gives $\mathcal{A}$ $\{(z_i, \mathsf{ek}_i)\}_{i \in [g]}, k$. Note that for key generation, $\mathsf{ek}_1 = \mathsf{ek}$ is chosen. Also, $(z_1, \ldots, z_g)$ is chosen at random in $\mathcal{Z}^g$, and $\mathcal{A}$'s challenger aborts if any of the z_g are in the range of G (that is, if $(z_1, \ldots, z_g)$ has a witness under R_g). Therefore, the view of $\mathcal{B}$ as a subroutine of $\mathcal{A}$ and the view of $\mathcal{B}$ in **Game 2** are identical. Therefore, the advantage of $\mathcal{A}$ is also non-negligible, a contradiction. □

Adaptive Security. In semi-static or active security (defined by Boneh and Zhandry [BZ14]), the same published values pv_j are used in many key exchanges, some involving the adversary. Obtaining semi-static or adaptive security from even the strongest forms of witness PRFs is not immediate. The issue, as noted by Boneh and Zhandry in the case of obfuscation, is that, even in the semi-static setting, the adversary may see the output of Eval on honest secrets, but using a malicious key ek. It may be possible for a malformed key to leak the honest secrets, thereby allowing the scheme to be broken. In more detail, consider an adversary $\mathcal{A}$ playing the role of user i, and suppose the maximum number of users in any group is 2. $\mathcal{A}$ generates and publishes params_i in a potentially malicious way (and also generates and publishes some z_i). Meanwhile, an honest user j publishes an honest ek_j and $z_j = G(s_j)$. Now, if $z_i < z_j$, user j computes the shared key for the group $\{i, j\}$ as $\mathsf{Eval}(\mathsf{ek}_i, (z_i, z_j), s_j, 2)$. While an honest ek_i would cause Eval to be independent of the witness, it may be possible for a dishonest ek_i to cause Eval to leak information about the witness.

Boneh and Zhandry circumvent this issue by using a special type of signature scheme, which they call a *puncturable* signature scheme, and only inputting signatures into Eval. Even if the entire signature leaks, it will not help the adversary produce the necessary signature to break the scheme. Such signature schemes can be build from witness indistinguishable proofs. It is straightforward to adaptive Boneh and Zhandry's construction to use witness PRFs instead of obfuscation. We omit the details.

4.3 Poly-many hardcore bits for any one-way function

A hardcore function for a function $f : \mathcal{S} \to \mathcal{Z}$ is a function $h : \mathcal{S} \to \mathcal{Y}$ such that $(f(s), h(s))$ for a random $s \xleftarrow{R} \mathcal{S}$ is indistinguishable from $(f(s), y)$ for a random $s \xleftarrow{R} \mathcal{S}$ and random $y \xleftarrow{R} \mathcal{Y}$. We now give our construction, based on the construction of [BST14]:

Construction 4.10. Let $f : \mathcal{S} \to \mathcal{Z}$ be any one-way function. Let $\mathsf{WPRF} = (\mathsf{Gen}, \mathsf{F}, \mathsf{Eval})$ be a witness PRF. We build a function $h : \mathcal{S} \to \mathcal{Y}$ as follows:

- Define $R_f(x, s) = 1$ if and only if $x = f(s)$.
- Run $(\mathsf{fk}, \mathsf{ek}) \xleftarrow{R} \mathsf{Gen}(\lambda, R)$.
- Define $h(s) = \mathsf{Eval}(\mathsf{ek}, f(s), s)$.

For security, let $\mathcal{D}_f$ be the following R_f-instance sampler: choose a random $s^* \in \mathcal{S}$, compute $z^* = f(s^*)$, and output $(z^*, \mathsf{Aux} = ())$.

Theorem 4.11. *If f is a one-way function and $(\mathsf{Gen}, \mathsf{F}, \mathsf{Eval})$ is extractable non-interactively secure for relation R_f and sampler $\mathcal{D}_f$, then h in Construction 4.10 is hardcore for f.*

Proof. Let $\mathcal{A}$ be an adversary that distinguishes h from random with inverse polynomial probability $1/q_{\mathcal{A}}$. That is, given $f(s^*)$ for a random s^*, $\mathcal{A}$ is able to distinguish $h(s^*)$ from a random string. Then $\mathcal{A}$ is actually a non-interactive adversary for WPRF relative to relation R and instance sampler $\mathcal{D}_f$. In other words, $\mathtt{WPRF.Adv}^{R}_{\mathcal{D},\mathcal{A}}(\lambda) \geq 1/q_{\mathcal{A}}$. The extracting security of the witness PRF implies that there is an extractor $\mathcal{E}$ and polynomial $q_{\mathcal{E}}$ such that $f(\mathcal{E}(\mathsf{ek}, s^*)) = s^*$. In otherwords, $\mathcal{E}$ breaks the one-wayness of f, reaching a contradiction. □

4.4 Witness Encryption

We show how to build witness encryption from witness PRFs. A witness encryption scheme is parameterized by a relation $R : \mathcal{X} \times \mathcal{W} \to \{0,1\}$, and consists of the following algorithms:

- $\mathsf{Enc}(\lambda, x, m)$ outputs a ciphertext c
- $\mathsf{Dec}(x, w, c)$ outputs a message m or $\perp$. For correctness, we require that if $R(x,w) = 1$, then $\mathsf{Dec}(x, w, \mathsf{Enc}(\lambda, x, m)) = m$ and if $R(x,w) = 0$, $\mathsf{Dec}(x, w, c) = \perp$.

For security, we use a notion similar to Bellare and Hoang [BH13], but with a minor modification. We have the notion of an R-instance sampler $\mathcal{D}$, which takes the security parameter λ and samples an instance x and auxilliary information Aux[12]. Let $\texttt{EXP}^R_{\mathcal{D},\mathcal{A}}(b, \lambda)$ denote the following experiment on an adversary $\mathcal{A}$: Run $(x, \mathsf{Aux}) \xleftarrow{R} \mathcal{D}(\lambda)$, and then run $\mathcal{A}(x, \mathsf{Aux})$. At some point, $\mathcal{A}$ produces a pair of messages (m_0, m_1), to which the challenger responds with $\mathsf{Enc}(\lambda, x, m_b)$. $\mathcal{A}$ then outputs a guess b' for b. The challenger checks if there is a w such that $R(x,w) = 1$ (that is, checks if $x \in L$), and if so, outputs a random bit. Otherwise, the challenger outputs b'. We define W_b to be the event of outputting 1 in experiment b, and define the advantage of $\mathcal{A}$ to be $\texttt{WENC.Adv}^R_{\mathcal{D},\mathcal{A}}(\lambda) = |\Pr[W_0] - \Pr[W_1]|$.

Definition 4.12. A witness encryption scheme is soundness secure for an R-instance sampler $\mathcal{D}$ if, for all adversaries $\mathcal{A}$, there is a negligible function negl such that $\texttt{WENC.Adv}^R_{\mathcal{D},\mathcal{A}}(\lambda) < \mathsf{negl}(\lambda)$.

We can also define an extractability definition where we remove the check that $x \in L$, and require that any distinguishing adversary gives rise to an extractor that can find a witness.

Our construction is the following:

Construction 4.13. Let R be a relation, and let $(\mathsf{Gen}, \mathsf{F}, \mathsf{Eval})$ be a witness PRF for R. We define a witness encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ where:

- $\mathsf{Enc}(\lambda, x, m)$ computes $(\mathsf{fk}, \mathsf{ek}) \xleftarrow{R} \mathsf{Gen}(\lambda, R)$, $K \leftarrow \mathsf{F}(\mathsf{fk}, x)$, and $c = K \oplus m$. Output the ciphertext (ek, c).
- $\mathsf{Dec}(x, w, (\mathsf{ek}, c))$ checks that $R(x,w) = 1$, and aborts otherwise. Then it computes $K \leftarrow \mathsf{Eval}(\mathsf{ek}, x, w)$, and outputs $c \oplus K$.

Correctness is immediate from the correctness of $(\mathsf{Gen}, \mathsf{F}, \mathsf{Eval})$. Moreover, we have the following straightforward security theorem:

Theorem 4.14. *If* $\mathsf{WPRF} = (\mathsf{Gen}, \mathsf{F}, \mathsf{Eval})$ *is non-interactively secure for relation* R *and* R*-instance generator* $\mathcal{D}$*, then Construction 4.13 is soundness secure for relation* R *and* R*-instance* $\mathcal{D}$ *(treated as a static instance generator for* WPRF*). Moreover, if* WPRF *is extracting, then so is Construction 4.13.*

We omit the proof, and instead present a stronger variant of witness encryption that we will prove secure.

[12] In [BH13], the sampler also outputs the challenge messages m_0, m_1. We let the adversary produce m_0, m_1. As Aux can contain the challenge, our notion is potentially stronger

4.4.1 Reusable Witness Encryption

All current witness encryption schemes, including ours above, have long ciphertexts and relatively inefficient encryption algorithms. This is due to the inefficient setup procedure for the underlying multilinear maps. In this section, we explore the setting where various messages are being witness encrypted to multiple instances, and try to amortize the computation and ciphertext length over many ciphertexts. More precisely, we define a reusable witness key encapsulation mechanism:

Definition 4.15. A reusable witness encryption scheme is a triple of algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ where:

- Gen takes as input a security parameter λ and a relation R, and outputs public parameters params as well as a master decryption key dk.
- Enc takes as input an instance x and the parameters params. It outputs a header Hdr and message encryption key k.
- Dec takes as input an instance x, header Hdr, witness w, and parameters params. It outputs a message encryption key k or $\perp$.
- Alternatively, Dec takes as input the master decryption key dk, instance x, and header Hdr (no witness), and outputs k

For correctness, we require for all (Hdr, k) outputted by $\mathsf{Enc}(\mathsf{params}, x)$, and all w such that $R(x, w) = 1$, that $\mathsf{Dec}(\mathsf{params}, x, \mathsf{Hdr}, w) = k$. We also require that $\mathsf{Dec}(\mathsf{dk}, x, \mathsf{Hdr}) = k$.

We observe that from a functionality perspective, if we ignore the master decryption key, witness encryption and reusable witness encryption are equivalent concepts: any witness encryption scheme is a reusable with a Gen algorithm that does nothing but output the security parameter, and any reusable witness encryption scheme can be converted into a regular witness encryption scheme by having the encryption procedure run Gen, and output the public parameters with the ciphertext. In this case, the computationally expensive part of encryption, namely Enc, must be run for every encryption. Moreover, the ciphertexts are large, whereas the public key is minimal. However, if a witness encryption scheme had a computationally expensive Gen, but much more efficient Enc, then the computationally expensive step only needs to be carried out once. Moreover, if ciphertexts are small, even if public keys are large the total communication will be less if running many encryptions. Therefore, we focus on building reusable witness encryption where the ciphertext are short and encryption procedures are relatively efficient.

We now give a security definition for reusable witness encryption. Let $\mathcal{D}$ be a PPT algorithm that takes as input parameters $\mathsf{params} \xleftarrow{R} \mathsf{Gen}(\lambda, R)$, is allowed to make decryption queries $\mathsf{Dec}(\mathsf{sk}, \cdot, \cdot)$, and outputs an instance x^* along with auxilliary information Aux. We call $\mathcal{D}$ an R-instance sampler for WENC.

For any instance sampler $\mathcal{D}$, let $\mathtt{EXP}^R_{\mathcal{D},\mathcal{A}}(b, \lambda)$ denote the following experiment on a PPT algorithm $\mathcal{A}$: run $\mathsf{params} \xleftarrow{R} \mathsf{Gen}(\lambda, R)$ and $(x^*, \mathsf{Aux}) \xleftarrow{R} \mathcal{D}(\mathsf{params})$. Let $(\mathsf{Hdr}^*, k_0) \xleftarrow{R} \mathsf{Enc}(\mathsf{params}, x)$ and $k_1 \xleftarrow{R} \mathcal{Y}$. Run $b' \xleftarrow{R} \mathcal{A}^{\mathsf{Dec}(\mathsf{sk},\cdot,\cdot)}(\mathsf{params}, x^*, \mathsf{Aux}, \mathsf{Hdr}^*, k_b)$, with the requirement that the oracle $\mathsf{Dec}(\mathsf{sk}, \cdot, \cdot)$ outputs $\perp$ on query (x^*, Hdr^*). If there is a witness w such that $R(x^*, w) = 1$ (in other words, if $x^* \in L$), then output a random bit and abort. Otherwise, output b'.

We define W_b to be the event of outputting 1 in experiment b, and define the advantage of $\mathcal{A}$ to be $\mathtt{WENC.Adv}^R_{\mathcal{D},\mathcal{A}}(\lambda) = |\Pr[W_0] - \Pr[W_1]|$.

Definition 4.16. $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a CCA secure reusable witness encryption scheme for a relation R and instance sampler $\mathcal{D}$ if, for all PPT adversaries $\mathcal{A}$, there is a negligible function negl such that $\mathtt{WENC.Adv}^{R}_{\mathcal{D},\mathcal{A}}(\lambda) < \mathsf{negl}(\lambda)$.

We can also get a CPA definition is we do not allow $\mathcal{A}$ to make decryption queries[13]. We can similarly get an extractable definition, where we remove the check that $x^* \in L$, and instead have any $\mathcal{A}$ with non-negligible advantage imply an extractor that can find a witness that $x^* \in L$.

Remark 4.17. We note that while our notion of re-usable witness encryption is new and has not been realized before, it is straightforward to build re-usable encryption from obfuscation by adapting the construction of [GGH+13b]. However, our scheme will be more efficient as it is built from witness PRFs instead of obfuscation.

Our Construction Our construction of reusable witness encryption is the following:

Construction 4.18. Let $(\mathsf{WPRF.Gen}, \mathsf{F}, \mathsf{Eval})$ be a witness PRF, and let $\mathsf{G} : \mathcal{S} \to \mathcal{Z}$ be a pseudorandom generator with $|\mathcal{S}|/|\mathcal{Z}| < \mathsf{negl}$. Construct the following public key witness encryption scheme $(\mathsf{WENC.Gen}, \mathsf{Enc}, \mathsf{Dec})$:

- $\mathsf{WENC.Gen}(\lambda, R)$: Suppose $R : \mathcal{X} \times \mathcal{W} \to \{0,1\}$. Assume that $\mathcal{S}$ and $\mathcal{W}$ are disjoint. Let $\mathcal{X}' = \mathcal{X} \times \mathcal{Z}$ and $\mathcal{W}' = \mathcal{W} \bigcup \mathcal{S}$. Finally, let $R' : \mathcal{X}' \times \mathcal{W}' \to \{0,1\}$ be the following function:

$$R'((x,z),w') = \begin{cases} R(x,w) & \text{if } w' = w \in \mathcal{W} \\ 1 & \text{if } w' = s \in \mathcal{S} \text{ and } \mathsf{G}(s) = z \\ 0 & \text{if } w' = s \in \mathcal{S} \text{ and } \mathsf{G}(s) \neq z \end{cases}$$

  That is, if w' is a witness for R, R' checks if w is valid for x. Otherwise, w' is a seed for G, and R' checks if the seed generates z.

