Spooky Encryption and its Applications

Yevgeniy Dodis
NYU

Shai Halevi
IBM Research

Ron D. Rothblum
MIT

Daniel Wichs
Northeastern University

March 10, 2016

Abstract

Consider a setting where inputs $x_1, \ldots, x_n$ are encrypted under independent public keys. Given the ciphertexts $\{c_i = \mathsf{Enc}_{\mathsf{pk}_i}(x_i)\}_i$, Alice outputs ciphertexts $c'_1, \ldots, c'_n$ that decrypt to $y_1, \ldots, y_n$ respectively. What relationships between the x_i's and y_i's can Alice induce?

Motivated by applications to delegating computations, Dwork, Langberg, Naor, Nissim and Reingold [DLN+04] showed that a semantically secure scheme disallows *signaling* in this setting, meaning that y_i cannot depend on x_j for $j \neq i$. On the other hand if the scheme is homomorphic then any *local* (component-wise) relationship is achievable, meaning that each y_i can be an arbitrary function of x_i. However, there are also relationships which are neither signaling nor local. Dwork et al. asked if it is possible to have encryption schemes that support such "spooky" relationships. Answering this question is the focus of our work.

Our first result shows that, under the LWE assumption, there exist encryption schemes supporting a large class of "spooky" relationships, which we call *additive function sharing* (AFS) spooky. In particular, for any polynomial-time function f, Alice can ensure that $y_1, \ldots, y_n$ are random subject to $\sum_{i=1}^n y_i = f(x_1, \ldots, x_n)$. For this result, the public keys all depend on common public randomness. Our second result shows that, assuming sub-exponentially hard indistinguishability obfuscation (iO) (and additional more standard assumptions), we can remove the common randomness and choose the public keys completely independently. Furthermore, in the case of $n = 2$ inputs, we get a scheme that supports an even larger class of spooky relationships.

We discuss several implications of AFS-spooky encryption. Firstly, it gives a strong counterexample to a method proposed by Aiello et al. [ABOR00] for building arguments for NP from homomorphic encryption. Secondly, it gives a simple 2-round multi-party computation protocol where, at the end of the first round, the parties can locally compute an additive secret sharing of the output. Lastly, it immediately yields a function secret sharing (FSS) scheme for all functions.

We also define a notion of *spooky-free* encryption, which ensures that no spooky relationship is achievable. We show that any non-malleable encryption scheme is spooky-free. Furthermore, we can construct spooky-free *homomorphic* encryption schemes from SNARKs, and it remains an open problem whether it is possible to do so from falsifiable assumptions.

Contents

1 Introduction

Imagine Alice and Bob, standing on different planets light years apart. They are "simultaneously" given some input bits x_1 and x_2 respectively, and must answer by outputting bits y_1 and y_2 respectively. Classical physics allows them to implement *local* (component-wise) strategies where y_1 is an arbitrary function of x_1 and y_2 is a function of x_2. On the other hand, the impossibility of faster-than-light communication disallows *signaling* strategies, meaning that the distribution of y_1 cannot depend on the value of x_2 and vice versa.

However, there are strategies that are neither local nor signaling. For example, perhaps Alice and Bob want to ensure that y_1, y_2 are random bits subject to $y_1 \oplus y_2 = x_1 \wedge x_2$. In this case, the distribution of y_1 does not depend on x_2 (and vice versa) so the strategy is not signaling, but it's also not local. Surprisingly some such strategies which are neither signaling nor local are achievable using quantum mechanics, if Alice and Bob share an entangled quantum state. Einstein referred to this phenomenon as "spooky action at a distance".

In this work, we consider an analogous scenario, first considered by Dwork et al. [DLN+04], where the separation between x_1, x_2 is enforced not via physical distance but by encrypting these bits under two independent public keys.[1] Here Alice gets the two ciphertexts $c_1 \leftarrow \mathsf{Enc}_{\mathsf{pk}_1}(x_1), c_2 \leftarrow \mathsf{Enc}_{\mathsf{pk}_2}(x_2)$, and outputs two other ciphertexts c'_1, c'_2 which are decrypted as $y_i \leftarrow \mathsf{Dec}_{\mathsf{sk}_i}(c'_i)$, $i = 1, 2$. As in the physical analogy, here too we can rule out signaling strategies (if the encryption is semantically secure), and can implement local strategies (if the encryption is homomorphic). But can we replace the entangled state from above by a special "spooky encryption scheme" that would allow Alice to implement spooky strategies? Answering this question is the focus of this work, and we obtain the following results:

- Assuming the hardness of learning with errors (LWE), there exists a secure encryption scheme in which Alice can implement a wide class of spooky strategies that we call *additive function sharing* (AFS) spooky. Namely, for any two-argument function $f : (\{0,1\}^*)^2 \to \{0,1\}$, Alice can convert encryption of inputs $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}_i}(x_i)$ to encryption of outputs $y_i \leftarrow \mathsf{Dec}_{\mathsf{sk}_i}(c'_i)$, ensuring that $y_1 \oplus y_2 = f(x_1, x_2)$, except for a small error probability.

  This construction, described in Section 3, is an almost immediate consequence of the LWE-based multi-key FHE scheme from [CM15, MW16], and it inherits from that multi-key scheme its dependence on a common random string. Namely, the public/secret key-pairs must be generated relative to some common public randomness. Also, the error probability in this spooky scheme depends on the LWE approximation factor: For polynomial approximation factor we get a polynomially-small error probability, and to get negligible error we must assume LWE hardness with a super-polynomial approximation factor.

- In Section 4 we describe a spooky scheme that supports arbitrary two-input spooky relations on short inputs, as well as a very wide class of two-input spooky relations on long inputs. This construction uses probabilistic indistinguishability obfuscation (piO), which is an extension of iO to probabilistic circuits recently introduced by Canetti *et al.* [CLTV15], in conjunction with lossy encryption schemes which are homomorphic and ensure circuit privacy against malicious adversaries. This construction works in the plain model without common-random string and has no error, and it can be realized based on exponentially strong iO, exponentially strong PRFs, and DDH.

[1] Dwork *et al.* consider a PIR scheme rather than an encryption scheme but the translation is immediate.

- In Section 5 we describe a transformation from a scheme that supports only two-input spooky relations on one-bit inputs to one that supports AFS spooky relations on arbitrary number of inputs (of arbitrarily length each). This transformation can be applied to both our LWE-based and piO-based constructions from above.

- We show several implications of (AFS-)spooky encryption. On a negative, it gives a strong counter-example to a method proposed by Aiello et al. [ABOR00] for building succinct arguments for NP from homomorphic encryption, resolving a question posted by [DLN+04]. On a positive, it immediately yields a function secret sharing (FSS) scheme for all functions [BGI15, GI14], and also gives a simple 2-round multi-party computation protocol where, at the end of the first round, the parties can locally compute an additive secret sharing of the output. These application are discussed in Section 6.

- We also study in Section 7 the concept of *spooky free* encryption, i.e., an encryption scheme where we can prove that no spooky strategy is feasible. We show that any non-malleable encryption scheme is spooky-free, and also build spooky-free *homomorphic* encryption schemes from SNARKs. It remains an open problem to construct spooky-free homomorphic encryption under more standard assumptions. Spooky-free homomorphic encryption can be used to instantiate the approach of Aiello et al. to get succinct arguments for NP.

1.1 Technical Overview

1.1.1 LWE-based construction

Our LWE-based construction builds on the multi-key FHE scheme from [CM15, MW16]. In that scheme (after some syntactic massaging) secret keys and single-key ciphertexts are vectors in $\mathbb{Z}_q^n$, and decryption consists of computing the inner product modulo q, $w = \langle \vec{s}, \vec{c} \rangle$, then rounding to the nearest multiple of $q/2$, outputting zero if w is closer to 0 or one if w is closer to $q/2$.

That scheme, however, also supports homomorphic computation across ciphertexts relative to different keys. Roughly, it features a "lifting procedure" where a dimension-n ciphertext vector relative to one key $\vec{s}_i$ is "lifted" to a dimension ℓn vector $\vec{c'} = (\vec{c'}_1, \ldots, \vec{c'}_\ell)$ relative to the concatenated key $\vec{s'} = (\vec{s}_1, \ldots, \vec{s}_\ell)$ of dimension ℓn. These lifted ciphertexts can still be computed on, and the decryption procedure proceeds just as before, except using the higher-dimension vectors. Namely, to decrypt $\vec{c'}$ using $\vec{s'}$, one first computes the inner product $w' = \left\langle \vec{s'}, \vec{c'} \right\rangle$ modulo q, then rounds to the nearest multiple of $q/2$. In other words, we compute the individual inner products $w_i = \left\langle \vec{s}_i, \vec{c'}_i \right\rangle$, then add them all up and round to the nearest multiple of $q/2$.

We observe (cf. Lemma 3.2) that for the special case of two keys, $\ell = 2$, instead of adding the w_i's and then rounding, we can first round each w_i to the nearest multiple of $q/2$ and then add, and this yields the same result with high probability. Specifically, the error probability is propositional to the rounding error for the overall sum w'. Hence by setting the parameters so that w' is very close to a multiple of $q/2$, we can ensure very low error probability.

This observation immediately yields additive function sharing (AFS) spooky encryption for two-argument functions: We just directly use the scheme from [CM15, MW16] to encrypt the two arguments x_1, x_2 under two keys, then use the multi-key evaluation procedure to compute a multi-key ciphertext $\vec{c'} = (\vec{c'}_1, \vec{c'}_2)$ encrypting the value $f(x_1, x_2)$. Viewing each $\vec{c'}_i$ as a single-key ciphertext, we just apply the usual decryption procedure of inner-product and rounding to each of them, and the resulting two bits are an additive secret sharing of $f(x_1, x_2)$, except with a small error probability. The error probability can be made negligible by relying on LWE with a super-polynomial approximation factor.

1.1.2 piO-based construction

The LWE-based construction from above inherits from the underlying FHE scheme [CM15, MW16] its dependence on common public randomness, and it suffers from a small probability of error (depending on parameters, this may or may not be negligible). In Section 4 we show that using iO we can construct an AFS encryption scheme without CRS and without errors, and moreover we can support *arbitrary spooky relations* on two bits, not just additive sharing. For this overview, however, let us focus on the simpler task of constructing AFS spooky scheme for the multiplication function $\mathsf{MULT}(b_1, b_2) = b_1 \cdot b_2$.

The starting point of the construction takes a homomorphic encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ and adds to the public key an obfuscation of the randomized functionality that decrypts, computes the functions f, and re-encrypts secret-sharing of the result. Specifically, let us denote for any $x_1, y_1 \in \{0,1\}$ the function $f_{x_1,y_1}(x_2) = x_1 \cdot x_2 \oplus y_1$, and consider the following randomized program:

```
Program $P_{\mathsf{sk}_1,\mathsf{pk}_1}(c_1, \mathsf{pk}_2, c_2)$
1. $y_1 \leftarrow \{0,1\}$.
2. $c_1' \leftarrow \mathsf{Enc}_{\mathsf{pk}_1}(y_1)$.
3. $x_1 = \mathsf{Dec}_{\mathsf{sk}_1}(c_1)$.
4. $c_2' = \mathsf{Eval}(\mathsf{pk}_2, f_{x_1,y_1}, c_2)$.
5. Output $(c_1', c_2')$.
```

Given the two pairs $(\mathsf{pk}_1, \mathsf{Enc}_{\mathsf{pk}_1}(x_1))$, $(\mathsf{pk}_2, \mathsf{Enc}_{\mathsf{pk}_2}(x_2))$, and access to the program $P_{\mathsf{sk}_1,\mathsf{pk}_1}$, we can run use $P_{\mathsf{sk}_1,\mathsf{pk}_1}(c_1, \mathsf{pk}_2, c_2)$ to get two ciphertexts c_1' and c_2', encrypting y_1, y_2, respectively, such that $y_1 \oplus y_2 = x_1 \cdot x_2$. We would like, therefore, to add an obfuscation of $P_{\mathsf{sk}_1,\mathsf{pk}_1}$ to the public key, thereby obtaining AFS spooky multiplication.

As described, however, this construction is not even secure when $P_{\mathsf{sk}_1,\mathsf{pk}_1}(c_1, \mathsf{pk}_2, c_2)$ is only accessed by a perfect black box. The reason is that if the underlying homomorphic encryption is not *circuit private*, then the evaluated ciphertext c_2' could leak information about x_1. To fix this issue, we require the use of circuit-private homomorphic encryption in this construction. In fact, since the adversary could run the program $P_{\mathsf{sk}_1,\mathsf{pk}_1}(c_1, \mathsf{pk}_2, c_2)$ on arbitrary inputs of its choice, we need a stronger notion of *circuit privacy against malicious adversaries* [OPP14], that guarantees privacy even if the public-key and ciphertext given to the evaluation algorithm are generated adversarially.

Using a malicious circuit private homomorphic encryption scheme, the construction above would be secure if the program $P_{\mathsf{sk}_1,\mathsf{pk}_1}(c_1, \mathsf{pk}_2, c_2)$ is accessed as a perfect black box (e.g., using VBB obfuscation). However, we would like to rely on the weaker notion of indistinguishability obfuscation (iO), or rather probabilistic iO [CLTV15] (since we are dealing with a randomized program). To do so, we need to somehow argue that the secret key sk_1 that is encoded within the program $P_{\mathsf{sk}_1,\mathsf{pk}_1}$ is hidden by the weaker obfuscation. To that end we use a technique introduced in the recent work of Canetti *et al.* [CLTV15]. In one of our hybrids we replace pk_1 with a lossy public-key that reveals nothing about y_1. Once we do that we can use (probabilistic) iO to switch to a program that does not contain sk_1 which implies that the iO indeed hides sk_1.

Finally, we point out that the construction above only uses homomorphic computations for single-bit functions $f_{x,y} : \{0,1\} \to \{0,1\}$ (in addition to probabilistic iO), and there are only four such function (identity, negation, constant 0 and constant 1). A secure and malicious-circuit-private encryption scheme that supports these operations was constructed by Naor and Pinkas [NP01] based on the DDH assumption.

1.1.3 From 2-spooky to n-spooky

Both the LWE and piO based constructions above only support two-argument spooky relations. Specifically the LWE-based scheme only supports AFS-spooky relations for two-argument functions, and the piO-based scheme supports a large class of spooky relations but again, only on two inputs. We extend the supported spooky relations by showing how to transform a scheme that supports multiple hops of AFS-spooky two-input multiplication and single-key additive homomorphism, into a leveled AFS spooky scheme for any number of inputs of any length.