  Now, run $(\mathsf{fk}, \mathsf{ek}) \xleftarrow{R} \mathsf{WPRF.Gen}(\lambda, R')$ and output $\mathsf{params} = \mathsf{ek}$ and $\mathsf{dk} = \mathsf{fk}$.

- $\mathsf{Enc}(\mathsf{ek}, x)$: Choose a random seed $s \xleftarrow{R} \mathcal{S}$, and let $z \leftarrow \mathsf{G}(s)$. Run $k \leftarrow \mathsf{Eval}(\mathsf{ek}, (x,z), s)$. Output z as the header, and k as the message encryption key.

- $\mathsf{Dec}(\mathsf{ek}, x, z, w) = \mathsf{Eval}(\mathsf{ek}, (x,z), w)$

- $\mathsf{Dec}(\mathsf{fk}, x, z) = \mathsf{F}(\mathsf{fk}, (x,z))$

Ciphertexts in our system are very compact: not including the instance, the size is $|z| + |m|$. Thus, the ciphertext size is equal to the length of the message plus a term proportional to the security parameter, which is essentially optimal for public key encryption schemes. The compactness of our ciphertexts allows us, for example, to use our re-usable witness encryption scheme to instantiate the attribute-based encryption scheme of Garg et al. [GGSW13]. The resulting attribute-based encryption scheme has ciphertexts whose size is equal to the sum of the size of the attribute, the size of the message, and a term proportional to the security parameter.

Notice that for $z = \mathsf{G}(s)$, s is always a witness for (x,z) relative to R'. Moreover, if w is a witness for x relative to R, it is also a witness for (x,z) relative to R' for all z. Correctness immediately

[13] If we still consider samplers that can make decryption queries, this notion is similar to CCA1 security for standard public key encryption.

follows. For security, note that a valid encryption is indistinguishable from an encryption generated by choosing a random $z \xleftarrow{R} \mathcal{Z}$ and computing $k \leftarrow \mathsf{F}(\mathsf{sk}_R, (x, z))$. However, assuming $\mathcal{Z}$ is much bigger than $\mathcal{S}$, with high probability z will not have a pre-image under G. Thus, if x has no witness relative to R, (x, z) will have no witness relative to R', meaning k is indistinguishable from random. Security follows. Before proving the statement, we define the following algorithm:

Let $\mathcal{D}_{\mathsf{WENC}}$ be an R-instance sampler for WENC. We construct a R'-instance sampler for WPRF, called $\mathcal{D}_{\mathsf{WPRF}}$, as follows. On input ek, run $\mathcal{D}_{\mathsf{WENC}}$ on $\mathsf{params} = \mathsf{ek}$. When $\mathcal{D}_{\mathsf{WENC}}$ makes a CCA query an instance $x \in \mathcal{X}$ and header $\mathsf{Hdr} = z$, make a F query on (x, z), and responds with the resulting value. When $\mathcal{D}_{\mathsf{WENC}}$ outputs an instance $x^* \in \mathcal{X}$ and auxiliary information Aux, produce a random string $z^* \in \mathcal{Z}$ and output the instance (x^*, z^*) and Aux.

Theorem 4.19. *If* G *is a secure pseudorandom generator and* $\mathsf{WPRF} = (\mathsf{WPRF.Gen}, \mathsf{F}, \mathsf{Eval})$ *is adaptive witness interactively (resp. non-interactively) secure for relation* R' *and instance sampler* $\mathcal{D}_{\mathsf{WPRF}}$*, then Construction 4.18 is a CCA (resp. CPA) secure re-usable witness encryption scheme for relation* R *and* R*-instance sampler* $\mathcal{D}_{\mathsf{WENC}}$*. Moreover, if* WPRF *is extracting, then so is* WENC*.*

Proof. We prove the CCA non-extracting case, the others being similar. Let $\mathcal{B}$ be a CCA adversary for $(\mathsf{WENC.Gen}, \mathsf{Enc}, \mathsf{Dec})$ with non-negligible advantage. We define **Game 0** as the standard attack game, and define **Game 1** as the alternate attack game where y^* is chosen as a random string. If $\mathcal{B}$ can distinguish the two cases, then we could construct an adversary breaking the security of G. Now we use $\mathcal{B}$ to build an adversary $\mathcal{A}$ for WPRF and instance sampler $\mathcal{D}_{\mathsf{WPRF}}$. On input $(\mathsf{ek}, (x^*, z^*), \mathsf{Aux}, k)$, $\mathcal{A}$ simulates $\mathcal{B}$ on input $(\mathsf{ek}, x^*, \mathsf{Hdr} = z^*, \mathsf{Aux}, k)$. Whenever $\mathcal{B}$ makes a CCA query on (x, z), $\mathcal{A}$ makes a F query on (x, z), and forwards the response to $\mathcal{B}$. $\mathcal{A}$ outputs the output of $\mathcal{B}$. Notice that the view of $\mathcal{B}$ is identical to that in **Game 1**. Moreover, with overwhelming probability, z^* is not in the image of G, so (x^*, z^*) is a valid instance for relation R' exactly when x^* is a valid instance for relation R. Therefore, the advantage of $\mathcal{A}$ is negligibly close to the advantage of $\mathcal{B}$, and is therefore non-negligible. □

4.5 Secret Sharing for mNP

We define the notion of re-usable secret sharing for mNP. Similarly to reusable witness encryption, re-usable secret sharing attempts to amortize an expensive setup procedure over many sharings. mNP is the class of *monotone* NP languages, meaning that if $\mathbf{x} \in \{0,1\}^n$ is in L with witness w, and $\mathbf{x}' \in \{0,1\}^n$ is an instance such that $x_i = 1 \Rightarrow x'_i = 1$, then $\mathbf{x}'$ is also in L and w is also a witness for $\mathbf{x}'$. Such languages are characterized by a relation $R : \{0,1\}^n \times \mathcal{W} \to \{0,1\}$ such that there are no NOT gates in any of the paths from the first set of input wires to the output, and $\mathbf{x} \in L$ if and only if there is a w such that $R(\mathbf{x}, w) = 1$.

Intuitively, in re-usable secret sharing for an mNP language L, a trusted party publishes parameters params, which allows anyone to secret share to sets of users in the language L. More precisely, we define the notion of a re-usable secret sharing key encapsulation mechanism.

Definition 4.20. A reusable secret sharing scheme for mNP is a triple of PPT algorithms $(\mathsf{Gen}, \mathsf{Share}, \mathsf{Recon})$ where:

- $\mathsf{Gen}(\lambda, R)$ takes as input a relation accepting n-bit instances, produces a public key params.
- $\mathsf{Share}(\mathsf{params})$ produces shares s_i for user i, and a secret encryption key $k \in \mathcal{Y}$.

- $\mathsf{Recon}(\mathsf{params}, \{s_i\}_{i\in S}, w)$ Outputs either $\perp$ or a key $k \in \mathcal{Y}$. For correctness, we require that if $R(\mathbf{x}, w) = 1$ where $x_i = 1$ if and only if $i \in S$, then Recon outputs the correct k, and for $R(\mathbf{x}, w) = 0$, Recon outputs $\perp$.

Let $\mathcal{D}$ be an algorithm that, on input security parameter λ, outputs an instance $\mathbf{x} \in \{0,1\}^n$ and auxilliary information Aux. Associate $\mathbf{x}$ with the set $S \subseteq [n]$ where $i \in S$ if and only if $x_i = 1$. We call $\mathcal{D}$ an R-instance sampler. Let $\mathtt{EXP}^R_{\mathcal{D},\mathcal{A}}(b, \lambda)$ be the following experiment on a PPT adversary $\mathcal{A}$: run $(\mathbf{x}, \mathsf{Aux}) \xleftarrow{R} \mathcal{D}(\lambda)$ and $\mathsf{params} \xleftarrow{R} \mathsf{Gen}(\lambda, R)$ and $(\{s_i\}_{i\in[n]}, k_0) \xleftarrow{R} \mathsf{Share}(\mathsf{params})$ and $k_1 \xleftarrow{R} \mathcal{Y}$, and give $\mathcal{A}$ the shares $\{s_i\}_{i\in S}$, Aux and the key k_b. $\mathcal{A}$ produces a guess b'. Let W_b be the event of outputting 1 in experiment b. Define $\mathtt{SS.Adv}^R_{\mathcal{D},\mathcal{A}}(\lambda) = |\Pr[W_0] - \Pr[W_1]|$.

Definition 4.21. $(\mathsf{Gen}, \mathsf{Share}, \mathsf{Recon})$ is secure for a relation R and instance sampler $\mathcal{D}$ if, for all adversaries $\mathcal{A}$, there is a negligible function negl such that $\mathtt{SS.Adv}^R_{\mathcal{D},\mathcal{A}}(\lambda) < \mathsf{negl}(\lambda)$.

Remark 4.22. We note that while the witness encryption-based scheme of [KNY14] is not re-usable in general, applying their construction to our reusable witness encryption scheme gives a reusable secret sharing scheme. Below, we give a a related construction directly from witness PRFs in order to demonstrate the improvement in share size.

Construction. For simplicity, we will assume that for every input length n, the language L contains a string $\mathbf{x} \in \{0,1\}^n$ and valid witness w that are both easy to compute[14]. We require this so that the sharer can compute the secret encryption key without knowing the secrets used in the Gen algorithm. It is straightforward to adapt our scheme to the setting where this is not the case.

Construction 4.23. Let $\mathsf{G} : \mathcal{S} \to \mathcal{Z}$ be a pseudorandom generator, and $(\mathsf{WPRF.Gen}, \mathsf{F}, \mathsf{Eval})$ be a witness PRF.

- $\mathsf{SS.Gen}(\lambda, R,)$: for a mNP relation $R : \{0,1\}^n \times \mathcal{W} \to \{0,1\}$, let $R' : \mathcal{Z}^n \times (\mathcal{S}^n \times \mathcal{W}) \to \{0,1\}$ be the following NP relation: on instance $\{z_i\}_{i\in[n]}$ and witness $\{s_i\}_{i\in[n]}, w$, compute $\mathbf{x} \in \{0,1\}^n$ where $x_i = \begin{cases} 1 & \text{if } \mathsf{G}(s_i) = z_i \\ 0 & \text{if } \mathsf{G}(s_i) \neq z_i \end{cases}$. Then compute $R(\mathbf{x}, w)$.

  Let L be the langauge defined by R. The language L' defined by R' is the set of $\{z_i\}_{i\in[n]}$ such that there is a subset of $S \subseteq [n]$ where

  - z_i has a pre-image under G for all $i \in S$
  - S corresponds to an instance $\mathbf{x} \in \{0,1\}^n$ such that $\mathbf{x} \in L$, where $x_i = 1$ if and only if $i \in S$.

  Run $(\mathsf{fk}, \mathsf{ek}) \xleftarrow{R} \mathsf{WPRF.Gen}(\lambda, R')$. Output $\mathsf{params} = \mathsf{ek}$.

- $\mathsf{Share}(\mathsf{params})$: Sample $s_i \xleftarrow{R} \mathcal{S}$ for $i \in [n]$ and compute $z_i = \mathsf{G}(s_i)$. Compute some instance $\mathbf{x}$ and witness w for R, and let $w' = (\{s_i\}_{x_i=1}, w)$. Compute $k \leftarrow \mathsf{Eval}(\mathsf{ek}, \{z_i\}_{i\in[n]}, w')$. The share for user i is $(s_i, \{z_j\}_{j\in[n]})$, and the secret encryption key is k.

[14] This is a natural requirement. Consider the setting where every user corresponds to an edge on a graph with m vertices, and we associate to every set of users the graph graph consisting of the user edges. We secret share to sets of users whose corresponding graph contains a Hamiltonian cycle. We can set $\mathbf{x}$ to be the complete graph on m vertices, and pick an arbitrary permutation on the vertices as the witness.

- $\mathsf{Recon}(\mathsf{ek}, \mathbf{x}, \{z_i\}_{i\in[n]}, \{s_i\}_{x_i=1}, w)$: check that $R(\mathbf{x}, w) = 1$ and that $\mathsf{G}(s_i) = z_i$ for each i where $x_i = 1$. For each i where $x_i = 0$, let $s'_i = \perp$, and let $s'_i = s_i$ for all other i. Let $w' = (\{s'_i\}_{i\in[n]}, w)$, and compute $k \leftarrow \mathsf{Eval}(\mathsf{ek}, \{z_i\}_{i\in[n]}, w')$.

Note that the size of a share is $|s| + n|z| \in O(n\lambda)$. However, the $\{z_i\}_{i\in[n]}$ are shared among all users and can therefore be transmitted in a single broadcast. In this way, the amortized share size per user is only $O(\lambda)$.

For security, the idea is that a set S of shares corresponding to an instance $\mathbf{x} \notin L$ cannot tell whether $\{z_i\}_{i\in[n]}$ is in the language L' or not. This is because all of the z_i where $i \notin S$ can be replaced with random strings, and the adversary cannot tell the difference. However, now these z_i have (with overwhelming probability) no pre-image under G. Therefore, all the subsets S' where z_i have pre-images under G must be subsets of S, and therefore correspond to instances not in L. This means $\{z_i\}_{i\in[n]}$ is no longer in the language. At this point, witness PRF security shows that the secret encryption key is hidden from the adversary, as desired.

More formally, let $\mathcal{D}_{\mathsf{SS}}$ be an R-instance sampler for $\mathsf{SS} = (\mathsf{SS.Gen}, \mathsf{Share}, \mathsf{Recon})$. Define the following static R'-instance sampler $\mathcal{D}_{\mathsf{WPRF}}$ for WPRF. Run $\mathcal{D}_{\mathsf{SS}}$ to obtain (x, Aux). Let $S \subseteq [n]$ be the set where $i \in S$ if and only if $y_i = 1$. For $i \in S$, sample random $s_i \xleftarrow{R} \mathcal{S}$ and $z_i \leftarrow \mathsf{G}(s_i)$. For all other i, let $z_i \xleftarrow{R} \mathcal{Z}$. Output $(\{z_i\}_{i\in[n]}, \mathsf{Aux}' = (\mathsf{Aux}, \{s_i\}_{i\in S}))$.