The transformation is inspired by the Goldreich-Micali-Wigderson MPC protocol [GMW87]: Suppose that we are given n public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_n$, bit-by-bit encryptions of the input values $\mathsf{Enc}_{\mathsf{pk}_i}(x_i)$, and an arithmetic circuit $C : (\{0,1\}^*)^n \to \{0,1\}$ that we want to evaluate (i.e., to produce encrypted shares of $C(x_1, \ldots, x_n)$). We process the circuit gate by gate, while maintaining the invariant that for every wire w we produce ciphertexts $\mathsf{Enc}_{\mathsf{pk}_1}(w_1), \ldots, \mathsf{Enc}_{\mathsf{pk}_n}(w_n)$ such that $\oplus_{i\in[n]} w_i$ is equal to the wire w's value. The wires are processed inductively in the following natural way:

1. For an *input* wire holding a bit b, which is part of the j'th input x_j, we take the ciphertext c that encrypts b relative to pk_j, and append to it the ciphertexts $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}_i}(0)$ for all $i \neq j$. Clearly the ciphertexts $(c_1, \ldots, c_{j-1}, c, c_{j+1}, \ldots, c_n)$ are encryptions of an additive sharing of the wire's value b.

2. For an addition gate with input wires u, v and output wire w, by induction we already have $\mathsf{Enc}_{\mathsf{pk}_1}(u_1), \ldots, \mathsf{Enc}_{\mathsf{pk}_n}(u_n)$ and $\mathsf{Enc}_{\mathsf{pk}_1}(v_1), \ldots, \mathsf{Enc}_{\mathsf{pk}_n}(v_n)$. Using just an additive homomorphism on each key individually, we can produce $\mathsf{Enc}_{\mathsf{pk}_1}(u_1 \oplus v_1), \ldots, \mathsf{Enc}_{\mathsf{pk}_n}(u_n \oplus v_n)$ which is the desired secret sharing.

3. For a multiplication gate with input wires u, v and output wire w, again by induction we already have $\mathsf{Enc}_{\mathsf{pk}_1}(u_1), \ldots, \mathsf{Enc}_{\mathsf{pk}_n}(u_n)$ and $\mathsf{Enc}_{\mathsf{pk}_1}(v_1), \ldots, \mathsf{Enc}_{\mathsf{pk}_n}(v_n)$. Using the AFS spooky multiplication we compute an encrypted *tensor product* of the $\vec{u}$ and $\vec{v}$ vectors. Namely, for every i, j we use spooky multiplication to compute

$$\big(\mathsf{Enc}_{\mathsf{pk}_i}(x_{i,j}), \mathsf{Enc}_{\mathsf{pk}_j}(y_{i,j})\big) \leftarrow \mathsf{SpookyMult}\big(\mathsf{Enc}_{\mathsf{pk}_i}(u_i), \mathsf{Enc}_{\mathsf{pk}_j}(u_j)\big),$$

   such that $x_{i,j} \oplus y_{i,j} = u_i \cdot v_j$. Then we collapse this tensor product back into an n-vector using the additive homomorphism relative to each key separately. That is, for every $i \in [n]$ we can compute a ciphertext $\mathsf{Enc}_{\mathsf{pk}_i}(w_i)$ such that $w_i = \bigoplus_{j\in[n]} x_{i,j} \oplus \bigoplus_{j\in[n]} y_{j,i}$.

   We observe that these ciphertexts form a secret sharing of $u \cdot v$. Indeed, adding up the plaintexts we get:

$$\bigoplus_{i\in[n]} \left(\bigoplus_{j\in[n]} x_{i,j} \oplus \bigoplus_{j\in[n]} z_{j,i} \right) = \bigoplus_{i,j\in[n]} (x_{i,j} \oplus y_{i,j}) = \bigoplus_{i,j\in[n]} u_i \cdot v_j = (\bigoplus_i u_i) \cdot (\bigoplus_j v_j) \qquad (1)$$

which is indeed the result of the multiplication gate.

Thus, if the scheme can support $2d$ interleaved hops of (two-key) spooky multiplication and (single-key) additive homomorphism then it is an AFS-spooky scheme for the class of all depth d arithmetic circuits. We note that the resulting scheme does not depend on the number of inputs or their length, and it only depends on the complexity of C inasmuch as the underlying scheme depends on the depth of the evaluated circuit. We also mention that the LWE-based scheme does not naturally allow multiple hops, but we can easily extend it to allow this via bootstrapping.

1.1.4 Applications of Spooky Encryption

In Section 6 we describe both positive and negative applications of spooky encryption. On the positive, it immediately yields a function secret sharing (FSS) scheme for all functions [BGI15, GI14]. Previously such a general function secret sharing scheme was only known to follow from sub-exponentially hard indistinguishability obfuscation [BGI15] whereas we can base it on LWE (using our LWE based spooky encryption).

Spooky encryption also gives a simple 2-round multi-party computation protocol. Roughly, AFS-spooky encryption lets each party broadcast an encryption of its input under its own key, then everyone individually performs the AFS-spooky evaluation locally, each party can locally decrypt and recover a share of the output, and the output is recover using another round of communication. There are some technicalities that should be addressed for this idea to work, and perhaps the easiest way of addressing them is to use AFS-spooky encryption to construct *multi-key FHE with threshold decryption (TMFHE)* (as defined in [MW16]), which can then be used to get a two-round protocol as shown in [MW16]. Using our obfuscation based construction (which does not require a CRS), this gives the first 2-round semi-honest secure MPC protocol in the plain model.[2]

On the negative side, AFS-spooky encryption yields a counter-example for the transformation of Aiello et al. [ABOR00] from multi-prover (MIP) to single-prover protocols. Their idea was to send all of the MIP queries to a single prover, but encrypted under independents keys of a homomorphic encryption scheme. The single prover can homomorphically implement the actions of the MIP provers on the individual encrypted queries, and hopefully the fact that the queries are encrypted under independent keys means that no cross-influence is possible. It is easy to see that spooky encryption violates this hope (by its very nature). Moreover, we show that this transformation can lead to a total break of soundness - in Section 6.1 we show how using AFS-spooky encryption can lead to an unsound single-prover protocol, when the transformation is applied to a simple two-prover protocol for graph 3-colorability.

1.1.5 Spooky-Free Encryption

Finally, in Section 7 we discuss the notion of spooky-free (SF) encryption, which provably ensures that any correlation that an attacker can induce between the original messages $(m_1, \ldots, m_n)$ and "tampered messages" $(m'_1, \ldots, m'_n)$, can be simulated by a "local simulator" that produces m'_i only as a function of m_i (and some shared randomness). See Definition 7.1. To validate this definition, we show that a spooky-free FHE suffices to prove the security of the natural approach

[2]In contrast, [GGHR14] and [MW16] construct 2-round protocols in the *CRS* model. As for security against a *malicious* adversary, [KO04] show that 5 rounds are necessary in the plain model (with respect to black-box proofs of security).

of Aiello *et al.* [ABOR00], which was discussed above, of converting a succinct MIP into a succinct one-round argument discussed above. Indeed, spooky-freeness ensures that the attacker cannot cause more damage from seeing all n ciphertexts than what it could have done by seeing each plaintext independently.

We then turn to the systematic study of spooky-free encryption. First, we show that spooky-freeness implies semantic security. On the other hand, a very weak form of non-malleability (called 1-non-malleability here, or 1-bounded CCA security in [CHH+07]) implies spooky-freeness. However, since the scheme is non-malleable, it is inherently not homomorphic and so we cannot use it to obtain a delegation scheme via the foregoing approach. See Section 7.2.

Indeed, to instantiate the approach of Aiello *et al.* constructing succinct arguments for NP, we need a *homomorphic* encryption scheme which is spooky free. As a proof of concept, in Section 7.3 we show how to built such a homomorphic spooky-free encryption using succinct non-interactive arguments of knowledge (SNARKs [GW11, BSW12]), true-simulation-extractable NIZKs [DHLW10] and regular FHE. While the use of SNARKs makes this construction uninteresting in the application to succinct arguments, the clean definition of SF-encryption, coupled with our "proof of concept" implementation, might open the door for future constructions that will be more useful.

1.2 Related Work

The starting point for this line of work is the natural approach, suggested by Aiello *et al.* [ABOR00], for constructing a secure delegation scheme by combining a multi-prover interactive proof-system (MIP) with a homomorphic encryption scheme as described above. This intuition was questioned by Dwork *et al.* [DLN+04] and our work confirms, under reasonable cryptographic assumptions, that indeed, the approach of [ABOR00] is not necessarily secure.

An approach to overcoming this barrier was taken by Kalai *et al.* [KRR13, KRR14]. They designed a specific MIP (for $\mathcal{P}$) that is sound even against arbitrary no-signaling adversaries. Since semantic-security rules out signaling strategies, they obtain a secure delegation protocol for any language in $\mathcal{P}$.

Multi-key FHE. A notion that is related to spooky-encryption, introduced by López-Alt *et al.* [LTV12] is that of multi-key FHE. In a multi-key FHE, similarly to a spooky encryption scheme, the homomorphic evaluation procedure gets as input n ciphertexts encrypted under different keys. The difference is that the output of the evaluation in a multikey FHE is a single ciphertext that can only be decrypted by combining all the n keys. In contrast, in a spooky encryption scheme the result of the spooky evaluation is n ciphertexts, $c_1, \ldots, c_n$ where each c_i is encrypted under the i^{th} original,. Thus, spooky encryption can be thought of as a specific type of multi-key FHE.

2 Definitions

2.1 Local, No-Signaling, and Spooky Relations

We say that two distributions D_1, D_2 over a (finite) universe $\mathcal{U}$ are ε-close if their statistical distance $\frac{1}{2}||D_1 - D_2||_1$ is at most ε, and denote it by $D_1 \overset{\varepsilon}{\approx} D_2$. We write $D_1 \equiv D_2$ to denote that the distributions are equivalent. We say that D_1, D_2 are δ-far if their statistical distance is *at least* δ.

Definition 2.1. *Let $f : \{0,1\}^{\ell_1} \times \cdots \{0,1\}^{\ell_n} \to \{0,1\}^{\ell'_1} \times \cdots \{0,1\}^{\ell'_n}$ be a* randomized *mapping from n input to n outputs. For input $\vec{x} = (x_1, \ldots, x_n)$ to f, we denote the i'th component of the output by $f(\vec{x})_i$, and more generally for a subset $I \subset [n]$ we denote the projected input by $\vec{x}_I = (x_i : i \in I)$ and the projected output by $f(\vec{x})_I = (f(\vec{x})_i : i \in I)$.*

- *f is* local *if there exist n randomized "component mappings" $f_i : \{0,1\}^{\ell_i} \to \{0,1\}^{\ell'_i}$ such that for all $(x_1, \ldots, x_n) \in \{0,1\}^{\ell_1} \times \cdots \{0,1\}^{\ell_n}$, the distribution $f(x_1, \ldots, x_n)$ is a product distribution $f(x_1, \ldots, x_n) \equiv f_1(x_1) \times \cdots \times f_n(x_n)$.*
- *f is* no-signaling *if for every subset $I \in [n]$ and every two inputs $\vec{x}, \vec{x}'$ with the same I projection, $\vec{x}_I = \vec{x}'_I$, the corresponding projected distributions are equal, $f(\vec{x})_I \equiv f(\vec{x}')_I$.*
- *We say that f is ε-*spooky *for some $\varepsilon > 0$ if it is no-signaling, but for every local f' there exists some input $\vec{x}$ such that $f(\vec{x})$ and $f'(\vec{x})$ are at least ε-far.*

These definitions extends to an ensemble of mappings $F = \{f_k : k \in \mathbb{N}\}$, with the mapping parameters n, ℓ_i, ℓ'_i and the distance bound ε possibly depending on the ensemble parameter k. In this case we say that F is spooky *if the f_k's are ε-spooky for a non-negligible ε.*

As an example, consider the randomized function $f(x_1, x_2) = (y_1, y_2)$ where y_1, y_2 are uniformly random subject to $y_1 \oplus y_2 = x_1 \wedge x_2$. This function is no-signaling since the distributions $f(x)_1$ and $f(x)_2$ are individually uniform, no matter what x is. However, it's easy to show that for any local function $f' = (f'_1, f'_2)$ there is an input $x = (x_1, x_2)$ such that $\Pr[f'_1(x_1) \oplus f'_2(x_2) = x_1 \wedge x_2] \leq 1/2$. Therefore The function f is ε-spooky for $\varepsilon = 1/2$.

2.2 Spooky Encryption

A public-key encryption scheme consists of a tuple $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ of polynomial-time algorithms. The key-generation algorithm Gen gets as input a security parameter $\kappa \in \mathbb{N}$ and outputs a pair of public/private keys $(\mathsf{pk}, \mathsf{sk})$. The encryption algorithm Enc gets as input the public-key pk and a bit $m \in \{0,1\}^{\mathsf{poly}(\kappa)}$ and outputs a ciphertext c, whereas the decryption algorithm Dec gets as input the private-key sk and the ciphertext c and outputs the plaintext bit m. The basic correctness guarantee is that $\Pr[\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m)) = m] > 1 - \mathsf{negl}(k)$, where the probability is over the randomness of all these algorithms. The security requirement is that for every pair of polynomial-sized adversaries (A_1, A_2) it holds that

$$\Pr_{\substack{(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa) \\ b \leftarrow \{0,1\}}} \left[\begin{array}{c} (m_0, m_1) \leftarrow A_1(\mathsf{pk}) \text{ s.t. } |m_0| = |m_1| \\ A_2\left(\mathsf{pk}, \mathsf{Enc}_{\mathsf{pk}}(m_b)\right) = b \end{array} \right] \leq \frac{1}{2} + \text{negl}(\kappa).$$

If the message space consists of just a single bit then we say that the scheme is a bit encryption scheme.