Theorem 4.24. *If* WPRF *is static witness non-interactively secure for relation* R' *and instance sampler* $\mathcal{D}_{\mathsf{WPRF}}$*, then* SS *is secure for relation* R *and instance sampler* $\mathcal{D}_{\mathsf{SS}}$*.*

Proof. Let $\mathcal{B}$ be an adversary for SS. Construct the following adversary $\mathcal{A}$ for WPRF. On input $\mathsf{ek}, \mathbf{x}, \{z_i\}_{i\in[n]}, \mathsf{Aux}, \{s_i\}_{i\in S}, k$, $\mathcal{A}$ runs $\mathcal{B}$ on input $\mathsf{ek}, \mathbf{x}, \{z_i\}_{i\in[n]}, \mathsf{Aux}, \{s_i\}_{i\in S}, k$. When $\mathcal{B}$ outputs a bit b', $\mathcal{A}$ outputs b'. Notice that the view of $\mathcal{B}$ as a subroutine of $\mathcal{A}$ is indistinguishable from the correct view of $\mathcal{B}$: the only difference are the z_i for $i \notin S$, which are generated randomly instead of pseudorandomly. Therefore, the advantage of $\mathcal{A}$ is negligible close to the advantage of $\mathcal{B}$, so the security of WPRF implies that they must both be negligible. □

4.6 Fully Distributed Broadcast Encryption

Boneh and Zhandry [BZ14] show that key exchange with small parameters gives a form of distributed broadcast encryption with short ciphertexts. In distributed broadcast encryption, each user generates their own secret key rather than having the secret key generated by a trusted authority. However, their system had large public keys. Ananth et al. [ABG+13] show how to reduce the public key size as well, but at the cost of losing the distributed property of the system. Achieving small public keys for a distributed broadcast scheme seems problematic, as each user must publish some value dependent on their secret key, so the total amount of public data is at least linear in the number of users. Another drawback of the distrubted encryption definition of Boneh and Zhandry is that part of the public key still needs to be computed by a trusted party.

We now put forward the notion of a *fully-distributed broadcast scheme*. In such a scheme, all parties are stateful, and keep a small secret key. There is also a small global public key, posted to some public bulletin board. When a new user joins the system, the user reads off the global public key, and generates their own secret key. Then the user publishes a user public key. All of the existing users use the new user public key to update their secret keys. Finally, the global public key is updated to incorporate the new user. Anyone is able to update the global public key. In this system, there is no a priori bound on the number of users.

Definition 4.25. Fully-dynamic broadcast encryption.

- $\mathsf{Init}(\lambda)$ is deterministic, and outputs an initial global public key $\mathsf{params}^{(0)}$.
- $\mathsf{Join}(\mathsf{params}^{(n)})$ generates a user secret key $\mathsf{sk}_{n+1}^{(n+1)}$ and user public key pk_{n+1} for user $n+1$. The user then publishes pk_{n+1}.
- $\mathsf{Update}(\mathsf{sk}_i^{(n)}, \mathsf{pk}_{n+1})$ generates a new user secret key $\mathsf{sk}_i^{(n+1)}$ for user i.
- $\mathsf{Inc}(\mathsf{params}^{(n)}, \mathsf{pk}_{n+1})$ is deterministic, and produces an updated global public key $\mathsf{params}^{(n+1)}$.
- $\mathsf{Enc}(\mathsf{params}^{(n)}, S)$ takes as input a subset $S \subseteq [n]$, and produces a header Hdr and message encryption key k.
- $\mathsf{Dec}(\mathsf{sk}_i^{(n)}, S, \mathsf{Hdr})$ checks if $i \in S$, and if so outputs the key k. Otherwise, output $\perp$.

Initially, $\mathsf{Init}(\lambda)$ is run, to obtain $\mathsf{params}^{(0)}$. Since Init is deterministic, there are no secrets involved and anyone can run it, arriving at the same value. $\mathsf{params}^{(0)}$ is then posted to a public bulletin board. Afterward, users will join the system by running Join, which generates a user secret key $\mathsf{sk}_{n+1}^{(n+1)}$, and a user public key pk_{n+1}. The user then publishes pk_{n+1}. All other users run Update on pk_{n+1} and their own secret key, to generate a new user secret key. Inc is also run to incorporate pk_{n+1} into the public parameters. Since Inc is deterministic, anyone can run it and arrive at the same value. In particular, users can verify that the public parameters are updated correctly. Finally, when someone wishes to encrypt a message, they will run Enc on the current public parameters, generating a ciphertext. Anyone in the intended recipient set can then decrypt using Dec.

For security, we consider an adaptive notion. In this notion, the adversary can control arbitrary subsets of users, and can adaptively corrupt them. We do not allow the adversary to alter the public key $\mathsf{params}^{(0)}$, except by joining new users to the system. This is a reasonable requirement, as any of the other users could keep a copy of the global public key and always make sure the key is correct and not tampered with.

Consider the following experiment $\mathtt{EXP}_{\mathcal{A}}(b, \lambda)$, played between an adversary $\mathcal{A}$ and a challenger:

- The challenger runs $\mathsf{Init}(\lambda)$ to obtain a global public key $\mathsf{params}^{(0)}$, which it then provides to $\mathcal{A}$. It also initializes a counter n to 0, and a set T to $\{\}$.
- $\mathcal{A}$ can make *register honest* queries on empty input. The challenger runs the following:

$$\begin{aligned}(\mathsf{sk}_{n+1}^{(n+1)}, \mathsf{pk}_{n+1}) &\xleftarrow{R} \mathsf{Join}(\mathsf{params}^{(n)}) \\ \mathsf{sk}_i^{(n+1)} &\leftarrow \mathsf{Update}(\mathsf{sk}_i^{(n)}, \mathsf{pk}_{n+1}) \text{ for } i \in T \\ \mathsf{params}^{(n+1)} &\leftarrow \mathsf{Inc}(\mathsf{params}^{(n)}, \mathsf{pk}_{n+1}) \\ T &\leftarrow T \bigcup \{n+1\} \\ n &\leftarrow n+1\end{aligned}$$

  The challenger then supplies the new public key $\mathsf{params}^{(n+1)}$ and the user public key $\mathsf{pk}_{n+1}^{(n+1)}$ to $\mathcal{A}$

- $\mathcal{A}$ can make *register corrupt* queries on input pk_{n+1}. The challenger runs the following:

$$\begin{aligned} \mathsf{sk}_i^{(n+1)} &\leftarrow \mathsf{Update}(\mathsf{sk}_i^{(n)}, \mathsf{pk}_{n+1}) \text{ for } i \in T \\ \mathsf{params}^{(n+1)} &\leftarrow \mathsf{Inc}(\mathsf{params}^{(n)}, \mathsf{pk}_{n+1}) \\ n &\leftarrow n+1 \end{aligned}$$

  The challenger then supplies the new public key $\mathsf{params}^{(n+1)}$ to the adversary.

- $\mathcal{A}$ can make *corrupt user* queries on input $i \in T$. The challenger responds by setting $T \leftarrow T \setminus \{i\}$, and giving $\mathsf{sk}_i^{(n)}$ to the adversary.

- $\mathcal{A}$ can make a single *challenge query* on target set $S \subseteq T$. The challenger then computes $(\mathsf{Hdr}^*, k_0) \xleftarrow{R} \mathsf{Enc}(\mathsf{params}^{(n)}, S)$, and lets $k_1 \xleftarrow{R} \mathcal{K}$. The challenger gives $\mathcal{A}$ the pair $(\mathsf{Hdr}^*, k^* = k_b)$.

- Finally, $\mathcal{A}$ outputs a guess b' for b.

We define W_b to be the event of outputting 1 in experiment b, and define the advantage to be $\mathtt{BE.Adv}_{\mathcal{A}}(\lambda) = |\Pr[W_0] - \Pr[W_1]|$.

Definition 4.26. A fully-distributed broadcast encryption scheme is adaptively secure if, for all adversaries $\mathcal{A}$, there is a negligible function negl such that $\mathtt{BE.Adv}_{\mathcal{A}}(\lambda) < \mathsf{negl}(\lambda)$.

Our Construction The idea behind our construction is as follows. Each user of the system will generate a random input s to a one-way function f, and publish the corresponding output z. Clearly, if the public parameters contained all published outputs, the parameters would be linear in the size of the users. Instead, similar to the scheme of Ananth et al. [ABG+13], we use a Merkle hash tree to hash down the public parameters to a small hash. In particular, we divide the n users into at most $\lceil \log n \rceil$ groups S_j where $|S_j| = 2^j$. For each group, we compute the Merkle tree hash of the public values of that group. The public parameters are then the hashes h_j for each group G_j. The secret key for a user in S_j will be their random s, as well as a "proof" that the corresponding output x was one of the leaves in the hash tree for h_j. The proof will consist of the nodes in the path from x to h_j in the Merkle tree, as well as all of the neighbors. Any false proof will lead directly to a collision for the underlying hash function.

Adding a user is simple: the user computes a random input s to f and publishes the output x. Add x as a new hash to the public parameters. As long as there are two hashes corresponding to Merkle trees of the same height, merge the two together by hashing their roots, and replacing the two values with the new hash. Merge the corresponding groups together as well. This can all be done publicly.

If a user belongs to a group that was merged, it is easy to update their proof by merging the path to the old root with the path from the old root to the new root.

We now give the construction in more detail:

Construction 4.27. Let $f : \mathcal{S} \to \mathcal{X}$ be a one-way function, $(\mathsf{Gen}, \mathsf{F}, \mathsf{Eval})$ be a witness PRF, and $H : \mathcal{X}^2 \to \mathcal{X}$ a collision-resistant hash function.

- $\mathsf{Init}(\lambda)$: initializes an empty list $L = ()$, and outputs $\mathsf{params}^{(0)} = (\lambda, n = 0, L)$. n will be the number of users, and L will be a list of hashes. Let $n = \sum_{j \in U} 2^j$ where $U \subseteq [0, \lfloor \log n \rfloor]$. Then $[n]$ will be divided into $|U|$ different sets $S_j, j \in U$, where $|S_j| = 2^j$ (the sets U and S_j can be inferred from n and do not need to be stored in the public parameters). L will contain a hash h_j, which will be the Merkle tree hash using H of the public values of the users in S_j.

- $\mathsf{Join}(\mathsf{params}^{(n)})$: $s \xleftarrow{R} \mathcal{S}$, $x \leftarrow f(s)$. Publish $\mathsf{pk}_{n+1} = x$. Create a local copy of L, and let j_{min} be the minimum non-negative integer not in U. Define $h_{j,L} = h_j$ for $j = 0, ..., j_{min} - 1$. Create a local tree π with $h_{0,R} = x$ initially as the root. For $j = 0, ..., j_{min} - 1$, create a new root $h_{j+1,R} = H(h_{j,L}, h_{j,R})$ with $h_{j,L}$ as the left child and $h_{j,R}$ as the right child. Mark x as the "target leaf." Note that π consists of a single path between the "target leaf" and the root, plus a sibling for every non-root node. We call such a tree a *rooted binary caterpillar* (RBC) tree. Define $h_{j_{min}} = h_{j_{min},R}$, the root of the RBC tree.

  Set $T = (T \setminus \{0, ..., j_{min} - 1\}) \bigcup \{j_{min}\}$. Remove all hashes h_j for $j = 0, \ldots, j_{min} - 1$ from L, and add the hash $h_{j_{min}}$. The secret key for user $n + 1$ is L, π, x, s.

- $\mathsf{Update}(\mathsf{sk}_i^{(n)}, \mathsf{pk}_{n+1})$: Update the local copy of L as in Join, obtaining an RBC tree π. If the root of π_i (the current tree for user i) became one of the leaves on π, prune the opposite branch in π, and then merge the two trees together to obtain a new RBC tree π_i. The target leaf of π_i is still the original target leaf from π_i.

- $\mathsf{Inc}(\mathsf{params}^{(n)}, \mathsf{pk}_{n+1})$: Update L as in Join, and then discard the tree π.

- $\mathsf{Enc}(\mathsf{params}^{(n)}, S)$: Let $\ell = \lfloor \log n \rfloor$. Define $R_{\leq \ell}$ to be the relation where instances are tuples $(j, h_j, h_S^{(j)})$ with $j \leq \ell$. A witness for (j, h_j, h_S) is a tuple (π, τ, s) where π, τ are RBC trees of height j with identical structure (including the same target leaf) such that the root of π is h_j, the root of τ is h_S, the target leaf of π is $f(s)$, and the target leaf of h_S is 1.

  Choose a random key k. For each $j \in T$, do the following:

  - Run $(\mathsf{fk}_j, \mathsf{ek}_j) \xleftarrow{R} \mathsf{Gen}(\lambda, R_{\leq \ell})$.
  - Let $\mathbf{v}^{(j)} \in \mathcal{X}^{2^j}$ be the vector defined as follows: iterate through the 2^j identities $i \in S_j$, and set $\mathbf{v}_i^{(j)} = 1$ if $i \in S$ and 0 otherwise.
  - Let $h_S^{(j)}$ be the Merkle tree hash of $\mathbf{v}^{(j)}$ computed by successively hashing pairs of elements together.
  - Let $K_j = k \oplus \mathsf{F}(\mathsf{fk}_j, (j, h_j, h_S^{(j)})$.

  Set k to be the message encryption key, and the header to be $\mathsf{Hdr} = \{j, K_j, \mathsf{ek}_j\}_{j \in T}$

- $\mathsf{Dec}(\mathsf{sk}_i^{(n)}, S, \mathsf{Hdr})$: If $i \notin S$, output $\perp$. Otherwise, let $i \in S_j$ for some $j \in T$. Compute the Merkle hash $h_S^{(j)}$ of $\mathbf{v}^{(j)}$ as above. Next prune the tree down to a tree τ_i that has structure is identical to π_i, including the target leaf at position i. If $i \in S$, the the target leaf will have value 1. Next, run $k \leftarrow K_j \oplus \mathsf{Eval}(\mathsf{ek}_j, (j, h_j, h_S^{(j)}), (\pi_i, \tau_i, s))$.

Correctness is immediate from the construction. Also notice that the list L contains at most ℓ pairs, so the total size ignoring the security parameter is $O(\log n)$. The secret key for a user

contains L, as well as a BCP tree π. π has depth at most ℓ, and has at most two nodes per level. therefore, the size is $O(\log n)$. Finally, each header component contains an evaluation key ek_j for a witness PRF for relation $R_{\leq \ell}$ where $\ell \leq \log n$. The size of $R_{\leq \ell}$ is polynomial in ℓ, so the size of ek_j is $\mathsf{poly}(\log n)$. In addition, there are at most ℓ header components, so the total header size is $\mathsf{poly}(\log n)$.

For security, we define a class of instance samplers, called *acceptable*, such that the instance $(j, h_j, h_S^{(j)})$ and auxilliary information Aux computed by $\mathcal{D}$ satisfies the following:

- h_j is computed as the Merkle tree hash of some $x_i \in \mathcal{X}$ for $i \in [2^j]$ for some j.
- Each of the x_i above are computed as $x_i = f(s_i)$ for some $s_i \in \mathcal{S}$
- $h_S^{(j)}$ is computed as the Merkle tree hash of some $\mathbf{v}^{(j)} \in \mathcal{X}^{2^j}$ where $v_i^{(j)}$ is either 0 or 1 for all j.
- For each i where $v_i^{(j)} = 1$, Aux does not depend on s_i (though it may be computed from x_i).

This class of instance samplers is very restricting. In particular, Aux is computed independently of any of the "easy-to-compute" witnesses for $(j, h_j, h_S^{(j)})$ (that is, those witness containing the correct proofs, and not false proofs that lead to collisions). This makes it unlikely the counter example of [GGHW14] can be adapted to apply to this setting.

Theorem 4.28. *Suppose H is collision resistant, f is one-way, and $(\mathsf{Gen}, \mathsf{F}, \mathsf{Eval})$ is an extracting non-interactive witness PRF for all static acceptable instance samplers $\mathcal{D}$. Then Construction 4.27 is adaptively secure.*

We note that if we require interactive security for WPRF, we can modify Construction 4.27 so that all header components use a single witness PRF key, and security will be maintained. This will save roughly a $\log n$ factor in the length of the ciphertexts.

We now prove Theorem 4.28:

Proof. Let $\mathcal{B}$ be an adaptive adversary for this scheme. Taking only a polynomial hit in the security parameter, we can assume without loss of generality that there is some polynomial p such that $\mathcal{B}$ always makes the challenge query when exactly $p(\lambda)$ users are registered. We break $\mathcal{B}$ into two parts: $\mathcal{B}_0$, which stops after submitting the challenge query and returns some state state, and $\mathcal{B}_1$ which takes as input state, the challenge set S, the challenge header Hdr^* and the challenge key k^*, and ultimately outputs a bit b'.

Let $\mathcal{D}$ be the following instance sampler. $\mathcal{D}$ simulates $\mathcal{B}_0$, and plays the role of challenger to $\mathcal{B}_0$. When $\mathcal{B}_0$ outputs a challenge set S, $\mathcal{D}$ picks a random $j \xleftarrow{R} L$, computes $h_S^{(j)}$ and h_j as above. It outputs the instance $(j, h_j, h_S^{(j)})$, and the auxiliary information $\mathsf{Aux} = \mathsf{state}$. We assume that state contains all of the queries made by $\mathcal{B}_0$. We also assume that, before the challenge query on S, $\mathcal{B}_0$ has asked for all of the secret keys for users outside of S. We note that $\mathcal{D}$ is static and acceptable.

Let $\mathcal{A}$ be the following adversary for $(\mathsf{Gen}, \mathsf{F}, \mathsf{Eval})$, derived from $\mathcal{B}_1$. On input the tuple $(j, h_j, h_S^{(j)}), \mathsf{state}, \mathsf{ek}, K$, $\mathcal{A}$ chooses a random $k \xleftarrow{R} \mathcal{K}$, and sets $K_j = k \oplus K$ and $\mathsf{ek}_j = \mathsf{ek}$. For all $j' \in T \setminus \{j\}$, run $(\mathsf{fk}_{j'}, \mathsf{ek}_{j'}) \xleftarrow{R} \mathsf{Gen}(\lambda, R_{\leq \ell})$. If $j' < j$, set $\mathsf{Hdr}_{j'} = (K_{j'}, \mathsf{ek}_{j'})$ where $K_{j'} \xleftarrow{R} \mathcal{K}$. For all tuples with $j' > j$, compute $K_{j'} \leftarrow k \oplus \mathsf{F}(\mathsf{fk}_{j'}, (h_{j'}, h_S^{(j')}))$ where $h^{j'}$ and $h_S^{(j')}$ are computed as above.

Then it provides $\mathcal{A}_1$ the header $\{K_{j'}, \mathsf{ek}_{j'}\}$, state state, and key k. Then $\mathcal{A}$ runs $\mathcal{B}_1$, playing the role of challenger to $\mathcal{B}_1$.

To analyze the advantage of $\mathcal{A}$, we define the following j hybrids. In hybrid h_j, $K_{j'}$ for $j' \leq j$ are chosen randomly for the challenge header, whereas $K_{j'}$ for $j' > j$ are chosen correctly. Then Hybrid 0 corresponds to experiment 0, and hybrid ℓ corresponds to experiment 1.

Let j be the value chosen by $\mathcal{D}$. Then, in experiment 0, $\mathcal{A}$ perfectly simulates the view of $\mathcal{B}$ in Hybrid $j-1$. Meanwhile, in experiment 1, $\mathcal{A}$ perfectly simulates the view of $\mathcal{B}$ in Hybrid j. Therefore, it is straightforward to show that the advantage of $\mathcal{A}$ is at least $1/\ell$ times the advantage of $\mathcal{B}$.

If $\mathcal{B}$ had non-negligible advantage, this implies that $\mathcal{A}$ has non-negligible advantage. But then there is an extractor $\mathcal{E}$ that, on input $(j, h_j, h_S^{(j)}), \mathsf{state}, \mathsf{ek}$, is able to find a witness for $j, h^j, h^{(j)}$. We can use such a witness $w = (\pi, \tau, s)$ to break the collision resistance of H or the one-wayness of f. Let $\mathbf{v}^{(j)}$ be the values that $\mathcal{D}$ hashed to obtain h^j. Let π' be the result of pruning the Merkle tree for h^j down to the same structure as π. Let τ' be defined similarly. There are several cases:

- π and π' have different values. This means, π and π' contain a collision for H.
- τ and τ' have different values. This means τ and τ' contain a collision for H.
- $f(s)$ is equal to the target leaf in π. Given that neither of the above holds, this means s is a pre-image of one of the user public keys.

The collision resistance of H implies that the first two cases above happen with at most negligible probability. Moreover, it is straightforward to use the third case above to invert f, meaning the third case occurs with negligible probability. This means the extractor must actually succeed with only non-negligible probability, a contradiction. □

5 An Abstraction: Subset-Sum Encoding

Now that we have seen many applications of witness PRFs, we begin our construction. In this section, we give an abstraction of functionality we need from multilinear maps. Our abstraction is called a *subset-sum* encoding. Roughly, a subset sum encoding is a way to encode vectors $\mathbf{t}$ such that (1) the encoding of $\mathbf{t} = \mathbf{A} \cdot \mathbf{w}$ for $\mathbf{w} \in \{0,1\}^n$ is efficiently computable given $\mathbf{w}$ and (2) the encoding of $\mathbf{t} \notin \mathsf{SubSums}(\mathbf{A})$ is indistinguishable from a random string. More formally, a subset-sum encoding is the following:

Definition 5.1. A *subset-sum encoding* is a triple of efficient algorithms $(\mathsf{Gen}, \mathsf{Encode}, \mathsf{Eval})$ where:

- Gen takes as input a security parameter λ and an integer matrix $\mathbf{A} \in \mathbb{Z}^{m \times n}$, and outputs an encoding key sk and an evaluation key ek.
- Encode takes as input the secret key sk and a vector $\mathbf{t} \in \mathbb{Z}^m$, and produces an encoding $\hat{\mathbf{t}} \in \mathcal{Y}$. Encode is deterministic.
- Eval takes as input the encoding key ek and a bit vector $\mathbf{w} \in \{0,1\}^n$, and outputs a value $\hat{\mathbf{t}}$ satisfying $\hat{\mathbf{t}} = \mathsf{Encode}(\mathsf{sk}, \mathbf{t})$ where $\mathbf{t} = \mathbf{A} \cdot \mathbf{w}$.

Security Notions. The security notions we define for subset-sum encodings are very similar to those for witness PRFs. Consider the following experiment $\texttt{EXP}^{\mathbf{A}}_{\mathcal{A}}(b, \lambda)$ between an adversar $\mathcal{A}$ and challenger, parameterized by a matrix $\mathbf{A} \in \mathbb{Z}^{m \times n}$, a bit b, and a security parameters λ:

- Run $(\mathsf{sk}, \mathsf{ek}) \xleftarrow{R} \mathsf{Gen}(\lambda, \mathbf{A})$, and give ek to $\mathcal{A}$
- $\mathcal{A}$ can adaptively make queries on targets $\mathbf{t}_i \in \{0,1\}^m$, to which the challenger responds with $\hat{\mathbf{t}_i} \leftarrow \mathsf{Encode}(\mathsf{sk}, \mathbf{t}_i) \in \mathcal{Y}$.
- $\mathcal{A}$ can make a single challenge query on a target $\mathbf{t}^*$. The challenger computes $y_0 = \hat{\mathbf{t}^*} \leftarrow \mathsf{Encode}(\mathsf{sk}, \mathbf{t}^*)$ and $y_1 \xleftarrow{R} \mathcal{Y}$, and responds with y_b.
- After making additional Encode queries, $\mathcal{A}$ produces a bit b'. The challenger checks that $\mathbf{t}^* \notin \{\mathbf{t}_i\}$ and $\mathbf{t}^* \notin \mathsf{SubSums}(\mathbf{A})$. If either check fails, the challenger outputs a random bit. Otherwise, it outputs b'.

Define W_b as the event the challenger outputs 1 in experiment b. Let $\texttt{SS.Adv}^{\mathbf{A}}_{\mathcal{A}}(\lambda) = |\Pr[W_0] - \Pr[W_1]|$.

Definition 5.2. $(\mathsf{Gen}, \mathsf{Encode}, \mathsf{Eval})$ is *adaptive target interactively secure* for a matrix $\mathbf{A}$ if, for all adversaries $\mathcal{A}$, there is a negligible function negl such that $\texttt{SS.Adv}^{\mathbf{A}}_{\mathcal{A}}(\lambda) < \mathsf{negl}(\lambda)$.

We can also define a weaker notion of *static target* security where $\mathcal{A}$ commits to $\mathbf{t}^*$ before seeing ek or making any Encode queries. Independently, we can also define *non-interactive* security where the adversary is not allowed to make any Encode queries.

Fine-grained security notions. Similar to witness PRFs, it is straight forward to define fine-grained security notions, where security holds relative to a specific instance sampler. We omit the details.

5.1 A simple instantiation from multilinear maps.

We now construct subset-sum encodings from asymmetric multilinear maps.

Construction 5.3. Let Setup be the generation algorithm for an asymmetric multilinear map. We build the following subset-sum encoding:

- $\mathsf{Gen}(\lambda, \mathbf{A})$: on input a matrix $\mathbf{A} \in \mathbb{Z}^{m \times n}$, let $B = \|\mathbf{A}\|_\infty$, and $p_{min} = 2nB + 1$. Run $\mathsf{params} \xleftarrow{R} \mathsf{Setup}(\lambda, n, p_{min})$ to get the description of a multilinear map $e : \mathbb{G}_1 \times \cdots \times \mathbb{G}_n \to \mathbb{G}_T$ on groups of prime order p, together with generators $g_1, \ldots, g_m, g_T$. Choose random $\boldsymbol{\alpha} \in (\mathbb{Z}_p^*)^m$. Denote by $\boldsymbol{\alpha}^{\mathbf{v}}$ the product $\prod_{i \in [m]} \alpha_i^{v_i}$ (since each component of $\boldsymbol{\alpha}$ is non-zero, this operation is well-defined for all integer vectors $\mathbf{v}_i$). Let $V_i = g_i^{\boldsymbol{\alpha}^{\mathbf{v}_i}}$ where $\mathbf{v}_i$ are the columns of $\mathbf{A}$. Publish $\mathsf{ek} = (\mathsf{params}, \{V_i\}_{i \in [n]})$ as the public parameters and $\mathsf{sk} = \boldsymbol{\alpha}$
- $\mathsf{Encode}(\mathsf{sk}, \mathbf{t}) = g_T^{\boldsymbol{\alpha}^{\mathbf{t}}}$, where $\mathbf{t} \in \mathsf{IntRange}(\mathbf{A})$.
- $\mathsf{Eval}(\mathsf{ek}, \mathbf{w})$: define $V_{i,1} = V_i$ and $V_{i,0} = g_i$. Then output $e(V_{1,w_1}, V_{2,w_2}, \ldots, V_{n,w_n})$.

For correctness, observe that $V_{i,w_i} = g_i^{\boldsymbol{\alpha}^{\mathbf{v}_i w_i}}$, and therefore

$$\begin{aligned} e(V_{1,w_1}, V_{2,w_2}, \ldots, V_{n,w_n}) &= e(g_1^{\boldsymbol{\alpha}^{\mathbf{v}_1 w_1}}, \ldots, g_n^{\boldsymbol{\alpha}^{\mathbf{v}_n w_n}}) = g_T^{\boldsymbol{\alpha}^{\sum_{i \in [n]} \mathbf{v}_i w_i}} = g_T^{\boldsymbol{\alpha}^{\mathbf{A} \cdot \mathbf{w}}} \\ &= \mathsf{Encode}(\mathsf{sk}, \mathbf{A} \cdot \mathbf{w}) \end{aligned}$$

Security. We assume the security of our subset-sum encodings, which translates to a new security assumption on multilinear maps, which we call the *(adaptive target interactive) multilinear subset-sum Diffie Hellman assumption.* For completeness, we formally define the assumption as follows. Let $\texttt{EXP}^{\mathbf{A}}_{\mathcal{A}}(b, \lambda)$ be the following experiment between an adversary $\mathcal{A}$ and challenger, parameterized by a matrix $\mathbf{A} \in \mathbb{Z}^{m \times n}$, a bit b, and a security parameter λ:

- Let $B = \|\mathbf{A}\|_\infty$, and $p_{min} = 2nB + 1$. Run $\mathsf{params} \xleftarrow{R} \mathsf{Setup}(\lambda, n, p_{min})$.
- Choose a random $\boldsymbol{\alpha} \in \mathbb{Z}_p^m$, and let $V_i = g_i^{\boldsymbol{\alpha}^{\mathbf{v}_i}}$ where $\mathbf{v}_i$ are the columns of $\mathbf{A}$. Give $(\mathsf{params}, \{V_i\}_{i \in [n]})$ to $\mathcal{A}$.
- $\mathcal{A}$ can make oracle queries on targets $\mathbf{t}_i \in \mathsf{IntRange}(\mathbf{A})$, to which the challenger responds with $g_T^{\boldsymbol{\alpha}^{\mathbf{t}_i}}$.
- $\mathcal{A}$ can make a single challenge query on a target $\mathbf{t}^* \in \mathsf{IntRange}(\mathbf{A})$. The challenger computes $y_0 = g_T^{\boldsymbol{\alpha}^{\mathbf{t}^*}}$ and $y_1 = g_T^r$ for a random $r \xleftarrow{R} \mathbb{Z}_p$, and responds with y_b.
- After making additional Encode queries, $\mathcal{A}$ produces a bit b'. The challenger checks that $\mathbf{t}^* \notin \{t_i\}$ and $t^* \notin \mathsf{SubSums}(\mathbf{A})$. If either check fails, the challenger outputs a random bit. Otherwise, it outputs b'.