Definition 2.2 (Spooky Encryption)**.** *Let $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key bit-encryption scheme and* Spooky-Eval *be a polynomial-time algorithm that takes as input a (possibly randomized) circuit with $n = n(\kappa)$ inputs and n outputs, $C : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$, and also n pairs of (public-key, ciphertext), and outputs n ciphertexts.*

Let $\mathcal{C}$ be a class of such circuits, we say that $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Spooky\text{-}Eval})$ *is a* $\mathcal{C}$-spooky encryption scheme *if for every security parameter κ, every randomized circuit $C \in \mathcal{C}$, and every input $\vec{x} = (x_1, \ldots, x_n)$ for C, the distributions*

$$SPOOK[C, x_1, \ldots, x_n] \stackrel{\text{def}}{=} \left\{ (\mathsf{Dec}(\mathsf{sk}_1, c'_1), \ldots, \mathsf{Dec}(\mathsf{sk}_n, c'_n)) : \begin{array}{c} \forall i \in [n] \ (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\kappa), \\ \vec{c}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i), \\ (c'_1, \ldots, \ c'_n) \leftarrow \mathsf{Spooky\text{-}Eval}(C, (\mathsf{pk}_i, \vec{c}_i)_i) \end{array} \right\}$$

and $C(x_1, \ldots, x_n)$ are close upto a negligible distance in κ.

We note that the name *spooky encryption* stems from the application of Definition 2.2 to circuits C that compute spooky mappings. Indeed, as shown by Dwork *et al.* [DLN+04], the semantic security of $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ implies that only (almost) no-signaling C's can be realized, and every homomorphic scheme can realize C's that compute product mappings.

Spooky Encryption with CRS. We say that $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Spooky\text{-}Eval})$ is a $\mathcal{C}$-spooky encryption scheme with CRS if Definition 2.2 is satisfied except that we allow all algorithms (and the adversary) to get as input also a public uniformly distributed common random string.

2.3 Additive-Function-Sharing Spooky Encryption

An important special case of spooky encryption allow us to take encryptions $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}_i}(x_i)$ under n independent keys of inputs $x_1, \ldots, x_n$ to an n-argument function f, and produce new ciphertexts under the same n keys that decrypt to additive secret-shares of $y = f(x_1, \ldots, x_n)$. An encryption scheme that supports such "non-interactive sharing" is called *additive-function-sharing spooky encryption* (or AFS-spooky). Several variants of this concept are defined below:

- We can either insist on getting a *random* secret sharing of y, or contend ourselves with *any* secret sharing. Below we call the latter variant *weak* AFS-spooky, and the former is strong AFS-spooky (or just AFS-spooky).

- Similarly to homomorphic encryption schemes, we can have either a *leveled* variant where key-generation receives an additional depth parameter d and the result supports only circuits of depth upto d, or a *fully* AFS-spooky scheme that supports any circuit with a fixed parameter setting.

- We can either allow non-negligible error probability (i.e., the probability that the computation fails to produce a secret-sharing of the right output y), or insist on a negligible error probability. Below we denote by ε-AFS-spooky the variant where the error probability is bounded by some ε (that need not be negligible), and the variant with negligible error probability is just AFS-spooky.

- Sometimes we want to consider only two-argument functions $f(x_1, x_2)$, a scheme that only supports two-argument functions is called AFS-2-spooky.

Definition 2.3 (AFS-Spooky)**.** *Let $\mathcal{E} =$ $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Spooky\text{-}Eval})$ be a scheme where $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a semantically secure public-key bit-encryption.*

- $\mathcal{E}$ *is a* weak ε-additive-function-sharing-spooky (weak-ε-AFS-Spooky) *if for every boolean circuit C computing an n-argument function $f : (\{0,1\}^*)^n \to \{0,1\}$, and any set of inputs $x_1, \ldots, x_n$ for C, it holds that*

$$\Pr\left[\bigoplus_{i=1}^{n} y_i = C(x_1,\ldots,x_n) : \begin{array}{l} (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\kappa), \vec{c}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i), \\ (c'_1, \ldots,\ c'_n) \leftarrow \mathsf{Spooky\text{-}Eval}(C, (\mathsf{pk}_i, \vec{c}_i)_i), \\ y_i \leftarrow \mathsf{Dec}(\mathsf{sk}_i, c'_i) \end{array}\right] \geq 1 - \varepsilon(\kappa).$$

  If the above only holds for $n \leq 2$ (two-argument functions) then we say that $\mathcal{E}$ is weak-ε-AFS-2-Spooky.

- $\mathcal{E}$ *is* ε-AFS-(2)-Spooky *if it is weak ε-AFS-(2-)Spooky, and in addition any $n-1$ of the shares y_i above are distributed ε-close to uniform.*

- $\mathcal{E}$ *is* ε-*leveled*-(weak-)AFS-(2-)Spooky *if the* Gen *procedure receives an additional depth parameter 1^d, and then the conditions above hold only for circuits of depth upto d.*

- $\mathcal{E}$ *is* (leveled-/weak-)AFS-(2-)Spooky *if it is ε-(leveled/weak) AFS-(2-)Spooky for some negligible function $\varepsilon(\kappa)$.*

Relation to Definition 2.2, and more variants. The variants from Definition 2.3 above are all special cases of the more general Definition 2.2. In particular, an AFS-Spooky scheme is $\mathcal{C}$-spooky relative to a class $\mathcal{C}$ that contains a randomized circuit C_f for every n-input function $f : (\{0,1\}^*)^n \to \{0,1\}$, where the output distribution of $C_f(x_1, \ldots, x_n)$ is negligibly close to random n-out-of-n additive secret-sharing of the value $f(x_1, \ldots, x_n)$. Similarly weak-AFS-Spooky is $\mathcal{C}$-spooky relative to a similar class $\mathcal{C}$, except the output of C_f can be any additive secret-sharing of $f(x_1, \ldots, x_n)$, not necessarily a random one.

We note that other variants are also possible (and sometimes useful), for example we can consider a non-binary function f, or secret-sharing modulo some $p > 2$ (or even more general forms of secret sharing). In particular, for the negative example in Section 6 to the multi-prover-to-single-prover transformation, it is convenient to consider a function f that outputs a color $c \in \{0,1,2\}$ and a secret-sharing modulo 3 of that color (and it is sufficient to use ε-leveled-weak-AFS spooky scheme, even for some constant $\varepsilon < 1/2$.

Weak vs. Strong AFS-Spooky Schemes. We observe that the distinction between the strong and weak variants of AFS-spooky is unimportant, indeed there is an easy transformation from any weak AFS-spooky scheme into a strong one. Hence in the sequel we will only be concerned with realizing the weak variant, but will allow ourselves to rely on the strong variant whenever needed.

Lemma 2.4. *There is a transformation that turns any weak AFS-spooky scheme $\mathcal{E}$ into a AFS-spooky scheme $\mathcal{E}'$, by only adding one bit to each ciphertext and only a small constant number of operations to each procedure.*

Proof. Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Spooky\text{-}Eval})$ be a AFS-spooky scheme, the new scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}', \mathsf{Dec}', \mathsf{Spooky\text{-}Eval}')$ has the same key-generation as $\mathcal{E}$, and the encryption, decryption, and evaluation are modified as follows.

- $\mathsf{Enc}'(\mathsf{pk}, m)$ first runs the underlying encryption procedure to get $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$, then outputs the ciphertext $c' = (0, c)$.

- $\mathsf{Spooky\text{-}Eval}'(C, (\mathsf{pk}_i, (b_i, c_i)_i))$ runs the underlying evaluation procedure to get $(c'_1, \ldots, c'_n) \leftarrow \mathsf{Spooky\text{-}Eval}'(C, (\mathsf{pk}_i, c_i)_i)$. Then it chooses n bits $b'_1, \ldots, b'_n$ uniformly at random subject to $b'_1 \oplus \cdots \oplus b'_n = 0$, and outputs $((b'_1, c'_1), \ldots, (b'_n, c'_n))$.
- $\mathsf{Dec}(\mathsf{pk}, (b, c))$ runs the underlying decryption to get $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$. Then it outputs $m = m' \oplus b$.

This modification adds a random additive secret-sharing of 0 to the original secret sharing of y, thus transforming the arbitrary sharing of y into a random sharing of the same y. $\square$

Deterministic Spooky Evaluation. As described, the transformation in the proof of Lemma 2.4 results in a randomized spooky-evaluation procedure, but it can be easily made deterministic by having the encryption procedure append the necessary random bits to the ciphertexts, and using these bits as the randomness needed for the secret-sharing during spooky-evaluation. In fact, the same technique can be used to derandomize any Spooky-Eval procedure, so we can always assume without loss of generality that our Spooky-Eval procedures are deterministic.

3 LWE-Based Spooky Encryption

3.1 Learning with Errors (LWE) and Multi-Key FHE

The LWE assumption roughly says that adding just a little noise to a set of linear equations makes them hard to solve. In our context, we consider equations modulo some integer q and the noise consists of numbers whose magnitude is much smaller than q, as expressed via a noise distribution χ that yields such "small numbers" with high probability. Below we identify $\mathbb{Z}_q$ with the symmetric interval $[-q/2, q/2)$ and let $[x]_q$ denote the reduction of x modulo q into this interval.

Definition 3.1 (Learning With Errors [Reg09]). *Let $n = n(\kappa), q = q(\kappa) \in \mathbb{Z}$ be functions of the security parameter κ and $\chi = \{\chi(\kappa)\}_\kappa$ be a distribution ensemble over $\mathbb{Z}$. The decision-LWE assumption with parameters (n, q, χ) says that for any polynomial $m = m(\kappa) \in \mathbb{Z}$, the following two distribution ensembles are computationally indistinguishable*

$$\begin{aligned} \mathcal{LWE}[n,m,q,\chi] &\overset{\text{def}}{=} \{(A,\vec{b}) : A \leftarrow \mathbb{Z}_q^{n\times m},\ \vec{s} \leftarrow \mathbb{Z}_q^n,\ \vec{e} \leftarrow \chi^m,\ b := [\vec{s}A + \vec{e}]_q\}, \\ \textit{and } \mathcal{U}[n,m,q] &\overset{\text{def}}{=} \{(A,\vec{b}) : A \leftarrow \mathbb{Z}_q^{n\times m},\ \vec{b} \leftarrow \mathbb{Z}_q^m\} \ (\textit{i.e., uniform over } \mathbb{Z}_q^{(n+1)\times m}). \end{aligned}$$

For $\alpha = \alpha(\kappa) \in (0,1)$, the α-DLWE assumption asserts the existence of parameters n, q, χ as above with n polynomial in κ, such that $e \leftarrow \chi$ yields $|e| < \alpha q$ with overwhelming probability.

Note that the α-DLWE assumption becomes stronger as α gets smaller, and it is known to be false in the extreme case where $\alpha = 2^{-\Omega(n)}$ using lattice-reduction techniques. On the other hand, we have ample evidence to belive the α-DLWE assumption with $\alpha = 1/\mathsf{poly}(n)$ [Reg09, Pei09, BLP+13], and it is commonly belived to hold also for super-polynomially (and perhaps even sub-exponentially) small α's.

We show that assuming hardness of the learning-with-errors problem, there exists a function-secret sharing (in the common-random-string model) for any n-argument function f. Our construction builds on the multi-key fully homomorphic encryption construction of Mukherjee and Wichs [MW16], which is a simplification of the Clear-McGoldrick scheme from [CM15]. We summarize the properties of this construction that we need for our purposes.

Theorem 1 ([CM15, MW16]). *Assuming the hardness of α-DLWE (for some $\alpha(\kappa)$), there exists a multi-key homomorphic encryption with the following properties:*

- *The construction works in the common-random-string model. For parameters $n, m, q = \mathsf{poly}(\kappa)$, all instances have access to a uniformly random matrix $A \in \mathbb{Z}_q^{(n-1)\times m}$.*
- *For any depth parameter d, the scheme supports multi-key evaluation of depth-d circuits using public keys of size $d \cdot \mathsf{poly}(\kappa)$, while secret keys are vectors $\vec{s} \in \mathbb{Z}_q^n$, regardless of the depth parameter.*

  Specifically, there is an efficient procedure Eval *that is given as input:*

  – *Parameters $d, \ell \in \mathbb{N}$, and ℓ public keys that support depth-d computations;*
  – *A depth-d circuit computing an ℓ-argument boolean function $f : (\{0,1\}^*)^\ell \to \{0,1\}$;*
  – *Public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ and* fresh encryptions *(bit-by-bit) of each argument $x_i \in \{0,1\}^*$ under key pk_i, denoted $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}_i}(x_i)$.*

  On such input, the Eval *procedure outputs a dimension $n\ell$-vector, $\vec{c}' = (\vec{c}'_1 \ldots \vec{c}'_\ell)$ (with each $\vec{c}'_i \in \mathbb{Z}_q^n$),*[3] *such that for the secret keys $\vec{s}_i$ corresponding to pk_i it holds that*

$$\sum_{i=1}^{\ell} \langle \vec{s}_i, \vec{c}'_i \rangle = \lfloor q/2 \rfloor \cdot f(x_1, \ldots, x_n) + e \pmod{q}$$

  for some error $e \in \mathbb{Z}_q$ with $|e| < \alpha q \cdot \mathsf{poly}(\kappa)$.

By further making a circular-security assumption, there exists a scheme that supports evaluation of circuits of any depth without growing the public keys.

3.2 LWE-Based AFS Spooky Encryption

Below we show that under the decision-LWE assumption we can construct AFS-spooky encryption schemes (in the common-random-string model). Namely, for every n-argument function $f(x_1, \ldots, x_n)$, given encryption of the arguments under n independent public keys, we can compute an encryption of shares under the same keys of an additive secret-sharing of the output $y = f(x_1, \ldots, x_n)$.

Theorem 2. *Assuming the hardness of α-DLWE, there exists a leveled ε-AFS-2-Spooky encryption scheme for $\varepsilon = \alpha \cdot \mathsf{poly}(\kappa)$. Further making a circular-security assumption, we get a (non-leveled) ε-AFS-2-spooky encryption scheme.*

Proof. We show that the encryption scheme from Theorem 1 is already essentially a leveled weak AFS-2-spooky encryption scheme. Specifically, Theorem 1 tells us that given the description of a depth-d circuit C, computing a 2-argument function $f : (\{0,1\}^*)^2 \to \{0,1\}$, together with two public-key and corresponding bit-by-bit encryptions, $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}_i}(x_i)$, the Eval procedure yields $(\vec{c}'_1, \vec{c}'_2) \leftarrow \mathsf{Eval}(C, (\mathsf{pk}_1, c_1), (\mathsf{pk}_2, x_2))$ such that $\langle \mathsf{sk}_1, \vec{c}'_1 \rangle + \langle \mathsf{sk}_2, \vec{c}'_2 \rangle = y \cdot q/2 + e \pmod{q}$, where the sk_i's are the secret keys corresponding to the pk_i's, $y = f(x_1, x_2)$, and $|e| < \alpha q \cdot \mathsf{poly}(\kappa) = \varepsilon q$.