Define W_b as the event that the challenger outputs 1 in experiment b. Let $\texttt{SSDH.Adv}^{\mathbf{A}}_{\mathcal{A}}(\lambda) = |\Pr[W_0] - \Pr[W_1]|$.

Definition 5.4. The adaptive target interactive multilinear subset-sum Diffie Hellman (SSDH) assumption holds relative to Setup if, for all adversaries $\mathcal{A}$, there is a negligible function negl such that $\texttt{SSDH.Adv}^{\mathbf{A}}_{\mathcal{A}}(\lambda) < \mathsf{negl}(\lambda)$.

Security of our subset-sum encodings immediately follows from the assumption:

Fact 5.5. *If the adaptive target interactive multilinear SSDH assumptions holds for* Setup*, the Construction 5.3 is an adaptive target interactively secure subset-sum encoding.*

We can also consider fine-grained SSDH assumptions and obtain the corresponding fine-grained security notions:

Fact 5.6. *If the (extracting) interactive/non-interactive subset-sum Diffie-Hellman assumption holds relative to* Setup *for a matrix* $\mathbf{A}$ *and an instance sampler* $\mathcal{D}$*, then* $(\mathsf{Gen}, \mathsf{Encode}, \mathsf{Eval})$ *is (extracting) interactively/non-interactively secure for matrix* $\mathbf{A}$ *and instance sampler* $\mathcal{D}$*.*

Flattening The Encodings We can convert any subset-sum encoding for $m = 1$ into a subsetsum encoding for any m. Let $\mathbf{A} \in \mathbb{Z}^{m \times n}$ and define $B = \|\mathbf{A}\|_\infty$. Then, for any $\mathbf{w} \in \{0,1\}^n$, $\|\mathbf{A} \cdot \mathbf{w}\|_\infty \leq nB$. Therefore, we can let $\mathbf{A}' = (1, nB+1, (nB+1)^2, \ldots, (nB+1)^{m-1}) \cdot \mathbf{A}$ be a single row, and run $\mathsf{Gen}(\lambda, \mathbf{A}')$ to get $(\mathsf{sk}, \mathsf{ek})$. To encode an element $\mathbf{t}$, compute $\mathbf{t}' = (1, nB, (nB)^2, \ldots, (nB)^{m-1}) \cdot \mathbf{t}$, and encode $\mathbf{t}'$. Finally, to evaluate on vector $\mathbf{w}$, simply run $\mathsf{Eval}(\mathsf{ek}, \mathbf{w})$.

Security translates since left-multiplying by $(1, nB, (nB)^2, \ldots, (nB)^{m-1})$ does not introduce any collisions. Therefore, we can always rely on subset-sum encodings, and thus the subset-sum Diffie-Hellman assumption, for $m = 1$. However, we recommend *not* using this conversion for two reasons:

- To prevent the exponent from "wrapping" mod $p-1$, $p-1$ needs to be larger than the maximum L_1-norm of the rows of $\mathbf{A}$. In this conversion, we are multiplying rows by exponential factors, meaning p needs to correspondingly be set much larger.
- In Section 7, we prove the security of our encodings in the generic multilinear map model. Generic security is only guaranteed if $\|\mathbf{A}\|_\infty/p$ is negligible. This means for security, p will have to be substantially larger after applying the conversion.

5.2 Limited Witness PRFs

We note that subset-sum encodings immediately give us witness PRFs for restricted classes. In particular, for a matrix $\mathbf{A}$, a subset-sum encoding is a witness PRF for the language $\mathsf{SubSums}(\mathbf{A})$. The various security notions for subset-sum encodings correspond exactly to the security notions for witness PRFs. In the next section, we show how to extend this to witness PRFs for any NP language.

6 Witness PRFs from Subset-Sum Encodings

We now show how to build witness PRFs from secure subset-sum encodings, completing our construction. Essentially, we provide a reduction from an instance x of any NP language L to subset-sum instance $(\mathbf{A}, \mathbf{t})$, where the matrix $\mathbf{A}$ is determined entirely by the language L, and is independent of x (except for its length). Thus, $\mathsf{SubSums}(\mathbf{A})$ corresponds exactly with L.

Our witness PRF for a language L is then a subset-sum encoding for the corresponding matrix $\mathbf{A}$. The value of the PRF on instance x is the encoding of the corresponding target $\mathbf{t}$. Given a witness w for x, the reduction gives a corresponding subset S of columns of $\mathbf{A}$ that sum to $\mathbf{t}$. This allows anyone with a witness to evaluate the PRF at x.

6.1 Verifying Computation as Subset Sum

We now give our reduction. As a starting point, given a circuit C and strings $\mathbf{x}, \mathbf{y}$, we show how to prove that $C(\mathbf{x}) = \mathbf{y}$ where verification is simply a subset-sum computation. More precisely, we desire the following procedures:

- $\mathsf{Convert}(C)$ takes as input a circuit of g gates mapping in bits to out bits, and outputs a matrix $\mathbf{C} \in \mathbb{Z}^{m \times (out+in+r)}$ and vector $\mathbf{b} \in \mathbb{Z}^m$, plus some parameter params_C, where m, r and $B = \|\mathbf{A}\|_\infty$ are polynomial in g, in, out. We call $\mathbf{C}, \mathbf{b}$ an enforcer for C.
- $\mathsf{Prove}(\mathsf{params}_C, \mathbf{x}, \mathbf{y})$ constructs a proof $\boldsymbol{\pi} \in \{0,1\}^r$ such that

$$\mathbf{C} \cdot \begin{pmatrix} \mathbf{y} \\ \mathbf{x} \\ \boldsymbol{\pi} \end{pmatrix} = \mathbf{b}$$

We require two properties: completeness, which means that Prove always succeeds, and soundness, which says that if $\mathbf{y} \neq C(\mathbf{x})$, then for all proofs $\boldsymbol{\pi} \in \{0,1\}^r$,

$$\mathbf{C} \cdot \begin{pmatrix} \mathbf{y} \\ \mathbf{x} \\ \boldsymbol{\pi} \end{pmatrix} \neq \mathbf{b}$$

Our construction is relatively straightforward. We first describe the scheme for single-gate circuits, and then show how to piece together many gates to form a circuit.

- AND gates: let $\mathbf{C}_{\mathsf{AND}}$ consist of a single row $(-2, 1, 1, -1)$ and $\mathbf{b}_{\mathsf{AND}} = 0$. Prove, on input $\mathbf{x} = (x_1, x2)$ and $y = x_1 \mathsf{AND} x_2 = x_1 x_2$ computes $\pi = x_1 \mathsf{XOR} x_2 = x_1 + x_2 - 2x_1 x_2$. Observe that

$$\mathbf{C}_{\mathsf{AND}} \cdot (y, x_1, x_2, \pi) = -2x_1 x_2 + x_1 + x_2 - (x + 1 + x_2 - 2x_1 x_2) = 0 = \mathbf{b}$$

  For $y = \mathsf{NOT}(x_1 \mathsf{AND} x_2) = 1 - x_1 x_2$ and any $\pi \in \{0, 1\}$,

$$\mathbf{C}_{\mathsf{AND}} \cdot (y, x_1, x_2, \pi) = -2 + 2x_1 x_2 + x_1 + x_2 - \pi = -2 + 4x_1 x_2 + (x_1 \mathsf{XOR} x_2) - \pi$$

  $-2 + 4x_1 x_2 \in \{-2, 2\}$ and $x_1 \mathsf{XOR} x_2 \in \{0, 1\}$, so $-2 + 4x_1 x_2 + (x_1 \mathsf{XOR} x_2) \in \{-2, -1, 2, 3\}$. Therefore, no setting of $\pi \in \{0, 1\}$ will cause $\mathbf{C}_{\mathsf{AND}} \cdot (y, x_1, x_2, \pi)$ to evaluate to 0.

  This AND gate construction is similar to what Groth, Ostrovsky, and Sahai [GOS06] use to build zero knowledge proofs of knowledge for circuit satisfiability.

- OR gates: let $\mathbf{C}_{\mathsf{OR}}$ consist of a single row $(-2, 1, 1, 1)$ and $\mathbf{b}_{\mathsf{OR}} = 0$. Prove, on input $\mathbf{x} = (x_1, x_2)$ and $y = x_1 \mathsf{OR} x_2 = x_1 + x_2 - x_1 x_2$ computes $\pi = x_1 \mathsf{XOR} x_2 = x_1 + x_2 - 2x_1 x_2$. Observe that

$$\mathbf{C}_{\mathsf{OR}} \cdot (y, x_1, x_2, \pi) = -2(x_1 + x_2 - x_1 x_2) + x_1 + x_2 + (x_1 + x_2 - 2x_1 x_2) = 0 = \mathbf{b}$$

  Similarly, for $y = \mathsf{NOT}(x_1 \mathsf{OR} x_2)$, it is straightforward to show that $\mathbf{C}_{\mathsf{OR}} \cdot (y, x_1, x_2, \pi) \neq 0$ for all $\pi \in \{0, 1\}$.

- NOT gates: let $\mathbf{C}_{\mathsf{NOT}}$ consist of a single row $(1 \;\; 1)$ and $\mathbf{b}_{\mathsf{NOT}} = 1$. Prove is trivial and outputs nothing. Verifying the correctness is straightforward.

- PASS gates: a pass gate is a fan-in one gate that just outputs its input. This gate will be useful for our construction, and we give two implementations:

  - The *simple* pass gate. Let $\mathbf{C}_{\mathsf{PASS},0}$ consist of a single row $(1, -1)$ and $\mathbf{b}_{\mathsf{PASS},0} = 0$. Prove is trivial and outputs nothing. Verifying the correctness is straightforward. This is more direct than implementing the pass gate as two not gates.
  - The *cheating* pass gate. Let $\mathbf{C}_{\mathsf{PASS},1}$ consist of a single row $(1, 1, -2)$ and $\mathbf{b}_{\mathsf{PASS},1} = 0$. Prove, on input x and $y = x$ computes $\pi = x$. Correctness is straightforward. Notice that for this gate, if we allow π to be a real number in $[0, 1]$, then we can can set $\pi = 1/2$ and $y = \mathsf{NOT} x$ and "prove" the wrong output. This will be important for our witness PRF construction.

- Other fan-in two gates: it is straightforward to build all fan-in two gates using the above ideas. Each gate requires just a single row in $\mathbf{C}$, and a single proof bit π. We omit the details. The advantage of implementing all fan-in two gates is that we can absorb NOT gates into the fan-in two gates, and therefore get NOT gates for free.

To handle arbitrary circuits, we evaluate the circuit gate by gate. For each gate, we assign a row of $\mathbf{C}$ which will enforce that the output of that row is correct using the above single-gate enforcers. We will also assign at most two columns, one for the actual result, and possibly one column for the proof. For any invalid computation with potential proof $\boldsymbol{\pi}$, $\boldsymbol{\pi}$ will give an assignment to the wires.

Since the result is incorrect, there must be some gate where the input wires are correct, but the output wire is incorrect. Assume the gate has two inputs, the one input gate being similar. For this gate, look at the corresponding row of $\mathbf{C}$, the two input columns, the output column, and the proof column. Also look at the row in $\mathbf{b}$. Also look at the restriction of $\boldsymbol{\pi}$ to these four components. This will correspond to a single gate enforcer, and the invalid proof $\boldsymbol{\pi}$ gives an invalid result, which we know to be impossible.

To reduce an NP language L defined by relation R to subset-sum, run $\mathsf{Convert}(R)$ to obtain a matrix $\mathbf{C}$, vector $\mathbf{b}$, and parameters params. Write $\mathbf{C} = (\mathbf{B}, \mathbf{A})$ so that $\mathbf{C} \cdot (r, \mathbf{x}, \mathbf{w}, \boldsymbol{\pi}) = \mathbf{B} \cdot (r, \mathbf{x}) + \mathbf{A} \cdot (\mathbf{w}, \boldsymbol{\pi})$. We then map an instance x for L to the subset-sum instance $(\mathbf{A}, \mathbf{t})$ where $\mathbf{t} = \mathbf{b} - \mathbf{B} \cdot (1, \mathbf{x})$. If $x \in L$, then there is a $\mathbf{w}$ such that $R(\mathbf{x}, \mathbf{w}) = 1$. In this case, we can run $\mathsf{Prove}(\mathsf{params}, (\mathbf{x}, \mathbf{w}), 1)$ to obtain a proof $\boldsymbol{\pi}$ where $\mathbf{C} \cdot (1, \mathbf{x}, \mathbf{w}, \boldsymbol{\pi}) = \mathbf{b}$. Equivalently,

$$\mathbf{A} \cdot (\mathbf{w}, \boldsymbol{\pi}) = \mathbf{b} - \mathbf{B} \cdot (1, \mathbf{x}) = \mathbf{t}$$

meaning $\mathbf{t} \in \mathsf{SubSums}(\mathbf{A})$. Conversely, if $\mathbf{t} = \mathbf{A} \cdot (\mathbf{w}, \boldsymbol{\pi})$ for some $\mathbf{w}, \boldsymbol{\pi}$ (so that $\mathbf{t} \in \mathsf{SubSums}(\mathbf{A})$), then the above shows that $R(\mathbf{x}, \mathbf{w}) = 1$, meaning that $\mathbf{x} \in L$. This completes the reduction. Observe that the matrix $\mathbf{A}$ depends only on R, and not on the instance $\mathbf{x}$, as desired.

6.2 Our Construction of Witness PRFs

We now give our construction of witness PRFs from subset-sum encodings. Unfortunately, the reduction above is not quite enough for witness PRFs, and some modifications need to be made. We now give the details:

Construction 6.1. Let $(\mathsf{SSE.Gen}, \mathsf{SSE.Encode}, \mathsf{SSE.Eval})$ be a secure subset-sum encoding.

- $\mathsf{WPRF.Gen}(\lambda, R)$: On relation $R : \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}$ consisting of g gates, let $R'(x, w) = (x, R(x, w))$. We can implement R' using the g gates of R, as well as k simple pass gates for the x part of the output. We will also add a cheating pass gate on the output of R. Therefore, we can compute $(\mathbf{C}, \mathbf{b}, \mathsf{params}_R) \leftarrow \mathsf{Convert}(R)$ using $\mathsf{Convert}$ from above. Notice that $\mathbf{C} \in \mathbb{Z}^{(k+g+1)\times(2k+g+\ell+2)}$.

  Now, rearrange the columns of $\mathbf{C}$ so that:

  - $\mathbf{C} = (\mathbf{B}, \mathbf{A})$ where $\mathbf{B} \in \mathbb{Z}^{(k+g+1)\times(k+1)}$ and $\mathbf{A} \in \mathbb{Z}^{(m+g+1)\times(m+g+n+1)}$
  - On input $(\mathbf{x}, \mathbf{w})$ and output $r = R(\mathbf{x}, \mathbf{w})$, Prove produces a vector $\mathbf{v} = (\mathbf{v}_0, \mathbf{v}_1) = ((\mathbf{x}, r), (\mathbf{x}, \mathbf{w}, \boldsymbol{\pi}))$ such that

    $$\mathbf{C} \cdot \mathbf{v} = \mathbf{B} \cdot \mathbf{v}_0 + \mathbf{A} \cdot \mathbf{v}_1 = \mathbf{b}$$

  Now, run and output $(\mathsf{fk}, \mathsf{ek}) = (\mathsf{sk}, \mathsf{ek}) \xleftarrow{R} \mathsf{SSE.Gen}(\lambda, \mathbf{A})$.

- $\mathsf{WPRF.F}(\mathsf{fk}, \mathbf{x})$: Let $\mathbf{t} = \mathbf{b} - \mathbf{B} \cdot (\mathbf{x}, 1)$. Run $\hat{\mathbf{t}} \leftarrow \mathsf{SSE.Encode}(\mathsf{fk}, \mathbf{t})$

- $\mathsf{WPRF.Eval}(\mathsf{ek}, \mathbf{x}, \mathbf{w})$: Let $\mathbf{v}_1 = (\mathbf{x}, \mathbf{w}, \boldsymbol{\pi})$ where $\boldsymbol{\pi}$ is the proof constructed above. Run and output $\hat{\mathbf{t}} \leftarrow \mathsf{SSE.Eval}(\mathsf{ek}, \mathbf{v}_1)$

Note that using $\mathsf{Convert}$ from above, $B \equiv \|\mathbf{A}\|_\infty = 2$.

Remark 6.2. $\mathbf{t} = \mathbf{b} - \mathbf{B}_R \cdot (\mathbf{x}, 1) \in \mathsf{IntRange}(\mathbf{A})$. This is because, on input $\mathbf{x}$ and any witness $\mathbf{w}$, either $R(\mathbf{x}, \mathbf{w}) = 1$, in which case we the proof $\boldsymbol{\pi}$ derived from this instance and witness give a subset sum. If $R(\mathbf{x}, \mathbf{w}) = 0$, we can produce a proof $\boldsymbol{\pi}' \in [0, 1]^{g+1}$ by setting all of the gates correctly, except the last "cheating" pass gate, which we set to 1 by setting $\pi = 1/2$ for that gate.

Remark 6.3. For each instance $\mathbf{x}$, the corresponding target $\mathbf{t} = \mathbf{b} - \mathbf{B}_R \cdot (\mathbf{x}, 1)$ is unique. This is due to our construction, where the only valid proofs $\boldsymbol{\pi}$ for input $\mathbf{x}$ must contain $\mathbf{x}$ (since it is part of the output of R'). Since the proof, and hence the subset, must be different for every $\mathbf{x}$, we have that the target must also be different for every $\mathbf{x}$. This is crucial for security, since mapping two instances to the same target would mean F is not a pseudorandom function.

6.3 Security of Our Construction

We now state the security of our construction. For an NP relation R, let $\mathbf{A}$ be the matrix defined above.

Theorem 6.4. *If* SSE *is an adaptive target interactively secure subset-sum encoding, then* WPRF *in Construction 6.1 is adaptive instance interactively secure.*

This theorem is an immediate consequence of our construction, and the fact that each instance maps to a unique target. We can also consider the fine-grained security notions. For an R-instance sampler $\mathcal{D}_{\mathsf{WPRF}}$ for WPRF, let $\mathcal{D}_{\mathsf{SSE}}$ be the following $\mathbf{A}$-target sampler for SSE: On input ek, simulate $\mathcal{D}_{\mathsf{WPRF}}$ on input ek. Whenever $\mathcal{D}_{\mathsf{WPRF}}$ makes a query to F on input $\mathbf{x}$, compute $\mathbf{t} = \mathbf{b} - \mathbf{B} \cdot (\mathbf{x}, 1)$, and make an encode query on $\mathbf{t}$, obtaining $\hat{\mathbf{t}}$. Return $\hat{\mathbf{t}}$ to $\mathcal{D}_{\mathsf{WPRF}}$. When $\mathcal{D}_{\mathsf{WPRF}}$ produces a witness $\mathbf{x}^*$, compute and output the target $\mathbf{t}^* = \mathbf{b} - \mathbf{B} \cdot (\mathbf{x}, 1)$. Security immediately follows:

Theorem 6.5. *If* SSE *is interactively (resp. non-interactively) secure for* $\mathbf{A}$ *and* $\mathcal{D}_{\mathsf{SSE}}$*, then* WPRF *is interactively (resp. non-interactively) secure for* R *and* $\mathcal{D}_{\mathsf{WPRF}}$*. Moreover, if* SSE *is extractable, then so is* WPRF*.*

6.4 Reducing Key Sizes Using Verifiable Computation

Our witness PRF requires a subset-sum encoding a matrix $\mathbf{A}$ of width proportional to the size of the circuit evaluating R. Our subset-sum encodings are in turn constructed from multilinear maps, and the total multilinearity will be equal to the width of the matrix $\mathbf{A}$. Since multilinearity is very expensive, it is important to reduce the size of the circuit $\mathbf{A}$.

The role of $\mathbf{A}$ is basically to prove that R has the output claimed, and operates by checking every step of the computation. In order to shrink $\mathbf{A}$, we could have Eval first compute $R(x, w)$, and simultaneously compute a proof π that the computation was correct. Then $\mathbf{A}$ does not check the evaluation of $R(x, w)$ directly, but instead checks the evaluation of the program that verifies π. If π and the verification algorithm can be made much smaller than R itself, this will decrease the size of $\mathbf{A}$.

One candidate approach would be to use PCPs. However, this will not work directly, as the randomness used by the verification algorithm would need to be hard-coded into the matrix $\mathbf{A}$, but the proof is generated in Eval, which gets (the ecoding of) $\mathbf{A}$ as input. Therefore, the prover could craft the proof to fool the verifier for the specific random coins used.

Instead, we can use verifiable computation, defined as follows:

Given a circuit R, a verifiable computation scheme consists of:

- $\mathsf{Gen}(\lambda, R)$ which outputs a verification key vk and evaluation key ek.
- $\mathsf{Compute}(\mathsf{ek}, x)$ computes $y = R(x)$ as well as a proof π.
- $\mathsf{Ver}(\mathsf{vk}, x, y, \pi)$ is a deterministic algorithm that outputs 0 or 1.

Given a verifiable computation scheme, we can build the following witness PRF

Construction 6.6. Let $(\mathsf{WPRF.Gen}, \mathsf{F}, \mathsf{Eval})$ be a witness PRF and $(\mathsf{VC.Gen}, \mathsf{Compute}, \mathsf{Ver})$ be a verifiable computation scheme. Build the following:

- $\mathsf{WPRF.Gen}'(\lambda, R')$: run $(\mathsf{vk}, \mathsf{ek}_0) \xleftarrow{R} \mathsf{VC.Gen}(\lambda, R')$. Then define the relation

$$R((\mathsf{vk}, \mathbf{x}), (\mathbf{w}, \pi)) = \mathsf{Ver}(\mathsf{vk}, (\mathbf{x}, \mathbf{w}), 1, \pi)$$

  Run $(\mathsf{fk}, \mathsf{ek}_1) \xleftarrow{R} \mathsf{WPRF.Gen}(\lambda, R)$. Output the secret key fk and public parameters $\mathsf{ek} = (\mathsf{vk}, \mathsf{ek}_0, \mathsf{ek}_1)$.
- $\mathsf{F}'(\mathsf{fk}, \mathbf{x}) = \mathsf{F}(\mathsf{fk}, (\mathsf{vk}, \mathbf{x}))$
- $\mathsf{Eval}'(\mathsf{ek}, \mathbf{x}, \mathbf{w})$: run $\mathsf{Compute}(\mathsf{ek}_0, (\mathbf{x}, \mathbf{w}))$ to obtain $b = R'(x, w)$ and a proof π. If $b = 0$, abort and output $\perp$. Otherwise, run $\mathsf{Eval}(\mathsf{ek}_1, (\mathsf{vk}, \mathbf{x}), (\mathbf{w}, \pi))$.

Correctness is immediate. Now the parameter size ek is the length of the evaluation key ek_0 plus the length of the parameter ek_1 for WPRF. However, the size of R is independent of the size of R', and only depends on the running time of Ver. Using the verifiable computation system of, say, Parno, Howell, Gentry and Raykova [PHGR13], this is linear in $|x|$ and $|w|$. The size of ek_0 however does depend linearly on the size of R. Thus, the total parameter size is depends only linearly on the size of R', rather than polynomially.

Security. Unfortunately, we need to rely on the strong form of extractable security. This is because false proofs do exist, and thus the languages defined by R and R' will not coincide. Since extracting security does not seem to imply non-extracting security, it is unlikely that we can prove F' non-extracting secure.

Given an R'-instance sampler $\mathcal{D}'$ for WPRF', we construct the following R-instance sampler $\mathcal{D}$ for WPRF. On input ek_1, run run $(\mathsf{ek}_0, \mathsf{vk}) \xleftarrow{R} \mathsf{VC.Gen}(\lambda, R')$ and give $\mathsf{ek} = (\mathsf{ek}_0, \mathsf{ek}_1, \mathsf{vk})$ to $\mathcal{D}'$. When $\mathcal{D}'$ makes a F' query on instance $\mathbf{x}$, make a F query on $(\mathsf{vk}, \mathbf{x})$. When $\mathcal{D}'$ outputs an instance $\mathbf{x}^*$, output $(\mathsf{vk}, \mathbf{x}^*)$, along with auxiliary information, namely Aux outputted by $\mathcal{D}'$ and ek_0.

Theorem 6.7. *If* WPRF *is extractable interactively (resp. non-interactively) secure for relation* R *and instance sampler* $\mathcal{D}$*, then* WPRF' *is extractable interactively (resp. non-interactively) secure for instance sampler* $\mathcal{D}'$*.*

Proof. We prove the interactive case, the other case begin similar. Let $\mathcal{A}'$ be an extractable adversary for WPRF' relative to instance sampler $\mathcal{D}'$. We construct the following adversary $\mathcal{A}$ for WPRF relative to $\mathcal{D}$. $\mathcal{A}$, on input $\mathsf{ek}_1, (\mathsf{vk}, \mathbf{x}^*), k, \mathsf{Aux}, \mathsf{ek}_0$, runs $\mathcal{A}'$ on $\mathsf{ek} = (\mathsf{ek}_0, \mathsf{ek}_1, \mathsf{vk}), \mathbf{x}^*, k, \mathsf{Aux}$. When $\mathcal{A}'$ makes a F' query on instance $\mathbf{x}$, answer by making a F query on $(\mathsf{vk}, \mathbf{x})$. When $\mathcal{A}'$ outputs a guess b', $\mathcal{A}$ outputs the same guess.

Suppose $\mathcal{A}'$ has non-negligible advantage $\epsilon = 1/q_{\mathcal{A}'}$. Observe that $\mathcal{A}$ perfectly simulates the view of $\mathcal{A}'$, so $\mathcal{A}$ also has advantage $1/q_{\mathcal{A}'}$. This implies an extractor $\mathcal{E}$ and polynomial $q_{\mathcal{E}}$ such that $\mathcal{E}$ has advantage at least $1/q_{\mathcal{E}}$. We construct the following extractor $\mathcal{E}'$ for WPRF'. On input $(\mathsf{ek}, x^*, \mathsf{Aux}, y^*, \{x_i, y_i\}, r)$, $\mathcal{E}'$ runs $\mathcal{E}$ on input $(\mathsf{ek}_1, (\mathsf{vk}, x^*), (\mathsf{Aux}, \mathsf{ek}_0), y^*, \{(\mathsf{vk}, x_i), y_i\}, r)$. $\mathcal{E}'$ perfectly simulates the view of $\mathcal{E}$, so the output (w, π) will satisfy $R'((\mathsf{vk}, \mathbf{x}^*), (w, \pi)) = 1$ with probability at least $1/q_{\mathcal{E}}$. Output w as a witness for $\mathbf{x}^*$.

Suppose $\mathcal{E}'$ has negligible advantage. This implies that $R(\mathbf{x}^*, w) = 1$ with negligible probability. But then π is an invalid proof, meaning $\mathcal{E}'$ can be used to break the security of VC, a contradiction. Therefore, $\mathcal{E}'$ has non-negligible advantage, say $1/2q_{\mathcal{E}}$.

□

7 Generic Hardness

In this section, we prove the generic hardness of our multilinear subset-sum Diffie-Hellman problem. We prove the hardness of (non-extracting) adaptive witness interactive assumption.

Generic Multilienar Group Model. We prove security in the so-called generic multilinear group model. In this model, rather than having direct access to group elements, the adversary only has access to *âĂćphlabels of group elements. It also has access to an oracle that allows it to multiply elements of the same group, as well as apply the multilinear operation. We allow the adversary to successively pair elements together, rather than only providing the full multilinear map. This reflects the structure of current map candidates.

More precisely, we have a family of groups $\mathbb{G}_{\mathbf{u}}$ where $\mathbf{u} \in \{0,1\}^n$. The target group is $\mathbb{G}_T = \mathbb{G}_{1^n}$, and $\mathbb{G}_i = \mathbb{G}_{\mathbf{e}_i}$, where $\mathbf{e}_i$ is the ith unit vector. We represent the groups using a function $\xi : \mathbb{Z}_p \times \{0,1\}^n \to \{0,1\}^m$, which maps elements of the ring $\mathbb{Z}_p$ (along with a group index $\mathbf{u} \in \{0,1\}^n$) into bit strings of length m. We provide the adversary with oracles Mult and Pair to compute the induced group and pairing operations:

- $\mathsf{Encode}(x, \mathbf{u})$ returns $\xi(x, \mathbf{u})$. Note that to compute a generator for a group $\mathbb{G}_{\mathbf{u}}$ as $\mathsf{Encode}(1, \mathbf{u})$
- $\mathsf{Mult}(\xi_1, \xi_2, b)$ If $\xi_1 = \xi(x_1, \mathbf{u})$ and $\xi_2 = \xi(x_2, \mathbf{u})$ for the same index $\mathbf{v}$, then return $\xi(x_1 + (-1)^b x_2, \mathbf{u})$. Otherwise output $\perp$.
- $\mathsf{Pair}(\xi_1, \xi_2)$ if $\xi_1 = \xi(x_1, \mathbf{u}_1)$ and $\xi_2 = \xi(x_2, \mathbf{u}_2)$ where $\mathbf{u}_1 + \mathbf{u}_2 \leq 1^n$ (that is, the component-wise sum over the integers has all entries less than or equal to 1), then output $\xi(x_1 x_2, \mathbf{u}_1 + \mathbf{u}_2)$. Otherwise output $\perp$.