[3]Referring to [MW16, Sec. 5.4], the vector $\vec{c}'_i$ is the result of the product $\hat{C}^{(i)} \times \hat{G}^{-1}(\hat{\vec{w}}^T)$, without the added noise term e_i^{sm}.

Denote $v_i = [\langle \mathsf{sk}_i, \vec{c}_i' \rangle]_q$ for $i = 1, 2$ and $v = [v_1 + v_2]_q$. Lemma 3.2 below says that instead of first adding the v_i's and then rounding to the nearest multiple of $q/2$, we can first round and then add, and this will yield the same result except with error probability of at most 2ε. The only catch is that Lemma 3.2 assumes that v_1, v_2 are chosen at random subject to their sum modulo q being v, whereas in our case we do not have this guarantee. To account for this, we modify our Spooky-Eval procedure, letting it choose a random shift amount $\delta \in \mathbb{Z}_q$ and adding/subtracting it from v_1, v_2, respectively.

In more detail, we change the encryption, decryption, and evaluation in a manner somewhat similar to the proof of Lemma 2.4. Namely, the encryption algorithm Enc' outputs the pair $(\vec{c}, 0)$ rather than just the ciphertext vector $\vec{c}$, and the decryption algorithm Dec', given a pair $(\vec{c}, \delta)$ first computes the shifted inner product $v := [\delta + \langle \mathsf{sk}, \vec{c} \rangle]_q$ and then outputs 0 if $|v| < q/4$ and 1 otherwise.

The Spooky-Eval procedure, given as input C and $(\mathsf{pk}_i, \tilde{c}_i)$, $i = 1, 2$, first strips the 0's from all the pairs $(\vec{c}, 0)$ in the encrypted $\tilde{c}_i$'s retaining only the underlying ciphertexts c_1, c_2. It then applies the undelying evaluation procedure to get

$$(\vec{c}_1', \vec{c}_2') \leftarrow \mathsf{Eval}(C, (\mathsf{pk}_1, c_1), (\mathsf{pk}_2, x_2)).$$

Next it chooses a uniform $\delta \in \mathbb{Z}_q$ and returns the two ciphertexts $(\vec{c}_1', \delta)$ and $(\vec{c}_2', -\delta)$. It is now clear that the values v_1, v_2 that are obtained during decryption are indeed individually random (over the choice of δ), but their sum modulo q does not depend on δ.

Applying Lemma 3.2 and denoting $y_i = \mathsf{Dec}'_{\mathsf{sk}}((\vec{c}_i', \delta_i))$, we have $y_1 \oplus y_2 = C(x_1, x_2)$ except with error probability at most 2ε (over the choise of δ). This yeilds a leveled 2ε-AFS-2-Spooky encryption scheme (using the leveled version of Theorem 1), or a "fully" 2ε-AFS-2-Spooky (if we also assume circular security).[4] $\square$

Lemma 3.2. *Fix some modulus $q \in \mathbb{Z}$, bit $b \in \{0, 1\}$, and a value $v \in \mathbb{Z}_q$ such that $v = b \cdot q/2 + e \pmod{q}$ for some bounded error $|e| < q/4$. Consider choosing v_1, v_2 uniformly at random in $\mathbb{Z}_q$ subject to $v_1 + v_2 = v \pmod{q}$, and denote $v_i = b_i \cdot q/2 + e_i \pmod{q}$ with $b_i = [\lceil v_1 \cdot 2/q \rfloor]_2 \in \{0, 1\}$ and $|e_i| \le q/4$. Then $\Pr_{v_1, v_2}[b_1 \oplus b_2 = b] > 1 - 2(|e| + 1)/q$.*

Proof. Consider first the case $b = 0$ and $v = e \ge 0$. For this case consider choosing at random $v_1 \in \mathbb{Z}_q$ and setting $v_2 = [v - v_1]_q = [e - v_1]_q$. It is easy to check that the condition $b_1 \oplus b_2 = b = 0$ is satisfied whenever we have

$$v_1, v_2 \in \big(\frac{-q}{4} + e, \frac{q}{4}\big) \quad \text{or} \quad v_1, v_2 \in \big[\frac{-q}{2}, \frac{-q}{4}\big) \cup \big(\frac{q}{4} + e, \frac{q}{2}\big).$$

The only error regions are $v_1, v_2 \in (\frac{-q}{4}, \frac{-q}{4} + e)$ and $v_1, v_2 \in (\frac{q}{4}, \frac{q}{4} + v)$, and (depending on rounding) also upto half of the points $v_1 \in \{\frac{-q}{4}, \ \frac{-q}{4} + e, \ \frac{q}{4}, \ \frac{q}{4} + e\} \cap \mathbb{Z}$. The proofs for the other three cases $((b = 0, e < 0), (b = 1, e > 0), (b = 1, e < 0))$ are symmetric. $\square$

3.3 Beyond AFS-2-Spooky Encryption

The construction from Theorem 2 does not directly extend to functions with more than two arguments, since Lemma 3.2 no longer holds for more than two v_i's (even for the no-error case of

[4] We note that the addition of the δ shift in fact yields directly the "strong" version of AFS-2-spooky.

$e = 0$). Instead, we can use the GMW-like transformation that was sketched in the introduction and is described in detail in Section 5 to get a general AFS-spooky scheme.

Recall that the 2-to-ℓ-spooky transformation alternates between two forms of sharing: the "standard" form in which a value y on some wire is represented by a sequence of ciphertexts $(c_i = \mathsf{Enc}_{\mathsf{pk}_i}(y_i))_i$ such that $\sum y_i = y \pmod 2$, and an "extended" form which is essentially an encryption of a tensor product of two y_i-sequences. Going from "standard" to "extended" is done using spooky evaluation of all the pairwise multiplications, and going back to "standard" is done using homomorphic addition (with respect to each key separately).

To support this transformation, we need an AFS-2-spooky scehme which is *multi-hop* (in the sense of [GHV10]), i.e. we need to apply the spooky evaluation procedure not just to fresh ciphertexts, but also to evaluated ciphertexts that resulted from previous applications of spooky evaluation. However, the AFS-2-spooky scheme as described in Theorem 2 does not meet this condition, since that scheme can only process fresh cipehrtexts. (This is an artifact of the underlying multi-key FHE scheme from [CM15, MW16], in which only fresh cipehrtexts can be processed in a multi-key fashion.)

To get a multi-hop scheme, we note that we can apply the same bootstrapping-based transformation as in [GHV10, Theorem 4], which transforms any compact fully-homomorphic scheme to a multi-hop one.[5] That is, by publishing fresh encryption of the secret keys sk_i under public keys pk'_i, one can evaluate any function $f(x_1, \ldots, x_\ell)$ on *any collection of decryptable ciphertexts* c_i (such that $\mathsf{Dec}_{\mathsf{sk}_i}(c_i) = x_i$) by evaluating the function

$$F_{f,c_1,\ldots,c_\ell}(\mathsf{sk}_1, \ldots, \mathsf{sk}_\ell) = f\big(\mathsf{Dec}_{\mathsf{sk}_1}(c_1), \ldots, \mathsf{Dec}_{\mathsf{sk}_\ell}(c_\ell)\big)$$

on the fresh encryption of the sk_i's. Applying this transformation in d layers of interleaved AFS-2-spooky multiplications and single-key addition (and setting the parameters so that the total error probability in all the AFS-2-spooky multiplicatoins is still small enough), we obtain a cryptosystem that supports the transformation from Theorem 9.

Theorem 3. *Assuming the hardness of α-*DLWE*, there exists a leveled FHE scheme that supports d interleaved levels of AFS-2-spooky multiplicatoins and single-key addition, with total error probability $\varepsilon = \alpha \cdot d \cdot \mathsf{poly}(\kappa)$.*

Proof. The cryptosystem from Theorem 1 is augmented by generating $d+2$ secret/public key pairs $(sk^{(i)}, \mathsf{pk}^{(i)})$, $i = 0, 1, \ldots, d+1$, and adding to the public key fresh encryption of each $\mathsf{sk}^{(i)}$ under $\mathsf{pk}^{(i+1)}$ for $i = 0, \ldots, d$. Namely, $\tilde{c}^{(i)} = \mathsf{Enc}_{\mathsf{pk}^{(i+1)}}(\mathsf{sk}^{(}i))$. Ciphertexts in the new cryptosystem are labeled by a *level number* between 0 and d, where a level-i cipehrtext is decryptable using the secret key $\mathsf{sk}^{(i)}$. Fresh encryption are at level 0, and they are encrypted using the first public key $\mathsf{pk}^{(0)}$.

Joining the addition layer after the i'th iteration to the multiplication layer in the $i+1$'st iteration, we get d levels of AFS-2-spooky evaluation of ℓ^2 function of the form

$$f_{i,j}\big((x_{i,1}, \ldots, x_{i,\ell}), (x_{j,1}, \ldots, x_{j,\ell})\big) = \Big(\bigoplus_{k=1}^{\ell} x_{i,k}\Big) \cdot \Big(\bigoplus_{k=1}^{\ell} x_{j,k}\Big),$$

and a final layer of single-key evaluation of homomorphic addition. Each of these functions at each level t is evaluated by applying the Eval procedure to functions of the form

$$\tilde{F}_{c_1\ldots,c_\ell,c'_1\ldots,c'_\ell}(\mathsf{sk}, \mathsf{sk}') \;=\; \Big(\bigoplus_{k=1}^{\ell} \mathsf{Dec}_{\mathsf{sk}}(c_k)\Big) \cdot \Big(\bigoplus_{k=1}^{\ell} \mathsf{Dec}_{\mathsf{sk}'}(c'_k)\Big),$$

[5]The transformation in [GHV10] is described for single-key FHE schemes, but it applies also to multi-key schemes.

evaluated on the fresh encryptions $\tilde{c}_i^{(t)}$, $\tilde{c}_j^{(t)}$. □

Corollary 4. *Assuming the hardness of α-DLWE, there exists a leveled ε-AFS-spooky encryption scheme for $\varepsilon = \alpha \cdot d \cdot \mathsf{poly}(\kappa)$. Further making a circular-security assumption, we get a (non-leveled) ε-AFS-spooky encryption scheme.* □

4 piO based Spooky Encryption

In this section we show a construction based on probabilistic iO, in conjunction with lossy homomorphic encryption, that can support many 2-key spooky relations, even beyond AFS-spooky. Compared to our LWE-based construction from Section 3, the construction here does not need a CRS and has no error probability, and it supports more spooky distributions. On the other hand, we are making a much stronger assumption here, and also we need a different scheme for different spooky relations.[6]

The construction in this section supports in particular the functionality that we need for our generic transformation from Section 5 that turns an AFS-2-spooky scheme to an AFS-n-spooky one. The resulting AFS-n-spooky also does not need a CRS and has no error probability. Moreover, for that case we have a single scheme supporting all AFS-spooky relations.

Organization of this Section. In Section 4.1 we introduce our tools, defining probabilistic indistinguishability obfuscation (using a slightly weaker variant of the definition of Canetti *et al.* [CLTV15]) and lossy homomorphic encryption with malicious circuit privacy. In Section 4.2 we describe and prove our construction for 2-input spooky encryption scheme, and finally in Section 4.3 we show how to obtain a multi-input AFS-spooky encryption.

4.1 Tools

4.1.1 Probabilistic Indistinguishability Obfuscation.

Our construction uses probabilistic iO, a notion that was recently introduced by Canetti *et al.* [CLTV15]. Loosely speaking, this is an obfuscator for probabilistic circuits with the guarantee that the obfuscations of any two "equivalent" circuits are computationally indistinguishable.

Canetti *et al.* define several variants of piO, where the main distinction is the precise formulation of what it means for circuits to be equivalent. Our definition corresponds to a (weakened variant) of their X-Ind piO (which can be realized assuming sub-exponentially secure iO and sub-exponentially secure OWF, see Theorem 5 below). Roughly, our variant only considers pairs of circuits with the property that for *every* input, their output distributions are *identical*, while the definition in [CLTV15] allows a small statistical gap.

To formally define piO, we consider a (possibly randomized) PPT sampling algorithm S that given as input a security parameter 1^κ, outputs a triple (C_0, C_1, z), where C_0 and C_1 are randomized circuits (to be obfuscated) and z is some auxiliary input. We say that a sampler S is an *equivalent-circuit-sampler* if with probability 1 it outputs circuits C_0 and C_1 such that for every x the circuits $C_0(x)$ and $C_1(x)$ generate identical distributions.

[6] We can extend the construction so that a single scheme can handle several spooky relations, as long as there is some way of representing relations where we can verify that a given relation is no-signaling.

Definition 4.1 (Probabilistic Indistinguishable Obfuscation (piO), [CLTV15])**.** *A* probabilistic indistinguishability obfuscator *is a probabilistic polynomial-time algorithm* piO *that, given as input a security parameter* 1^κ *and a* probabilistic *circuit* C*, outputs a circuit* $C' = \mathsf{piO}(1^\kappa, C)$ *(which may be deterministic) of size at most* $|C'| = \mathsf{poly}(\kappa, |C|)$ *such that the following two properties hold:*

1. *For every individual input* x*, the distribution* $C(x)$ *and* $\big(\mathsf{piO}(1^\kappa, C)\big)(x)$ *are identical.*[7]

2. *For every* equivalent-circuit-sampler S*, the following two distribution are computationally indistinguishable:*

$$\{(C_0, C_1, \mathsf{piO}(1^\kappa, C_1), z) : (C_0, C_1, z) \leftarrow S(1^\kappa)\} \overset{c}{=} \{(C_0, C_1, \mathsf{piO}(1^\kappa, C_2), z) : (C_0, C_1, z) \leftarrow S(1^\kappa)\}$$

We note that our correctness guarantee is incomparable to that given by [CLTV15]. Indeed, motivated by their PRF based construction, the [CLTV15] definition basically requires that no PPT adversary can distinguish between oracle access to C and to $\mathsf{piO}(1^\kappa, C)$ (so long as the adversary is not allowed to repeat its queries). On the one hand our definition is weaker in that it only considers each input individually, but on the other hand it is stronger in that it requires that for each such individual input the distributions are *identical*. Our correctness guarantee can be easily obtained using the [CLTV15] construction by using an underlying PRF $\{f_s\}_s$ with the property that $f_s(x)$ is individually uniformly random for every x. The latter can be easily obtained by taking any PRF and xor-ing its output with a fixed random string.

Theorem 5 ([CLTV15])**.** *Assume the existence of a sub-exponentially indistinguishable indistinguishability obfuscator for circuits and a sub-exponentially secure puncturable PRF. Then, there exists a* probabilistic indistinguishability obfuscator*.*

4.1.2 Lossy Encryption

Loosely speaking, a lossy encryption scheme is an encryption scheme in which public-keys are indistinguishable from "lossy keys," and ciphertexts generated using such lossy keys contain no information about their plaintext.

Definition 4.2 ((Perfectly) Lossy Encryption)**.** *We say that an encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* lossy *if there exists a PPT algorithm* $\widetilde{\mathsf{Gen}}$ *that on input* 1^κ *outputs* $\widetilde{\mathsf{pk}}$ *such that:*

- *The distribution* $\widetilde{\mathsf{pk}}$ *is computationally indistinguishable from a public-key* pk *generated by* $\mathsf{Gen}(1^\kappa)$*.*

- *For every two equal-length messages* m_0 *and* m_1*, the distributions* $\mathsf{Enc}\left(\widetilde{\mathsf{pk}}, m_0\right)$ *and* $\mathsf{Enc}\left(\widetilde{\mathsf{pk}}, m_1\right)$ *are* identically *distributed for every* $\widetilde{\mathsf{pk}} \leftarrow \widetilde{\mathsf{Gen}}(1^\kappa)$*.*

A natural relaxation allows for some negligible statistical deviation in the second condition. For our proof to work however we insist on *perfectly* lossy encryption and throughout this work whenever we say lossy encryption we mean perfectly lossy. We note that (perfectly) lossy encryption can be based on any perfectly re-randomizing encryption which in turn can be based on Quadratic Residuosity or DDH.

Remark 4.3. *If an encryption scheme is lossy then it is semantically secure.*

[7]The latter distribution is defined also over the randomnees of piO. Note that this does not imply that the joint distribution for multiple inputs will be the same in the two cases.

4.1.3 Malicious Circuit-Private Encryption

A public-key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, with message space $\{0,1\}^\ell$, is a homomorphic encryption scheme for a class of Boolean circuits $\mathcal{C}$ on ℓ-bit inputs if there exists a PPT algorithm Eval, such that for every key-pair $(\mathsf{pk}, \mathsf{sk})$, circuit $C \in \mathcal{C}$ and ciphertext $c = \mathsf{Enc}_{\mathsf{pk}}(x)$, where $x \in \{0,1\}^\ell$, on input (C, c) the algorithm $\mathsf{Eval}_{\mathsf{pk}}$ outputs c^* such that $\mathsf{Dec}_{\mathsf{sk}}(c^*) = C(x)$. If the length of c^* does not depend on C then we say that the scheme is compact.

As noted in the introduction, our construction requires a homomorphic encryption scheme that has *malicious circuit privacy*, a notion recently studied by Ostrovsky *et al.* [OPP14], which means that the ciphertext c^* does not reveal any non-trivial information about the circuit C which was used to generate it, even for an adversarially designed public-key pk and ciphertext c.

Previous works (e.g., [OPP14]) consider an *unbounded* simulation based definition. However, for our application we cannot afford unbounded simulation and so we settle for an indistinguishability based definition.[8]

Definition 4.4 ((Perfect) Malicious Circuit Privacy)**.** *A homomorphic encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *for a class of circuits* $\mathcal{C}$ *has* perfect malicious circuit privacy *if for every alleged public-key* pk *(including those not in the support of* Gen*), and ciphertext* c^* *(including those not in the support of* Gen*,* Enc*), there exists an "effective plaintext"* $x \in \{0,1\}^\ell$ *such that for every two circuits* $C_1, C_2 \in \mathcal{C}$ *with* $C_1(x) = C_2(x)$*, it holds that* $\mathsf{Eval}_{\mathsf{pk}}(C_1, c)$ *and* $\mathsf{Eval}_{\mathsf{pk}}(C_2, c)$ *are identically distributed (where the probability is only over the randomness of* Eval*).*

Of course one could allow some negligible statistical deviation between the two distributions in the definition but as was the case for lossy encryption, for our application we cannot allow any such deviation.

Naor and Pinkas [NP01] constructed a 2-message oblivious transfer protocol with perfect security against a malicious receiver based on the DDH assumption (see presentation in [HL10, Section 7.2] which also shows that the construction satisfies the indistinguishability based definition). The [NP01] protocol can be easily transformed into a (non-compact) homomorphic encryption scheme with malicious circuit privacy for inputs of logarithmic length. A folklore construction combines the latter with an information theoretic variant of Yao's garbled circuit for NC_1 [IK00] to obtain malicious circuit privacy for logarithmic *depth* circuits.

Theorem 6 (Folklore)**.** *Assuming the hardness of* DDH*, there exists a lossy homomorphic encryption scheme with malicious circuit privacy for* NC_1 *circuits.*

The fact that the encryption scheme can be lossy follows from the fact that the encryption algorithm in [NP01] basically just generates DDH tuples. We can make the scheme lossy by generating a single DDH triple as part of the public-key and re-randomizing it in the encryption procedure (rather than generating a fresh DDH tuple). The indistinguishable lossy key will contain a random triplet of group element which do not form a DDH tuple.

Remark 4.5. *Ostrovsky et al. [OPP14] show that using their unbounded (statistical) simulation based definition, there exists a scheme for any poly-size circuit assuming compact* FHE *in addition to* DDH *(or any bounded-depth circuit assuming leveled* FHE*). We believe that the their techniques should apply also to our (perfect) indistinguishability based definition but we leave this to a future revision.*

[8]Note that there are known impossibility results for achieving efficient simulation, see [OPP14] and references therein.

Homomorphic Evaluation of Randomized Circuits. For our application, we need to handle randomized circuits rather than deterministic ones. We extend the syntax of homomorphic evaluation to support evaluating a randomized circuit: For a randomized circuit C on ℓ-bit inputs, a public key pk, and ciphertext c that allegedly encrypts the input to C, we let $\mathsf{Eval}_{\mathsf{pk}}(C, c)$ denote the distribution which is induced by choosing randomness r for C, defining $C[r](\cdot) = C(\cdot\,; r)$, and then outputting $\mathsf{Eval}_{\mathsf{pk}}(C[r], c)$. We rely on the following lemma (which is folklore, although we could not find a proof of it in the literature).

Lemma 4.6. *Let* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *be a homomorphic encryption scheme for a class of* randomized *circuits* $\mathcal{C}$*, that enjoys perfect malicious circuit privacy.*

Then, for every alleged public-key pk *and ciphertext* c*, there exists an "effective plaintext"* $x \in \{0,1\}^\ell$ *such that for every two circuits* $C_1, C_2 \in \mathcal{C}$ *for which the output distributions* $C_1(x)$ *and* $C_2(x)$ *are identical, it holds that also the two distributions* $\mathsf{Eval}_{\mathsf{pk}}(C_1, c)$ *and* $\mathsf{Eval}_{\mathsf{pk}}(C_2, c)$ *are identical.*

Proof. This lemma follows by mapping the randomness of the two circuits. First, we can assume without loss of generality that both circuits rely on the same randomness space (e.g., by considering the joint space which is the Cartesian product of the spaces for the two circuits). Denote this joint randomness space by $\mathcal{R}$. Since $C_1(x; r)$ and $C_2(x; r)$ are identically distributed (over the choise of $r \in \mathcal{R}$), there exists a permutation π over $\mathcal{R}$ such that for every $r \in \mathcal{R}$ it holds that $C_1(x; r) = C_2(x; \pi(r))$ (as deterministic circuits).

Note that the distribution $\mathsf{Eval}_{\mathsf{pk}}(C_1, c)$ is a convex combination of the distributions $\mathsf{Eval}_{\mathsf{pk}}(C_1[r], c)$ for the different r's, and $\mathsf{Eval}_{\mathsf{pk}}(C_2, c)$ is a convex combination of the distributions $\mathsf{Eval}_{\mathsf{pk}}(C_2[\pi(r)], c)$. We conclude the proof by appealing to the circuit privacy of the encrypiton scheme. $\square$

4.2 Two-Key Spooky Encryption from piO

Our construction relies on a property of two-input relations that we call *re-sampleability*. Roughly, it should be possible to sample efficiently from the distribution of the second coordinate conditioned on a particular fixed value for the first coordinate.

Definition 4.7 (Efficiently Re-Sampleable)**.** *A randomized polynomial-size circuit* $C : \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2} \to \{0,1\}^{\ell'_1} \times \{0,1\}^{\ell'_2}$ *is* efficiently re-sampleable *if there exists a polynomial-size randomized "resampling circuit"* RS_C*, such that for any input* (x_1, x_2) *to* C*, the distribution* $C(x_1, x_2)$ *is identical to the "resampled distribution"* $\{(y_1, y'_2) : (y_1, y_2) \leftarrow C(x_1, x_2),\ y'_2 \leftarrow RS_C(x_1, x_2, y_1)\}$.

We construct a 2-key spooky scheme that supports any 2 input/output circuit that is both *efficiently re-sampleable* and *no-signaling.*

Theorem 7 (2-Key Spooky Encryption from piO)**.** *Let* $C : \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2} \to \{0,1\}^{\ell'_1} \times \{0,1\}^{\ell'_2}$ *be an efficiently re-sampleable no-signaling circuit, with re-sampling circuit* RS_C*. If there exist (1)* piO*, and (2) a perfectly-lossy homomorphic encryption scheme that can evaluate* C *and* RS_C*, and is perfectly malicious circuit private, then there exists a* C*-spooky encryption scheme, which is also perfectly lossy (and hence semantically secure).*

We stress that the encryption scheme that we need for Theorem 7 must be able to evaluate C and RS_C and be perfectly malicious circuit private, but *it need not be compact.* Theorem 6 gives such a scheme for NC_1 circuits based on DDH and as noted in Remark 4.5, we believe that a scheme that supports any poly-size circuit can be constructed assuming both DDH and FHE (note that [CLTV15] show that full-fledged FHE can be built based on piO).

Remark 4.8 (Almost No-Signaling)**.** *We note that for Theorem 7 we need C to be perfectly no-signaling, specifically we need $C(x,y)_1$ and $C(x,y')_1$ to be identically distributed for all x,y,y'. A natural relaxation of no-signaling circuits, considered in previous works (e.g., [DLN+04, KRR13, KRR14]), allows these distributions to be statistically close (rather than identical). Such circuit is called* almost no-signaling.

Dwork et al. (using a slightly different terminology) showed that a semantically secure scheme can only be spooky for almost no-signaling circuit. The question of constructing a scheme that supports also almost no-signaling circuits is left to future work.

4.2.1 Proof of Theorem 7

Let piO be a probabilistic indistinguishability obfuscator and let $(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$ be the encryption scheme from the theorem statement. Let $\widetilde{\mathsf{Gen}}$ be the corresponding lossy key generation algorithm and Eval the homomorphic evaluation algorithm with malicious circuit privacy.

Each instance of our construction uses two public/secret keys pairs, where only the first pair is used for "normal encryption and decryption," and the other pair is only used for spooky evaluation. In addition to the two pairs, the public key also contains an obfuscated program that implements spooky evaluation using the secret key. That obfuscated program has a secret key hard-wired, and given two ciphertexts c_1, c_2 it decrypt the first one, then evaluates the re-sampling circuit RS_C homomorphically on the other. A complete description of the resulting scheme is found in Fig. 1.

We first show that the scheme supports spooky evaluation of the circuit C and then show that it is a lossy encryption scheme (and in particular is semantically secure).

Lemma 4.9. *The scheme* $(\mathsf{Gen\text{-}Spooky},\mathsf{Enc\text{-}Spooky},\mathsf{Dec\text{-}Spooky},\mathsf{Spooky\text{-}Eval})$ *is C-spooky.*

Proof. The spooky evaluation procedure gets as input two public-keys $\mathsf{pk\text{-}spooky}_1 = (\mathsf{pk1}_1,\mathsf{pk2}_1,\tilde{P}_1)$, $\mathsf{pk\text{-}spooky}_2 = (\mathsf{pk1}_2,\mathsf{pk2}_2,\tilde{P}_2)$, and matching ciphertexts $c_1 = \mathsf{Enc\text{-}Spooky}(\mathsf{pk\text{-}spooky}_1, x_1)$ and $c_2 = \mathsf{Enc\text{-}Spooky}(\mathsf{pk\text{-}spooky}_2, x_2)$ (for some inputs x_1, x_2 to C). It simply runs the obfuscated program $\tilde{P}_1 = \mathsf{piO}(1^\kappa, P[\mathsf{sk1}_1,\mathsf{pk2}_1])$ on input $(c_1,\mathsf{pk1}_2,c_2)$ and returns its output.

By construction and using the correctness of piO, this procedure outputs c_1' and c_2' such that $c_1' \leftarrow \mathsf{Enc}(\mathsf{pk2}_1, y_1)$, where $y_1 \leftarrow \big(C(x_1, 0^{\ell_2})\big)_1$, and $c_2' \leftarrow \mathsf{Eval}_{\mathsf{pk1}_2}(\mathsf{RS}[x_1,y_1,r],c_2)$, where $\mathsf{RS}[x_1,y_1,r](x_2) \equiv RS_C(x_1,x_2,y_1;r)$. By the no-signaling property y_1 is distributed identically to $y_1' \leftarrow \big(C(x_1,x_2)\big)_1$ and so c_2' is distributed as $\mathsf{Eval}_{\mathsf{pk1}_2}(\mathsf{RS}[x_1,y_1',r],c_2)$. Hence

$$\begin{aligned}\mathsf{Dec\text{-}Spooky}(\mathsf{sk\text{-}spooky}_1,c_1') &= \mathsf{Dec}_{\mathsf{sk1}_1}\big(\mathsf{Enc}(\mathsf{pk2}_1,y_1')\big) = y_1' \\ \text{and } \mathsf{Dec\text{-}Spooky}(\mathsf{sk\text{-}spooky}_2,c_2') &= \mathsf{RS}[x_1,y_1',r](x_2) = RS_C\big(x_1,x_2,y_1';r\big)_2.\end{aligned}$$

By definition of the re-sampling procedure, the joint distribution $\Big(\mathsf{Dec\text{-}Spooky}(\mathsf{sk\text{-}spooky}_1,c_1'),$ $\mathsf{Dec\text{-}Spooky}(\mathsf{sk\text{-}spooky}_2,c_2')\Big)$ is identical to $C(x_1,x_2)$, as required. □

Lemma 4.10. *The scheme* $(\mathsf{Gen\text{-}Spooky},\mathsf{Enc\text{-}Spooky},\mathsf{Dec\text{-}Spooky})$ *is a perfectly lossy encryption scheme.*

The probabilistic circuit $P[\mathsf{sk1}, \mathsf{pk2}](c_1, \mathsf{pk}, c)$:

Hardwired: a private-key $\mathsf{sk1}$ and a public-key $\mathsf{pk2}$.
Input: a ciphertext c_1 (presumably under $\mathsf{pk1}$),
and additional (presumably matching) public-key pk and ciphertext c.