The following theorem shows the hardness of the general multilinear subset-sum Diffie-Hellman problem:

Theorem 7.1. *Let $\mathbf{A} \in \mathbb{Z}^{m \times n}$ be a matrix with entries bounded by B. Assume B/p is negligible, where p is the group order. Then for any adaptive instance interactive adversary $\mathcal{A}$ for the multilinear subset-sum Diffie-Hellman problem, $\mathcal{A}$ has negligible advantage.*

Now we prove Theorem 7.1:

Proof. Fix a matrix $\mathbf{A} \in \mathbb{Z}^{m \times n}$ with entries bounded by B, and adversary $\mathcal{A}$. We will set m arbitrarily high so that with overwhelming probability, ξ is injective and moreover the adversary cannot guess any representations, but must instead make queries to $\mathsf{Encode}, \mathsf{Mult}, \mathsf{Pair}$

Consider the execution of the experiment on $\mathcal{A}$. A random vector $\boldsymbol{\alpha} \xleftarrow{R} (\mathbb{Z}_p^*)^m$ is chosen at random, and the labels $V_i = \xi(\boldsymbol{\alpha}^{\mathbf{v}_i}, \mathbf{e}_i)$ for $i \in [n]$ are given to $\mathcal{A}$. $\mathcal{A}$ is allowed to make queries on vectors $\mathbf{t} \in \mathsf{IntRange}(\mathbf{A})$, to which we respond with $E_{\mathbf{t}} = \xi(\boldsymbol{\alpha}^{\mathbf{t}}, 1^n)$. $\mathcal{A}$ can also make a polynomial number of queries to $\mathsf{Encode}, \mathsf{Mult}, \mathsf{Pair}$ to perform group pairing operations. At some point, $\mathcal{A}$ produces a target $\mathbf{t}^*$ that was not among the queries made so far. Set $y_0 = E_{\mathbf{t}^*} = \xi(\boldsymbol{\alpha}^{\mathbf{t}^*}, 1^n)$. We also choose a random β and set $y_1 = \xi(\beta, 1^n)$. $\mathcal{A}$ is given y_b in response. $\mathcal{A}$ is allowed to make additional queries on $\mathbf{t} \neq \mathbf{t}^*$ to get values $E_{\mathbf{t}}$. Finally, $\mathcal{A}$ produces a guess b' for b. $\mathcal{A}$ has advantage $|\Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0]| = \epsilon$. Our goal is to show that ϵ is negligible.

Now let $\mathcal{B}$ be an (inefficient) algorithm that plays the above game with $\mathcal{A}$. Rather than choose $\boldsymbol{\alpha}, y_0, y_1$, $\mathcal{B}$ treats them as formal variables. $\mathcal{B}$ maintains a list $L = \{(p_j, \mathbf{u}_j, \xi_j)\}$ where p_j is a polynomial in $\boldsymbol{\alpha}, y_0, y_1$, $\mathbf{u}_j \in \{0,1\}^m$ indexes the groups, and ξ_j is a string in $\{0,1\}^m$. Note that the exponent of α_i in any polynomial may be negative. The list is initialized with the tuples $(\boldsymbol{\alpha}^{\mathbf{v}_i}, \mathbf{e}_i, \xi_{i,1})$ and $(1, \mathbf{e}_i, \xi_{i,0})$ for $i \in [n]$, where $\xi_{i,b}$ are randomly generated strings in $\{0,1\}^m$.

The game starts with $\mathcal{B}$ giving $\mathcal{A}$ the tuple of strings $\{\xi_{i,b}\}_{i \in [n], b \in \{0,1\}}$. Now $\mathcal{A}$ is allowed to make the following queries:

$\mathsf{Encode}(r, \mathbf{u})$: If $r \in \mathbb{Z}_p$ and $\mathbf{u} \in \{0,1\}^n$, then $\mathcal{B}$ looks for a tuple $(p, \mathbf{u}, \xi) \in L$, where p is the constant polynomial equal to r. If such a tuple exists, then $\mathcal{B}$ responds with ξ. Otherwise, $\mathcal{B}$ generates a random string $\xi \in \{0,1\}^m$, adds the tuple $(\pi, \mathbf{u}, \xi)$ to L, and responds with ξ.

$\mathsf{Mult}(\xi_k, \xi_\ell, b)$: $\mathcal{B}$ looks for tuples $(p_k, \mathbf{u}_k, \xi_k), (p_\ell, \mathbf{u}_\ell, \xi_e ll) \in L$. If one of the tuples is not found, $\mathcal{B}$ responds with $\perp$. If both are found, but $\mathbf{u}_k \neq \mathbf{u}_\ell$, then $\mathcal{B}$ responds with $\perp$. Otherwise, $\mathcal{B}$ lets $\mathbf{u} \equiv \mathbf{u}_k = \mathbf{u}_\ell$, and computes the polynomial $p = p_k + (-1)^b p_\ell$. Then $\mathcal{B}$ looks for the tuples $(p, \mathbf{u}, \xi) \in L$, and if the tuple is found $\mathcal{B}$ responds with ξ. Otherwise, $\mathcal{B}$ generates a random string ξ, adds $(p, \mathbf{u}, \xi)$ to L, and responds with ξ.

$\mathsf{Pair}(\xi_k, \xi_\ell)$: $\mathcal{B}$ looks for tuples $(p_k, \mathbf{u}_k, \xi_k), (p_\ell, \mathbf{u}_\ell, \xi_\ell) \in L$. If one of the tuples is not found, $\mathcal{B}$ responds with $\perp$. If both are found, but $\mathbf{u}_k \wedge \mathbf{u}_\ell \neq 0^n$ (in other words, $\mathbf{u}_k$ and $\mathbf{u}_\ell$ have a 1 in the same locaiton), then $\mathcal{B}$ also responds with $\perp$. Otherwise, let $\mathbf{u} = \mathbf{u}_k + \mathbf{u}_\ell$ (addition over $\mathbb{Z}$), and let $p = p_k \cdot p_\ell$. $\mathcal{B}$ looks for $(p, \mathbf{u}, \xi) \in L$, and if found, responds with ξ. Otherwise, $\mathcal{B}$ generates a random $\xi \in \{0,1\}^m$, adds $(p, \mathbf{u}, \xi)$ to L, and responds with ξ.

$O(\mathbf{t})$: $\mathcal{B}$ looks for a tuple $(\boldsymbol{\alpha}^{\mathbf{t}}, 1^n, \xi) \in L$, and responds with ξ if the tuple is found. Otherwise, $\mathcal{B}$ generates a random $\xi \in \{0,1\}^m$, adds $(\boldsymbol{\alpha}^{\mathbf{t}}, 1^n, \xi)$ to L, and responds with ξ. Let Q be the set of queries to O.

$Challenge(\mathbf{t}^*)$: $\mathcal{B}$ (inefficiently) tests if $\mathbf{t}^* \in \mathsf{SubSums}(\mathbf{A})$. If so, $\mathcal{B}$ aborts the simulation. Otherwise, $\mathcal{B}$ creates a new formal variable y, and adds the following tuple to L: $(y, 1^n, \xi)$. Then $\mathcal{B}$ gives to $\mathcal{A}$ the value ξ.

$\mathcal{A}$ ultimately produces a guess b'. Now, $\mathcal{B}$ chooses a random $b \in \{0,1\}$, as well as values for $\boldsymbol{\alpha} \in (\mathbb{Z}_p^*)^m$. If $b = 0$, $\mathcal{B}$ sets $y = \alpha^{\mathbf{t}^*}$, and otherwise, $\mathcal{B}$ chooses a random $y \in \mathbb{Z}_p$.

The simulation provided by $\mathcal{B}$ is perfect unless the choice for the variables $\boldsymbol{\alpha}, y$ results in an equality in the values of two polynomials p_k, p_ℓ in L that is not an equality for polynomials. More precisely, the simulation is perfect unless for some k, ℓ the following hold:

- $\mathbf{u}_k = \mathbf{u}_\ell$
- $p_k(\boldsymbol{\alpha}, y_0, y_1) = p_\ell(\boldsymbol{\alpha}, y_0, y_1)$, yet the polynomials p_k, p_ℓ are not equal.

Let Fail be the event that these conditions hold for some k, ℓ. Our goal is to bound the probability Fail occurs. First, in the $b = 0$ case, consider setting $y = \boldsymbol{\alpha}^{\mathbf{t}^*}$ as polynomials *before* assigning values to $\boldsymbol{\alpha}$. We claim that this does not create any new polynomial equalities. Suppose towards contradiction that this setting causes $p_k = p_\ell$ for some k, ℓ where equality did not hold previously. Then $p_k - p_\ell = 0$. Consider expanding $p_k - p_\ell$ out into monomials prior to the substitution. First, this expansion must contain a y term, and this term cannot have been multiplied by other variables (since polynomials involving y can only exist in the group $\mathbb{G}_{1^n}$). The expansion may also contain $\boldsymbol{\alpha}^{\mathbf{t}}$ terms for all $\mathbf{t} \in Q$, also not multiplied by any other variable (since $\mathbf{t} \in Q$ were only provided in the group $\mathbb{G}_{1^n}$). All other terms came from multiplying and pairing the V_i and g_i together. In particular, each remaining term must come from pairing a subset of the V_i together with the complementing subset of the g_i. Therefore, we can write $p_k - p_\ell$ as

$$p_k - p_\ell = C^* y + \sum_{\mathbf{t} \in Q} C_{\mathbf{t}} \boldsymbol{\alpha}^{\mathbf{t}} + \sum_{\mathbf{t} \in \mathsf{SubSums}(\mathbf{A})} D_{\mathbf{t}} \boldsymbol{\alpha}^{\mathbf{t}}$$

Recall that $\alpha^(p-1) = 1 \mod p$ for all $\alpha \in \mathbb{Z}_p \setminus \{0\}$. This means we should only consider monomials with exponents reduced mod $p-1$ to the range $(-(p-1)/2, (p-1)/2]$. Since all $\mathbf{t} \in Q$ are required to satisfy $\mathbf{t} \in \mathsf{IntRange}(\mathbf{A})$, meaning $\|\mathbf{t}\|_\infty \leq n\|\mathbf{A}\|_\infty$, and $p > 2n\|\mathbf{A}\|_\infty$, all of the exponents in $\boldsymbol{\alpha}^{\mathbf{t}}$ are already reduced. The same applies to $\boldsymbol{\alpha}^{\mathbf{t}^*}$. Since $\mathbf{t}^* \notin Q$ and $\mathbf{t}^* \notin \mathsf{SubSums}(\mathbf{A})$, the monomial $\boldsymbol{\alpha}^{\mathbf{t}^*}$ did not exist prior to substitution. Therefore substituting y with $\boldsymbol{\alpha}^{\mathbf{t}^*}$ cannot make the polynomial zero.

Now, notice that all of the V_i are monomials where each α_i has exponent at most B and at least $-B$. This means the exponent of α_i lies in the range $[-nB, nB]$ for any monomial in the expansion of $p_k - p_\ell$. Consider $p = \boldsymbol{\alpha}^{nB}(p_k - p_\ell)$ (where the exponentiation applies to each of the components of $\boldsymbol{\alpha}$). p is then a proper polynomial where all coefficients are non-negative, and the total degree is at most $2mnB$. Moreover, $p = 0$ as a polynomial if and only if $p_k = p_\ell$ as polynomials. Lastly, the only zeros of p are when $\alpha_i = 0$ for some i, or $p_k - p_\ell = 0$. Since $\boldsymbol{\alpha}$ is chosen to have only non-zero components, this leaves only the zeros of $p_k - p_\ell$. The Swartz-Zippel lemma then shows that if $p \neq 0$, the probability that the polynomial evaluates to zero is at most $2mnB/(p-1)$. This means that p_k and p_ℓ evaluate to the same value with probability at most $2mnB/(p-1)$.

Let q_e, q_m, q_p, q_o be the number of encode, multiply, pair, and O queries made by $\mathcal{A}$. Then the total length of L is at most $q_e + q_m + q_p + q_o + n + 1$. Therefore, the number of pairs is at most $(q_e + q_m + q_p + q_o + n + 1)^2/2$, and so Fail happens with probability at most $mnB(q_e + q_m + q_p + q_o + n + 1)^2/2(p-1)$. If Fail does not occur, $\mathcal{B}$'s simulation is perfect, and in this case b is independent from $\mathcal{A}$'s view since b was chosen after the simulation. It is straightforward to show that $\mathcal{A}$'s advantage is then at most $mnB(q_e + q_m + q_p + q_o + n + 2)^2/2(p-1)$. For any polynomial number of queries, this is negligible provided B/p is negligible, as desired.

□

7.1 Multi-Relation Witness PRFs

In this section, we define the notion of multi-relation witness PRFs, which allows the use of multiple relations, similar to how constrained PRFs allow multiple constrained keys. We note that we do not

actually need this notion of witness PRFs for any of our applications, but instead define this as a natural generalization worth exploring that may be useful for other applications.

Definition 7.2. A multi-relation witness PRF is a triple of algorithms $(\mathsf{Gen}, \mathsf{F}, \mathsf{Eval})$ such that:

- Gen is a randomized algorithm that takes as input a security parameter λ and a bound s on the size of possible relations, and produces a secret function key fk.
- F is a deterministic algorithm that takes as input the function key fk and an input $x \in \mathcal{X}$, and produces some output $y \in \mathcal{Y}$ for some set $\mathcal{Y}$.
- KeyGen is a possibly randomized algorithm that takes as input the function key fk and a relation R where $|R| \leq s$. It outputs an evaluation key ek_R
- Eval is a deterministic algorithm that takes as input the evaluation key ek_R for relation R, and input $x \in \mathcal{X}$, and a witness $w \in \mathcal{W}$, and produces an output $y \in \mathcal{Y}$ or $\perp$.
- For correctness, we require $\mathsf{Eval}(\mathsf{ek}_R, x, w) = \begin{cases} \mathsf{F}(\mathsf{fk}, x) & \text{if } R(x,w) = 1 \\ \perp & \text{if } R(x,w) = 0 \end{cases}$ for all $x \in \mathcal{X}, w \in \mathcal{W}$.

Defining security for multi-relation witness PRFs is straightforward. The relation is no longer a parameter for the security game, but is adaptively chosen by the adversary; moreover, the adversary can request evaluation keys for many relations. On the challenge query, the challenger checks that the instance has no witness relative to any of the queried relations. Moreover, on queries after the challenge, the challenger checks that the challenge instance has no witness relative to that relation. We note that PRF queries are now redundant: a PRF query can be answered by making a relation query on a point function that outputs 1 only on the instance of interest. We can also define a static variant, where the adversary makes the challenge query before any other query.