1. Decrypt $x_1 \leftarrow \mathsf{Dec}_{\mathsf{sk1}}(c_1)$;[a]
2. Choose randomness $r, r' \leftarrow \{0,1\}^*$ for C and RS_C, respectively;
3. Set $y_1 \leftarrow C(x_1, 0^{\ell_2}; r)_1$ and encrypt $c'_1 \leftarrow \mathsf{Enc}_{\mathsf{pk2}}(y_1)$;
4. Define the circuit $\mathsf{RS}[x_1, r, r'](x_2) \;\equiv\; RS_C(x_1, x_2, \overbrace{C(x_1, 0^{\ell_2}; r)_1}^{=y_1}; r')$;
5. Compute homomorphically $c'_2 \leftarrow \mathsf{Eval}_{\mathsf{pk}}(\mathsf{RS}[x_1, r, r'], c)$.
6. Output $\big((2, c'_1), (1, c'_2)\big)$.[b]

piO based Spooky Encryption

- $\mathsf{Gen\text{-}Spooky}(1^\kappa)$:
  1. Select $(\mathsf{pk1}, \mathsf{sk1}), (\mathsf{pk2}, \mathsf{sk2}) \leftarrow \mathsf{Gen}(1^\kappa)$, and set $\tilde{P} \leftarrow \mathsf{piO}(1^\kappa, P[\mathsf{sk1}, \mathsf{pk2}])$.
  2. Output the secret key $\mathsf{sk\text{-}spooky} = (\mathsf{sk1}, \mathsf{sk2})$ and public key $\mathsf{pk\text{-}spooky} = \Big(\mathsf{pk1}, \mathsf{pk2}, \tilde{P}\Big)$.
- $\mathsf{Enc\text{-}Spooky}\Big((\mathsf{pk1}, \mathsf{pk2}, \tilde{P}), x\Big)$: Output $\big(1, \mathsf{Enc}_{\mathsf{pk1}}(x)\big)$.
- $\mathsf{Dec\text{-}Spooky}\big((\mathsf{sk1}, \mathsf{sk2}), (tag, c)\big)$: If $tag = 1$ output $\mathsf{Dec}_{\mathsf{sk1}}(c)$, else output $\mathsf{Dec}_{\mathsf{sk2}}(c)$.
- $\mathsf{Spooky\text{-}Eval}\big((\mathsf{pk1}_1, \mathsf{pk2}_1, \tilde{P}_1), c_1,\ (\mathsf{pk1}_2, \mathsf{pk2}_2, \tilde{P}_2), c_2, \big)$: Output $\tilde{P}_1(c_1, \mathsf{pk1}_2, c_2)$.

[a] We assume that Dec always returns some value, even if c_1 is not a valid ciphertext.
[b] The tags "2", "1" signal to the decryption algorithm which secret key to use.

Figure 1: piO based Spooky Encryption

The probabilistic circuit $P'[\mathsf{sk1}, \mathsf{pk2}](c_1, \mathsf{pk}, c)$:

The same as $P[\mathsf{sk1}, \mathsf{pk2}](c_1, \mathsf{pk}, c)$, but setting $c_1 \leftarrow \mathsf{Enc}_{\mathsf{pk}_2}(0^{\ell_1})$ in Step 3 rather than $c_1 \leftarrow \mathsf{Enc}_{\mathsf{pk}_2}(y_1)$.

The probabilistic circuit $P''[\mathsf{pk2}]$:

Hardwired: a public-key $\mathsf{pk2}$.
Input: a ciphertext c_1, a public-key pk and a ciphertext c (presumably under pk).

1. Encrypt $c'_1 \leftarrow \mathsf{Enc}_{\mathsf{pk2}}(0^{\ell_1})$.
2. Choose randomness $r \leftarrow \{0,1\}^*$ for C, and define $f[r](\cdot) \equiv C(0^{\ell_1}, \cdot\,; r)_2$.
3. Compute homomorphically $c'_2 \leftarrow \mathsf{Eval}_{\mathsf{pk}}(f[r], c)$.
4. Output $\big((2, c_1), (1, c'_2)\big)$.

Figure 2: The Probabilistic Circuits $P'[\mathsf{sk1}, \mathsf{pk2}]$ and $P''[\mathsf{pk2}]$

Proof. We need to show that there is an alternative key-generation procedure $\widetilde{\mathsf{Gen\text{-}Spooky}}$, producing public keys that are indistinguishable from the real ones, but such that ciphertexts encrypted relative to these keys contain no information about the encrypted plaintext.

The main challenge in establishing the lossiness of the scheme is in showing that the public-keys are indistinguishable from lossy keys despite the obfuscated programs in the public-key (which depend on the corresponding secret keys). Toward that end, we will (gradually) show that these obfuscated programs are computationally indistinguishable from programs that do not depend on the secret keys.

Below we state and prove a few claims, where we consider the distributions $(\mathsf{pk1}, \mathsf{sk1}), (\mathsf{pk2}, \mathsf{sk2}) \leftarrow \mathsf{Gen}(1^\kappa)$ and $\widetilde{\mathsf{pk1}}, \widetilde{\mathsf{pk2}} \leftarrow \widetilde{\mathsf{Gen}}(1^\kappa)$, where $\widetilde{\mathsf{Gen}}$ is the lossy key-generation of the underlying encryption scheme.

Claim 4.10.1. $\Big(\mathsf{pk1}, \mathsf{pk2}, \mathsf{piO}(1^\kappa, P[\mathsf{sk1}, \mathsf{pk2}])\Big) \overset{c}{=} \Big(\mathsf{pk1}, \widetilde{\mathsf{pk2}}, \mathsf{piO}(1^\kappa, P[\mathsf{sk1}, \widetilde{\mathsf{pk2}}])\Big)$.

Proof. Follows from the indistinguishability between standard and lossy public-keys of the underlying scheme (see Definition 4.2). □

Claim 4.10.2. $\Big(\mathsf{pk1}, \widetilde{\mathsf{pk2}}, \mathsf{piO}(1^\kappa, P[\mathsf{sk1}, \widetilde{\mathsf{pk2}}])\Big) \overset{c}{=} \Big(\mathsf{pk1}, \widetilde{\mathsf{pk2}}, \mathsf{piO}(1^\kappa, P'[\mathsf{sk1}, \widetilde{\mathsf{pk2}}])\Big)$*, where* $P'[\mathsf{sk1}, \widetilde{\mathsf{pk2}}]$ *is similar to* $P[\mathsf{sk1}, \widetilde{\mathsf{pk2}}]$ *except that it encrypts* 0^{ℓ_1} *rather than* y_1 *in Step 3, see Fig. 2.*

Proof. Follows from the fact that $\widetilde{\mathsf{pk2}}$ is a *lossy* public-key and therefore $\mathsf{Enc}_{\widetilde{\mathsf{pk2}}}(0^{\ell_1})$ and $\mathsf{Enc}_{\widetilde{\mathsf{pk2}}}(y_1)$ are *identically* distributed (cf. Definition 4.2). □

We proceed to the main claim:

Claim 4.10.3. $\Big(\mathsf{pk1}, \widetilde{\mathsf{pk2}}, \mathsf{piO}(1^\kappa, P'[\mathsf{sk1}, \widetilde{\mathsf{pk2}}])\Big) \overset{c}{=} \Big(\mathsf{pk1}, \widetilde{\mathsf{pk2}}, \mathsf{piO}(1^\kappa, P''[\widetilde{\mathsf{pk2}}])\Big)$*, where the program* $P''[\widetilde{\mathsf{pk2}}]$*, defined in Fig. 2, does not have the secret key* $\mathsf{sk1}$ *(hence it cannot recover* x_1 *or compute* y_1*), so it evaluates homomorphically* $C(0^{\ell_1}, x_2)_2$ *rather than* $RS_C(x_1, x_2, y_1)$ *on* $c = \mathsf{Enc}_{\mathsf{pk}}(x_2)$*.*

Proof. We will show that for every valid secret key $\mathsf{sk1}$ and arbitrary public key $\widetilde{\mathsf{pk2}}$, the randomized programs $P'[\mathsf{sk1}, \widetilde{\mathsf{pk2}}]$ and $P''[\widetilde{\mathsf{pk2}}]$ are functionally identical, in the sense that their outputs are identically distributed for every input. The claim will then follow from the fact that piO is a probabilistic indistinguishability obfuscator (see Definition 4.1).

Note that the first output $c'_1 = \mathsf{Enc}_{\mathsf{pk}}(0^{\ell'_1})$ is generated identically by the two programs, and is independent of everything else that happens in these programs, so we only need to show that the second output c'_2 is identically distributed. To show this, we first establish that c'_2 is an encryption under pk of a value y_2 that is distributed identically in the two programs, and then we appeal to the malicious circuit-privacy of the underlying scheme to conclude that also c'_2 itself is identically distributed.

For starters, fix some arbitrary $x_1 \in \{0,1\}^{\ell_1}$ and $x \in \{0,1\}^{\ell_2}$, and consider the following distributions

$$\begin{aligned}
\mathcal{D}_1[x_1, x] &= \big\{y_1 \leftarrow C(x_1, 0^{\ell_2})_1,\ \text{output } y_2 \leftarrow RS_C(x_1, x, y_1)\big\}, && \text{// Output distribution of } P' \\
\mathcal{D}_2[x_1, x] &= \big\{y_1 \leftarrow C(x_1, x)_1,\quad \text{output } y_2 \leftarrow RS_C(x_1, x, y_1)\big\}, && \\
\mathcal{D}_3[x_1, x] &= \big\{\text{output } y_2 \leftarrow C(x_1, x)_2\big\}, && \\
\mathcal{D}_4[x] &= \big\{\text{output } y_2 \leftarrow C(0^{\ell_1}, x)_2\big\}. && \text{// Output distribution of } P''
\end{aligned}$$

Since C is a no-signaling circuit then we have $\mathcal{D}_1[x_1, x] = \mathcal{D}_2[x_1, x]$ and $\mathcal{D}_3[x_1, x] = \mathcal{D}_4[x]$, and since R_C is the re-sampling circuit for C then we also have $\mathcal{D}_2[x_1, x] = \mathcal{D}_3[x_1, x]$. We therefore conclude that the two distributions $\mathcal{D}_1[x_1, x]$ and $\mathcal{D}_4[x]$ are identical for every x_1, x.

Now consider $x_1 = \mathsf{Dec}_{\mathsf{sk1}}(c_1)$ and x the "effective plaintext" for pk, c (which must exist since the underlying scheme is malicious circuit-private). Recall that the second output of $P'[\mathsf{sk1}, \widetilde{\mathsf{pk2}}]$ consists of a homomorphic evaluation of $\mathcal{D}_1[x_1, x]$, while the second output of $P''[\widetilde{\mathsf{pk2}}]$ consists of homomorphic evaluation of $\mathcal{D}_4[x]$. Applying Lemma 4.6, we conclude that these outputs are identically distributed.

Having established that the output distributions of $P'[\mathsf{sk1}, \widetilde{\mathsf{pk2}}]$ and $P''[\widetilde{\mathsf{pk2}}]$ are identical (for every input), Claim 4.10.3 follows because piO is a probabilistic indistinguishability obfuscator. $\square$

Claim 4.10.4. $\left(\mathsf{pk1}, \widetilde{\mathsf{pk2}}, \mathsf{piO}(P''_{\widetilde{\mathsf{pk2}}})\right) \overset{c}{=} \left(\widetilde{\mathsf{pk1}}, \widetilde{\mathsf{pk2}}, \mathsf{piO}(P''_{\widetilde{\mathsf{pk2}}})\right)$.

Proof. This claim too follows from the indistinguishability between standard and lossy public-keys of the underlying scheme (see Definition 4.2). $\square$

Combining Claims 4.10.1-4.10.4, we conclude that the two distributions $\left(\mathsf{pk1}, \mathsf{pk2}, \mathsf{piO}(P_{\mathsf{sk1},\mathsf{pk2}})\right)$ and $\left(\widetilde{\mathsf{pk1}}, \widetilde{\mathsf{pk2}}, \mathsf{piO}(P''_{\widetilde{\mathsf{pk2}}})\right)$ are computationally indistinguishable. We complete the proof of Lemma 4.10 by observing that keys drawn from the latter distribution are lossy, since the key $\widetilde{\mathsf{pk1}}$ is lossy, the $\mathsf{Enc\text{-}Spooky}$ procedure just uses the underlying encryption procedure with $\widetilde{\mathsf{pk1}}$, and the program $P''[\mathsf{pk2}]$ that we obfuscate is independent of $\widetilde{\mathsf{pk1}}$. $\square$

4.3 piO based Multi-key Spooky Encryption

To obtain a multi-key spooky encryption scheme we would like to invoke our general transformation from 2-key spooky encryption to n-key spooky encryption (see Theorem 9). The scheme in Theorem 7 can clearly be instantiated to support spooky multiplication. To use Theorem 9 however, we need it to also multiple hops of both (single-key) additive homomorphism and spooky multiplication. This is obtained by the following lemma:

Lemma 4.11. *Assume existence of (1)* piO *and (2) a lossy encryption scheme that is homomorphic for all one-bit to one-bit functions with perfect malicious circuit privacy.*

Then, for every $d = d(\kappa)$, there exists an encryption scheme that supports d interleaved levels of AFS-2-spooky multiplications and single-key additions.

Proof Sketch. To obtain an additive homomorphism, we use a construction of Canetti *et al.* [CLTV15] which, assuming piO, transforms any lossy encryption into a d-leveled FHE. This is done by taking d copies of keys of the original lossy scheme and publishing $d-1$ obfuscated programs where the i^{th} obfuscated program takes as input two ciphertexts encrypted under the i^{th} key, decrypts them (using the i^{th} private-key which is hard-wired) applies one operation (AND, XOR, NAND, etc.) and encrypts the result under the $(i+1)^{\text{th}}$ key. Using the fact that the scheme is lossy, Canetti *et al.* show that the piO obfuscation hides the hard-wired private keys and semantic security is maintained.

For our application, we need to compute multiple spooky multiplications, and then sub them up with single-key addition. To get n-input AFS-spooky we need to sum up n ciphertexts, which can be done using an addition tree of depth $d = \log n$.

Looking more closely at the [CLTV15] construction, we observe that by setting $d = i \log n$ we can already support i interleaving hops of (single-key) additive homomorphism and 2-input spooky multiplications. This follows from the fact that the [CLTV15] transformation has the property that after every additive homomorphic operation, we obtain a fresh ciphertext (under a new-key). $\square$

Using the scheme from Lemma 4.11 and applying Theorem 9, we obtain the following result:

Theorem 8 (n-Key Spooky from piO)**.** *Assume existence of (1)* piO *and (2) a lossy encryption scheme that is homomorphic for all single-bit to single-bit functions with perfect malicious circuit privacy. Then there exists a leveled AFS-spooky encryption scheme.*

5 From 2-Input to n-Input AFS-Spooky

Below we describe our transformation from AFS-2-spooky to AFS-n spooky scheme, inspired by the GMW MPC protocol [GMW87] (see the presentation in [Gol04, Section 7]).

Theorem 9 (2-Spooky to n-Spooky)**.** *Let $d = d(\kappa)$ and assume that there exists a public-key bit-encryption scheme that supports $2d$ (interleaving) hops of (1) single-key compact additive homomorphism and (2) two-key spooky multiplication. Then, that same scheme is a d-level AFS-spooky encryption.*

Proof. Let $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be the encryption scheme in the theorem statement, let $\mathsf{Spooky\text{-}Mult}$ be the spooky multiplication PPT algorithm and let Eval be the single-key homomorphic evaluation algorithm (that supports compact additive homomorphism). We show a procedure that given as input:

1. A depth-d, fan-in-2, n-input arithmetic circuit over $GF(2)$, $C : (\{0,1\}^*)^n \to \{0,1\}$;
2. n public-keys $\mathsf{pk}_1, \dots, \mathsf{pk}_n$; and
3. n ciphertexts $c_1, \dots, c_n$, where $c_j = \mathsf{Enc}(\mathsf{pk}_j, x_j)$,

outputs a sequence of ciphertexts $c'_1, \dots, c'_n$ such that $\sum_{j \in [n]} \mathsf{Dec}_{\mathsf{sk}_j}(c'_j) = C(x_1, \dots, x_n)$ (where all arithmetic is over $\mathsf{GF}(2)$).

The procedure processes the circuit wire by wire. We maintain the invariant that whenever a wire w is processed, the procedure generates ciphertexts $c_1^{(w)}, \dots, c_n^{(w)}$ such that $\sum_{j \in [n]} \mathsf{Dec}_{\mathsf{sk}_j}(c_j^{(w)})$ is the correct value of the wire w (when the circuit C is evaluated on input $(x_1, \dots, x_n)$. Furthermore, if the wire w is at distance i from the input then $c_1^{(w)}, \dots, c_n^{(w)}$ have passed at most $2i$ hops of homomorphic operations. In particular, at the end of the process the procedure will have generated the sequence of ciphertexts $c_1^{\text{out}}, \dots, c_n^{\text{out}}$ such that $\sum_{j \in [n]} \mathsf{Dec}_{\mathsf{sk}_j}(c_j^{\text{out}})$ is equal to the output value of the circuit, as required. We proceed to describe how the wires are (inductively) processed.

Consider an *input* wire w, corresponding to an input bit b which is part of the i^{th} input x_i, and for which we are given the input ciphertext $c = \mathsf{Enc}_{\mathsf{pk}_i}(b)$. For that wire we set $c_i^{(w)} = c$ and $c_j^{(w)} = \mathsf{Enc}_{\mathsf{pk}_{j'}}(0)$ for all $j \neq i$. Hence, $\sum_{j \in [n]} \mathsf{Dec}_{\mathsf{sk}_j}(c_j^{(w)}) = \mathsf{Dec}_{\mathsf{sk}_i}(c) = b$, which is the correct value for the wire w.

Consider a gate g with input wires u, v and output wire w. Let b_u (resp., b_v) be the value on the wire u (resp., v) when C is evaluated on input $(x_1, \dots, x_n)$. By induction, we have already generated ciphertexts $c_1^{(u)}, \dots, c_n^{(u)}$ and $c_1^{(v)}, \dots, c_n^{(v)}$ such that $\sum_{j\in[n]} \mathsf{Dec}_{\mathsf{sk}_j}(c_j^{(u)}) = b_u$ and $\sum_{j\in[n]} \mathsf{Dec}_{\mathsf{sk}_j}(c_j^{(v)}) = b_v$.

For the case that g is an addition gate, we set $c_j^{(w)} = \mathsf{Eval}\left(\mathsf{pk}_j, \oplus, c_j^{(u)}, c_j^{(v)}\right)$ and we get:

$$\sum_{j\in[n]} \mathsf{Dec}_{\mathsf{sk}_j}(c_j^{(w)}) = \sum_{j\in[n]} \mathsf{Dec}_{\mathsf{sk}_j}(\mathsf{Eval}_{\mathsf{pk}_j}(\oplus, c_j^{(u)}, c_j^{(v)})) = \sum_{j\in[n]} \mathsf{Dec}_{\mathsf{sk}_j}(c_j^{(u)}) \oplus \mathsf{Dec}_{\mathsf{sk}_j}(c_j^{(v)}) = b_u \oplus b_v,$$

which is the correct value for the wire w. Furthermore, each new ciphertext was obtained by just a single homomorphic operation.

Now consider the case that g is a multiplication gate. We first compute auxiliary ciphertexts $(f_{j,j'}, g_{j,j'}) = \mathsf{Spooky\text{-}Mult}(\mathsf{pk}_j, \mathsf{pk}_{j'}, c_j^{(u)}, c_{j'}^{(v)})$, for every $j, j' \in [n]$. We then set

$$c_j^{(w)} = \mathsf{Eval}_{\mathsf{pk}_j}(\oplus, f_{j,1}, \dots, f_{j,n}, g_{1,j}, \dots, g_{n,j}).$$

We obtain that:

$$\begin{aligned}
\sum_{j\in[n]} \mathsf{Dec}_{\mathsf{sk}_j}\left(c_j^{(w)}\right) &= \sum_{j\in[n]} \mathsf{Dec}_{\mathsf{sk}_j}\left(\mathsf{Eval}_{\mathsf{pk}_j}(\oplus, x_{j,1}, \dots, x_{j,n}, y_{1,j}, \dots, y_{n,j})\right) \\
&= \sum_{j\in[n]} \sum_{j'\in[n]} \mathsf{Dec}_{\mathsf{sk}_j}(f_{j,j'}) \oplus \mathsf{Dec}_{\mathsf{sk}_j}(g_{j',j}) \\
&= \sum_{j\in[n]} \sum_{j'\in[n]} \mathsf{Dec}_{\mathsf{sk}_j}(c_j^{(u)}) \cdot \mathsf{Dec}_{\mathsf{sk}_j}(c_{j'}^{(v)}) \\
&= \left(\sum_{j\in[n]} \mathsf{Dec}_{\mathsf{sk}_j}(c_j^{(u)})\right) \cdot \left(\sum_{j'\in[n]} \mathsf{Dec}_{\mathsf{sk}_j}(c_{j'}^{(v)})\right) \\
&= b_u \cdot b_v,
\end{aligned}$$

which is the correct value for the wire w (where the fourth equality is due to the Spooky-Mult guarantee). Furthermore, each new ciphertext was obtained by applying two hops of homomorphic operations. □

6 Applications of Spooky Encryption

We describe below both positive and negative applications of spooky encryption. We begin in Section 6.1 by showing how to use it to construct a counter-example to the transformation of Aiello et al. [ABOR00] from multi-prover to single-prover protocols. Then, in Section 6.2 we show how to use it to get two-round secure computation protocols. Finally, in Section 6.3 we describe how we obtain a function secret sharing scheme [BGI15] for all functions.

6.1 Counter Example for the [ABOR00] Heuristic

Building on [DLN+04], we show that AFS-2-spooky encryption gives a counter-example to a natural method proposed by Aiello et al. [ABOR00] for building succinct arguments for NP, resolving a question posed by [DLN+04]. The suggestion of Aiello *et al.* [ABOR00] was to take any multi-prover interactive proof-system (MIP) and to use that proof-system using only a *single* prover by sending all of the MIP queries encrypted under independents keys of a homomorphic encryption scheme.[9] The fact that the scheme is homomorphic allows the honest prover to answer the different queries (homomorphically) and the intuition was that the use of different keys means that only local homomorphisms are possible. Dwork *et al.* [DLN+04] questioned this intuition and raised the question of whether there exist spooky encryption schemes that allow for other kinds of attacks which can break the soundness of the [ABOR00] protocol. We show that this is indeed the case: there exists an MIP (suggested by [DLN+04]) which, when combined with any AFS-2-spooky encryption scheme via the [ABOR00] transformation, yields an insecure protocol. The MIP that we use is based on a PCP for 3-coloring due to Petrank [Pet94]:

Theorem 10. *[Pet94] There exists a universal constant $\varepsilon > 0$ such that distinguishing between the following two types of graphs is* NP *complete:*

- *G is 3-colorable.*
- *Every 3-coloring of G has at least ε fraction of monochromatic edges.*

This PCP leads to the following natural MIP protocol between a verifier V and two non-communicating provers P_1 and P_2 (who, in case G is 3-colorable, also have access to the same 3-coloring of G).

1. V chooses a random edge $(u, v) \in E$, then with probability $1/3$ it sets $q_1 = u$ and $q_2 = v$, with probability $1/3$ it sets $q_1 = u$ and $q_2 = u$, and with probability $1/3$ it sets $q_1 = v$ and $q_2 = v$. V sends q_1 to P_1 and q_2 to P_2.
2. Each P_i sends the color $a_i \in \{0, 1, 2\}$ of the vertex q_i (encoded as two bits).
3. V accepts if $q_1 = q_2$ and $a_1 = a_2$, or if $q_1 \neq q_2$ and $a_1 \neq a_2$.

Completeness is immediate. For soundness, let G be a graph such that every 3-coloring miscolors more than an ε fraction of the edges, where ε is the constant from Theorem 10. Fix prover strategies P_1 and P_2 that make the verifier accept G with probability $1 - \varepsilon/6$. As usual, we may assume without loss of generality that P_1 and P_2 are deterministic (by fixing their coins to those that maximize the acceptance probability of G). Hence they define colorings $P_1, P_2 : V(G) \to \{0, 1, 2\}$ of the graph.

For at least a $1-\varepsilon/2$ fraction of edges (u, v) it must hold that $P_1(u) = P_2(u)$ *and* $P_1(v) = P_2(v)$ (else the verifier rejects with probability greater than $\varepsilon/6$). Combining this fact with our assumption on every 3-coloring of G (and in particular P_1) we obtain that for at least an $\varepsilon/2$ fraction of edges $P_1(u) = P_2(v)$. Thus, with probability $\varepsilon/6$ the verifier rejects.

Insecurity of the 3-coloring MIP. Composed the foregoing MIP with any AFS-2-spooky encryption scheme yields an insecure protocol. More specifically, the cheating prover is given ciphertexts $c_1 = \mathsf{Enc}_{\mathsf{pk}_1}(q_1)$ and $c_2 = \mathsf{Enc}_{\mathsf{pk}_2}(q_2)$. Loosely speaking, using the spooky evaluation algorithm it

[9] Actually, the original suggestion in [ABOR00] was to use a PCP (rather than an MIP). Dwork *et al.* [DLN+04] show that using PCPs is not sound and raise the question of whether soundness can be obtained by replacing the PCP with an MIP.

can produce ciphertexts $\mathsf{Enc}_{\mathsf{pk}_1}(a_1)$ and $\mathsf{Enc}_{\mathsf{pk}_2}(a_2)$ for bits $a_1, a_2 \in \{0,1\}$ such that $a_1 = a_2$ if and only if $u = v$. It sends as its answers to V the ciphertext $\big(\mathsf{Enc}_{\mathsf{pk}_1}(0), \mathsf{Enc}_{\mathsf{pk}_1}(a_1)\big)$ as its answer to the first query and $\big(\mathsf{Enc}_{\mathsf{pk}_1}(0), \mathsf{Enc}_{\mathsf{pk}_1}(a_2)\big)$ as its answer to the second query (the extra encryption of 0 is used simply because the verifier expects an answer with 2 bits).

Now, if the verifier choose $q_1 = u$ and $q_2 = v$ (corresponding to the first of the three possibilities) then $q_1 \neq q_2$ and so $a_1 \neq a_2$ and the verifier accepts. Otherwise, (i.e. if $q_1 = q_2$) then we have that $a_1 = a_2$ and again the verifier accepts. Hence, we have shown a strategy that breaks the soundness of the scheme with probability 1.

Remark 6.1. *The above* MIP *has a large (yet constant) soundness error. We can obtain an* MIP *with soundness error (say) $1/2$ by repeating the base* MIP *$O(1/\varepsilon)$ times (using $O(1/\varepsilon)$ provers). Applying the Aiello et al. [ABOR00] transformation to the resulting* MIP *still yields an insecure protocol since the cheating prover can attack each base* MIP *separately.*

6.2 2-Round MPC from AFS-Spooky Encryption

AFS-spooky encryption seems to be a useful tool for minimally-interactive multi-party protocols: it lets each party broadcast an encryption of its input under its own key, then everyone individually performs the AFS-spooky evaluation locally, and each party can locally decrypt and recover a share of the output (relative to an additive n-out-of-n secret-sharing scheme). Finally another round of communication can be used to recover the secret from all the shares. Implementing this the approach requires attention to some details, such as ensuring that the spooky evaluation is deterministic (so that all the parties arrive at the same sharing) and making the shares simulatable (which can be done by having each party distribute a random additive sharing of 0 in the first round and then adding all their received shares to their spooky generated share before broadcasting it in the second round).