We can similarly define fine-grained security notions where we define an *instance sampler* $\mathcal{D}$ as a randomized algorithm that possibly makes relation queries, and outputs a challenge instance, along with auxiliary information. We then think of an adversary $\mathcal{A}$ as picking up where $\mathcal{D}$ left off, trying to distinguish, but having no control over the witness. A witness PRF is secure for instance sampler $\mathcal{D}$ if all efficient adversaries have negligible advantage.

Lastly, it is straightforward to define extracting security with respect to an instance sample $\mathcal{D}$, where we allow the instance to have witnesses relative to queries relations, but require that any adversary that can distinguish must be able to compute a witness relative to one of the relations.

7.2 Candidate Constructions

We will not need multi-relation witness PRFs for this work. However, there may be applications not considered here where such witness PRFs are useful. Therefore, we now give two candidate constructions of multi-relation witness PRFs from standard (single-relation) witness PRFs.

A generic construction. The idea behind our first construction is that the evaluation key ek_R for a relation R will consist of a signature on R. A witness for x then consists of a witness w for x relative to the relation R, and a signature on R. Unfortunately, we do not know how to prove the security of the resulting construction, but we conjecture that it is secure when instantiated with our single-relation witness PRF.

Construction 7.3. Let $\mathsf{Gen}', \mathsf{F}', \mathsf{Eval}'$ be a single-relation witness PRF, and $\mathsf{Gen}_{sig}, \mathsf{Sign}_{sig}, \mathsf{Ver}_{sig}$ be a signature scheme. We construct a multi-relation witness PRF:

- $\mathsf{Gen}(\lambda, s)$. Run $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}_{sig}(\lambda)$. Construct the relation

$$U_{\mathsf{vk}}(x, (R, \sigma, w)) = \mathsf{Ver}_{\mathsf{Sig}}(\mathsf{vk}, R, \sigma) \wedge R(x, w)$$

  In other words, witnesses for x are tuples R, σ, w where σ is a valid signature on R, and w is a witness for x relative to R. Run $(\mathsf{fk}', \mathsf{ek}') \leftarrow \mathsf{Gen}'(\lambda, U_{\mathsf{vk}})$. Output the secret function key $\mathsf{fk} = (\mathsf{fk}', \mathsf{ek}', \mathsf{sk})$

- $\mathsf{F}(\mathsf{fk}, x) = \mathsf{F}'(\mathsf{fk}', x)$

- $\mathsf{KeyGen}(\mathsf{fk}, R)$ runs $\sigma_R \leftarrow \mathsf{Sign}(\mathsf{sk}, R)$ and outputs $\mathsf{ek}_R = (\mathsf{ek}', \sigma_R)$

- $\mathsf{Eval}(\mathsf{ek}_R, x, w) = \mathsf{Eval}'(\mathsf{ek}', x, (R, \sigma_R, w))$.

The above construction seems to be secure, since the only way to use Eval to compute F at x is to supply it with a signature on some relation R (which are unforgeable) and a witness relative to R. Therefore, security holds in a model where $\mathsf{Eval}(\mathsf{ek}_R, \cdot, \cdot)$ is given as a black box. Unfortunately, this intuition does not seem to translate into a security proof. Signatures exist on all relations, in particular relations that accept all inputs. The obvious approaches to proving security would seem to require the extracting notion of security for the underlying single-relation witness PRF. Unfortunately, using extracting security does not appear to be enough, as the reduction needs to be able to sign any relation the adversary queries on, but clearly should not be able to sign any relation: in particular, after the challenge is made, the reduction should not be able to sign relations for which the challenge instance has an easy witness. Unfortunately, deciding if the challenge has an easy witness for a relation is not possible with polynomial-size circuits, so approaches using functional signatures [BGI14, BCP14] do not seem to work. We therefore conjecture that Construction 7.3, when combined with our construction of (signle-relation) witness PRFs, is a secure multi-relation witness PRF.

A second construction. We also sketch how to modify our witness PRF construction directly to obtain a multi-relation scheme. The idea is that we instantiate our scheme with the universal relation $U(x, (R, w)) = R(x, w)$. Of course, this witness PRF trivially allows anyone to evaluate the PRF at any point x, using the witness $(R_x, 0)$ where R_x is the relation that accepts only x.

Opening up the construction, we see that to evaluate F on x using a witness (R, w), certain elements are paired together. Let k be the size of the input, ℓ the size of the description of the relation, m the size of the witness. These elements can be organized into four categories:

- Elements $V_{i,b}$ for $i \in [\ell]$, where $V_{i,1} = g_i^{\boldsymbol{\alpha}^{\mathbf{v}_i}}$ and $V_{i,0} = g_i$, which are selected using the bits of the (description of the) relation R.

- Elements $W_{i,b}$ for $i \in [k]$, where $W_{i,1} = g_{\ell+i}^{\boldsymbol{\alpha}^{\mathbf{v}_{\ell+i}}}$ and $W_{i,0} = g_{\ell+i}$, which are selected using the bits of the instance x.

- Elements $X_{i,b}$ for $i \in [m]$, where $X_{i,1} = g_{k+\ell+i}^{\boldsymbol{\alpha}^{\mathbf{v}_{k+\ell+i}}}$ and $X_{i,0} = g_{k+\ell+i}$, which are selected using the bits of w.

- Elements $Y_{i,b}$ for $i \in [r]$, where $Y_{i,1} = g_{k+\ell+m+i}^{\boldsymbol{\alpha}^{\mathbf{v}}{}_{k+\ell+m+i}}$ and $X_{i,0} = g_{k+\ell+m+i}$, which are selected based on the wire values in the computation of $U(x, (R, w))$.

In other words, to evaluate on instance x, relation R, and witness w, let $y \in \{0,1\}^r$ be derived from the wire values in the computation of $U(x, (R, w))$. Then compute

$$e(V_{1,R_1}, \ldots, V_{\ell,R_\ell}, W_{1,x_1}, \ldots, W_{k,x_k}, X_{1,w_1}, \ldots, X_{m,w_m}, Y_{1,y_1}, \ldots, Y_{r,y_r})$$

where R_i are the bits of the description of R.

Now we modify the scheme as follows. We assume that the multilinear map supports a "partial pairing" operation. This means, if the total multilinearity is n, there are groups $\mathbb{G}_S$ for any subset $S \subseteq [n]$ with generators g_S. For any two disjoint sets $S_0, S_1 \subseteq [n]$, there is a pairing operation $e_{S_0,S_1} : \mathbb{G}_{S_0} \times \mathbb{G}_{S_1} \to \mathbb{G}_{S_0 \cup S_1}$ where $e_{S_0,S_1}(g_{S_0}^a, g_{S_1}^b) = g_{S_0 \cup S_1}^{ab}$. By repeated application, we can extend this pairing to act on any number of groups corresponding to disjoint subsets. When clear from context, we will omit the subscripts on e. All known multilinear map instantiations allow for this find-grained structure.

In our modified scheme, the secret function key fk will consist of the $V_{i,b}, W_{i,b}, X_{i,b}$, and $Y_{i,b}$. The evaluation key ek_R for relation R will consist of the $W_{i,b}, X_{i,b}$, and $Y_{i,b}$. This part will be the same for all R. ek_R will also contain the group element $V_R = e(V_{1,R_1}, \ldots, V_{\ell,R_\ell}) \in \mathbb{G}_{[\ell]}$.

Using the secret function key, we can compute F on any input x. Let $R(x', w)$ accept if $x = x'$ and reject otherwise. Let y be the string derived from the wire values in the computation of $U(x, (R, 0^m))$. Then compute

$$\mathsf{F}(\mathsf{fk}, x) = e(V_{1,R_1}, \ldots, V_{\ell,R_\ell}, W_{1,x_1}, \ldots, W_{k,x_k}, X_{1,0}, \ldots, X_{m,0}, Y_{1,y_1}, \ldots, Y_{r,y_r})$$

Using the evaluation key ek_R for relation R, and a witness w such that $R(x, w)$ accepts, we can compute $\mathsf{F}(\mathsf{fk}, x)$ as follows. Let y be the string derived from the wire values in the computation of $U(x, (R, w))$. Compute $W_x = e(W_{1,x_1}, \ldots, W_{k,x_k})$, $X_w = e(X_{1,w_1}, \ldots, X_{m,w_n})$, and $Y_y = e(Y_{1,y_1}, \ldots, Y_{r,y_r})$.

Then compute $e(V_R, W_x, X_w, Y_y)$, where V_R comes from the evaluation key ek_R. This is equivalent to using the witness (R, w) for x in the un-modified scheme, and it therefore follows that this quantity is equal to $\mathsf{F}(\mathsf{fk}, x)$, as desired.

Security. It is straightforward to argue security in the generic multilinear map model. At a high level, the only way to compute a target-group encoding given many evaluation keys is to choose one V_R for a relation R, choose a subset of the $W_{i,b}$, $X_{i,b}$, and $Y_{i,b}$, and then pair them all together. If, for any given R, there is no witness w such that $R(x, w) = 1$, the analysis of Section 7 easily extends to show that there is no subset of the $W_{i,b}, X_{i,b}, Y_{i,b}$ that can be paired with V_R to give $\mathsf{F}(\mathsf{fk}, x)$. If there is no witness relative to any of the R that the adversary has a secret key for, it follows that $\mathsf{F}(\mathsf{fk}, x)$ is hard to compute, even given the V_R. Pseudorandomness follows from a similar argument.

Acknowledgments

This work is supported by DARPA. Opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA.

References

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013. http://eprint.iacr.org/2013/689.

[AGIS14] Prabhanjan Vijendra Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding barrington's theorem. pages 646–658, 2014.

[App13] Benny Applebaum. Bootstrapping obfuscators via fast pseudorandom functions. Cryptology ePrint Archive, Report 2013/699, 2013. http://eprint.iacr.org/2013/699.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. pages 52–73, 2014.

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

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. pages 501–519, 2014.

[BGK+14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. pages 221–238, 2014.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. pages 258–275, 2005.

[BH13] Mihir Bellare and Viet Tung Hoang. Adaptive witness encryption and asymmetric password-based cryptography. Cryptology ePrint Archive, Report 2013/704, 2013. http://eprint.iacr.org/2013/704.

[Bla79] G. R. Blakley. Safeguarding cryptographic keys. *Proceedings of AFIPS 1979 National Computer Conference*, 48:313–317, 1979.

[BLR+14] Dan Boneh, Kevin Lewi, Mariana Raykova, Amit Sahai, Mark Zhandry, and Joe Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. Cryptology ePrint Archive, Report 2014/834, 2014. http://eprint.iacr.org/2014/834.

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. pages 1–25, 2014.

[BS02] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. Cryptology ePrint Archive, Report 2002/080, 2002. http://eprint.iacr.org/2002/080.

[BST14] Mihir Bellare, Igors Stepanovs, and Stefano Tessaro. Poly-many hardcore bits for any one-way function and a framework for differing-inputs obfuscation. pages 102–121, 2014.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. pages 280–300, 2013.

[BWZ14a] Dan Boneh, Brent Waters, and Mark Zhandry. Low overhead broadcast encryption from multilinear maps. pages 206–223, 2014.

[BWZ14b] Dan Boneh, David J. Wu, and Joe Zimmerman. Immunizing multilinear maps against zeroizing attacks. Cryptology ePrint Archive, Report 2014/930, 2014. http://eprint.iacr.org/2014/930.

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

[CGH01] Dario Catalano, Rosario Gennaro, and Nick Howgrave-Graham. The bit security of Paillier's encryption scheme and its applications. pages 229–243, 2001.

[CHL+14] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. Cryptology ePrint Archive, Report 2014/906, 2014. http://eprint.iacr.org/2014/906.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. pages 476–493, 2013.

[CLT14] Jean-Sebastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Cryptanalysis of two candidate fixes of multilinear maps over the integers. Cryptology ePrint Archive, Report 2014/975, 2014. http://eprint.iacr.org/2014/975.

[CLT15] Jean-Sebastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. Cryptology ePrint Archive, Report 2015/162, 2015. http://eprint.iacr.org/.

[CRV10] Ran Canetti, Guy N. Rothblum, and Mayank Varia. Obfuscation of hyperplane membership. pages 72–89, 2010.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. pages 45–64, 2002.

[CZ14] Yu Chen and Zongyang Zhang. Publicly evaluable pseudorandom functions and their applications. Cryptology ePrint Archive, Report 2014/306, 2014. http://eprint.iacr.org/2014/306.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. pages 1–17, 2013.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. pages 40–49, 2013.

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. pages 74–94, 2014.

[GGHW14] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. pages 518–535, 2014.

[GGHZ14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Fully secure functional encryption without obfuscation. Cryptology ePrint Archive, Report 2014/666, 2014. http://eprint.iacr.org/2014/666.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. pages 467–476, 2013.

[GHMS14] Craig Gentry, Shai Halevi, Hemanta K. Maji, and Amit Sahai. Zeroizing without zeroes: Cryptanalyzing multilinear maps without encodings of zero. Cryptology ePrint Archive, Report 2014/929, 2014. http://eprint.iacr.org/2014/929.

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. pages 536–553, 2013.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. pages 25–32, 1989.

[GLSW14] Craig Gentry, Allison Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. Cryptology ePrint Archive, Report 2014/309, 2014. http://eprint.iacr.org/2014/309.

[GLW14] Craig Gentry, Allison B. Lewko, and Brent Waters. Witness encryption from instance independent assumptions. pages 426–443, 2014.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. pages 339–358, 2006.

[GW09] Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). pages 171–188, 2009.

[HSW14] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. pages 201–220, 2014.

[Jou04] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. 17(4):263–276, September 2004.

[KNY14] Ilan Komargodski, Moni Naor, and Eylon Yogev. Secret-sharing for NP. pages 254–273, 2014.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. pages 669–684, 2013.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. pages 238–252, 2013.

[PPS13] Omkant Pandey, Manoj Prabhakaran, and Amit Sahai. Obfuscation-based non-black-box simulation and four message concurrent zero knowledge for NP. Cryptology ePrint Archive, Report 2013/754, 2013. http://eprint.iacr.org/2013/754.

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. pages 500–517, 2014.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.

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

[SZ14] Amit Sahai and Mark Zhandry. Obfuscating low-rank matrix branching programs. Cryptology ePrint Archive, Report 2014/773, 2014. http://eprint.iacr.org/2014/773.

[Zha14] Mark Zhandry. Adaptively secure broadcast encryption with small system parameters. Cryptology ePrint Archive, Report 2014/757, 2014. http://eprint.iacr.org/2014/757.

[Zim14] Joe Zimmerman. How to obfuscate programs directly. Cryptology ePrint Archive, Report 2014/776, 2014. http://eprint.iacr.org/2014/776.