A different (but similar) avenue for implementing 2-round MPC, is by reducing AFS-spooky encryption to *multi-key FHE with threshold decryption (TMFHE)*. This primitive was recently formalized by Mukherjee and Wichs [MW16], who showed how to use it to generically construct 2-round MPC. Just like spooky encryption, a TMFHE scheme can homomorphically process n ciphertexts $c_1, \dots, c_n$, encrypting values $x_1, \dots, x_n$ under independent public keys $\mathsf{pk}_1, \dots, \mathsf{pk}_n$, producing for any function f a ciphertext $c^* = \mathsf{Eval}(f, (\mathsf{pk}_1, c_1), \dots, (\mathsf{pk}_n, c_n))$. The ciphertexts c^* cannot be decrypted by any single secret keys sk_i individually, but each party can compute a partial decryption $y_i = \mathsf{PartDec}_{\mathsf{sk}_i}(c^*)$ and these y's can be combined to get $y = \mathsf{FinDec}(y_1, \dots, y_n) = f(x_1, \dots, x_n)$. For security, Mukherjee and Wichs required that for each individual i, the partial decryption y_i can be simulated given the evaluated ciphertext c^*, the final output y and the secret key sk_j for $j \neq i$ (see [MW16] for formal definitions).

We observe that an AFS-spooky encryption with perfect correctness immediately yields a TMFHE scheme. The homomorphic evaluation procedure Eval of the TMFHE runs the $\mathsf{Spooky\text{-}Eval}$ procedure of the AFS-spooky encryption and sets $c^* = (c'_1, \dots, c'_n)$ to be the resulting ciphertexts. The partial decryption procedure $\mathsf{PartDec}_{\mathsf{sk}_i}(c^*)$ outputs $y_i = \mathsf{Dec}_{\mathsf{sk}_i}(c'_i)$ and the combination procedure $\mathsf{FinDec}(y_1, \dots, y_n)$ outputs $y = \bigoplus_{i=1}^{n} y_i$. For security, we observe that each partial decryption y_i can be simulated given $c^* = (c'_1, \dots, c'_n)$, y and sk_j for $j \neq i$ by computing $y_j = \mathsf{Dec}_{\mathsf{sk}_j}(c'_j)$ and setting $y_i = y \oplus (\bigoplus_{j \neq i} y_j)$. [10] This proves the following theorem.

[10]We note that imperfect correctness of the AFS-spooky scheme will translate into a security problem for the TMFHE scheme, as the simulated y_i will have a different distribution than the real ones.

Theorem 11. *An AFS-spooky encryption scheme with perfect correctness implies a multi-key FHE with threshold decryption (TMFHE).*

Using the above theorem and the results of [MW16] which constructs a 2-round MPC from TMFHE, we get the following corollaries.

Corollary 12. *Assuming the existence of a weak AFS-spooky encryption scheme:*

- *There exists a 2-round* MPC *protocol with semi-honest security. If the encryption scheme is in the plain model then so is the* MPC *protocol and if the encryption scheme requires a CRS then so does the* MPC *protocol.*
- *Furthermore, assuming the existence of NIZKs in the CRS model, there exists a 2-round* MPC *protocol with malicious security in the CRS model.*

Combining this with our construction of AFS-spooky encryption without a CRS from iO, we get the first construction of a 2-round semi-honest MPC protocol in the plain model.

Corollary 13. *Assume existence of (1)* piO *and (2) a lossy encryption scheme that is homomorphic for all single-bit to single-bit functions with perfect malicious circuit privacy. Then, there exists a 2-round* MPC *protocol with semi-honest security in the plain model.*

6.3 Function Secret Sharing

Function secret sharing (FSS), recently introduced by Boyle, Gilboa and Ishai, allows a dealer to split a function f into k succinctly described functions $\hat{f}_1, \ldots, \hat{f}_k$ such that (1) any strict subset of the $\hat{f}_i$'s reveals nothing about f and (2) for any x it holds that the values $\hat{f}_1(x), \ldots, \hat{f}_k(x)$ are an additive secret sharing of $f(x)$. Boyle *et al.* gave constructions under standard assumptions for certain restricted families of functions and a general construction for *any* poly-size circuit, based on piO. We show how to construct such a general FSS scheme given any AFS-spooky encryption scheme. In particular, we obtain a leveled FSS scheme assuming only LWE.

To construct such an FSS scheme, the dealer first generates a k-out-of-k secret sharing $f_1, \ldots, f_k$ of the *description* of the function f. The dealer also generates k key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)_{i\in[k]}$ for the AFS spooky scheme and publishes $\hat{f}_i \overset{\text{def}}{=} \big(\mathsf{sk}_i, \mathsf{pk}_1, \ldots, \mathsf{pk}_k, \mathsf{Enc}_{\mathsf{pk}_1}(f_1), \ldots, \mathsf{Enc}_{\mathsf{pk}_k}(f_k)\big)$ as the i^{th} share. Assuming the scheme is semantically secure, any strict subset of the $\hat{f}_i$'s hides the original function f (upto its description length).

For the FSS functionality, given an input x we can consider the circuit C_x that takes as input k shares of a function f, adds them up and applies the resulting function to the input x (which, say, is hardwired). To evaluate $\hat{f}_i$ on x, we run the spooky evaluation algorithm, which we assume wlog is deterministic, on $\mathsf{Enc}_{\mathsf{pk}_1}(f_1), \ldots, \mathsf{Enc}_{\mathsf{pk}_k}(f_k)$ with respect to the circuit C_x. Thus, given each $\hat{f}_i$ separately, we can generate the same ciphertexts $c_1, \ldots, c_k$ which are encryptions of an additive secret sharing of $f(x)$. Each function $\hat{f}_i$ can then be used to decrypt c_i and publish its share of $f(x)$.

A De-Centralized View. We remark that the above construction can be viewed as a de-centralized FSS. More specifically, we may have some k (not necessarily secret or functional) shares $f_1, \ldots, f_k$ of a function f, where each share is owned by a different player. Player i can generate a key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ and broadcast $(\mathsf{pk}_i, \mathsf{Enc}_{\mathsf{pk}_i}(f_i))$ to all other players. Using our scheme, after learning the input x, the players can (non-interactively) generate an additive secret sharing of $f(x)$.

7 Spooky-Free Encryption

We turn now to study *spooky-free* encryption, i.e. an encryption scheme that ensures that no spooky relations can be realized by an adversary. The formal definition roughly states that any correlation that an attacker can induce between the original messages $(m_1, \ldots, m_n)$ and "tampered messages" $(m'_1, \ldots, m'_n)$, can be simulated by a "local simulator" that produces m'_i only as a function of m_i (and some shared randomness).

Definition 7.1 (Spooky-Free Encryption)**.** *An encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* spooky-free *if for every PPT adversary* $\mathcal{A}$ *there exists a PPT simulator* $\mathcal{S}$ *such that for all PPT message distributions* $\mathcal{D}$*, the two distributions* $\mathbf{REAL}_{\mathcal{D},\mathcal{A}}(\kappa)$ *and* $\mathbf{SIM}_{\mathcal{D},\mathcal{S}}(\kappa)$ *specified below are computationally indistinguishable:*

$\underline{\mathbf{REAL}_{\mathcal{D},\mathcal{A}}(\kappa)}$:
1. *Sample* $(m_1, \ldots, m_n, \alpha) \leftarrow \mathcal{D}(1^\kappa)$*;* // α *is auxiliary information*
2. *Choose* $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\kappa)$ *and set* $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}_i}(m_i)$ *for* $i = 1, \ldots, n$*;*
3. *Let* $(c'_1, \ldots, c'_n) \leftarrow \mathcal{A}(\mathsf{pk}_1, \ldots, \mathsf{pk}_n, c_1, \ldots, c_n)$*;*
4. *Set* $m'_i = \mathsf{Dec}_{\mathsf{sk}_i}(c_i)$ *for* $i = 1, \ldots, n$*;*
5. *Output* $(m_1, \ldots, m_n, m'_1, \ldots, m'_n, \alpha)$*.*

$\underline{\mathbf{SIM}_{\mathcal{D},\mathcal{S}}(\kappa)}$:
1. *Sample* $(m_1, \ldots, m_n, \alpha) \leftarrow \mathcal{D}(1^\kappa)$*;* // α *is auxiliary information*
2. *Sample a random tape* r *and let* $m'_i = \mathcal{S}(1^\kappa, 1^n, i, m_i; r)$ *for* $i = 1, \ldots, n$*;*
3. *Output* $(m_1, \ldots, m_n, m'_1, \ldots, m'_n, \alpha)$*.*

It is not hard to see that spooky-freeness for $n \geq 2$ implies semantic security. As a small subtlety, here the attacker must choose the messages it claims to distinguish before seeing the public-key, since the message sampler $\mathcal{D}$ does not know anything public keys used in the real experiment (we defined it this way, as stronger security was not needed for our delegation application). Of course, this minor difference from standard semantic security is without loss of generality when the message space is polynomial small (e.g., for bit encryption).

Lemma 7.2. *A spooky-free scheme for* $n \geq 2$ *is semantically secure (in the "selective" sense discussed above).*

Proof. Suppose that a scheme $(\mathsf{Enc}, \mathsf{Dec}, \mathsf{Gen})$ is *not semantically secure*, and let B be an attacker than can distinguish $\mathsf{Enc}_{\mathsf{pk}}(x_0)$ from $\mathsf{Enc}_{\mathsf{pk}}(x_1)$. We use B to construct a sampler $\mathcal{D}$ and attacker $\mathcal{A}$ that can fool any simulator $\mathcal{S}$ with non-negligible probability. We assume that $\mathcal{D}$ and $\mathcal{A}$ (and S) know the messages x_0 and x_1 whose encryption B can distinguish.

$\mathcal{D}$ draws at random $m_1 \leftarrow \{x_0, x_1\}$ and sets $m_i := 0$ for $i > 1$. Upon seeing n ciphertexts $c_1, \ldots, c_n$, $\mathcal{A}$ gives c_1 to B, asking him to guess whether it encrypts x_0 or x_1. Let σ be the guess that B makes, then we know that $m_1 = x_\sigma$ with probability $\geq 1/2 + \varepsilon$. $\mathcal{A}$ then sets $c'_i = c_i$ for all $i \neq 2$, and sets c'_2 to be a fresh encryption of x_σ under pk_2.

As we can see, the output of the real experiment has the tuple (m_1, m'_2) distributed as (x_b, x_σ), where b is a random bit and $\sigma = b$ with probability $\geq 1/2 + \varepsilon$. On the other hand, the simulator for the second message m'_2 is only given $m_2 = 0$ as the input, and has to guess σ' s.t., $\Pr[b = \sigma'] \geq 1/2 + \varepsilon$, which is impossible information-theoretically. □

Below we show in Section 7.1 that spooky-free homomorphic encryption is exactly the ingredient needed to instantiate the idea of Aiello et al. [ABOR00] for converting general multi-prover

(MIP) systems into single-prover arguments.[11] Then in Section 7.2 we show that non-malleable encryption is always spooky-free (albeit without any homomorphic capabilities). Finally, in Section 7.3 we construct a spooky-free FHE scheme using a strong security component called succinct non-interactive argument of knowledge (SNARK).[12]

Spooky-Free Encryption with CRS. Definition 7.1 can be naturally extended to the common-reference-string model. We use this relaxation in Section 7.3 to gain somewhat better efficiency (at the price of a slightly harder proof of security). We stress that, unlike the setting of spooky encryption from Section 3, using CRS is not done to gain extra (say, homomorphic) functionality for our spooky-free scheme, and our construction remains spooky-free (but slower) if all the public keys are chosen completely independently.

7.1 Application: Succinct 1-round Arguments for NP

Here we formalize the transformation suggested by Aiello et al. [ABOR00] (and cryptanalyzed by Dwork at al. [DLN+04]), and show that homomorphic spooky-free encryption is the missing ingredient which is needed to realize that transformation.

Definition 7.3 (Multi-Prover Interactive Proofs (MIP) for NP)**.** *An n-prover interactive proof (MIP) for an NP relation R consists of PPT algorithms $(V = (V_1, V_2), P_1, \ldots, P_n)$. Given a statement $(x, w) \in R$, the interactions between V and $P_1, \ldots, P_n$ has the following syntax:*

- *The verifier generates queries $(q_1, \ldots, q_n) \leftarrow V_1(1^\kappa, x)$ and remembers its secret state α.*
- *Each prover is given the query q_i and computes an answer $a_i \leftarrow P_i(1^\kappa, x, w, q_i)$.*
- *The verifier runs $V_2(x, (q_1, \ldots, q_n), (a_1, \ldots, a_n), \alpha)$ and outputs 1 (accept) or 0 (reject).*

We require that the MIP satisfies the following properties:

Completeness: *For any $(x, w) \in R$, the interactions between V and $P_1, \ldots, P_n$ results with V outputting 1.*

Soundness: *For any $x \notin R$ and any (not necessarily efficient) malicious provers $P'_1, \ldots, P'_n$, the probability that V outputs 1 after interacting with $P'_1, \ldots, P'_n$ on x is negligible in κ.*

(Succinctness): *The communication complexity $\sum_i (|q_i| + |a_i|) = \mathsf{poly}(\kappa) \cdot \mathsf{polylog}(|x| + |w|)$.*

It is well known (e.g., by combining results in [BGKW88, BFLS91, FRS94], see also Section 6 or [DLN+04]) that there exists a succinct 2-prover MIP for any language in NP.

Definition 7.4 (One-Round Succinct Argument)**.** *A one-round succinct argument (ORSA) for an NP relation R consists of PPT algorithms (Gen, Ver, Prove). We require that the argument satisfies the following properties:*

[11] An alternate route for instantiating the [ABOR00] idea due to [KRR13, KRR14] is to use special types of MIP, which satisfy a stronger soundness condition, together with any (possibly spooky) homomorphic encryption scheme.

[12] Of course, this construction does not give any new one-round delegation schemes, since SNARKs trivially imply the existence of such a scheme directly (i.e., without building spooky-free encryption). Still, if better constructions of spooky-free FHE are found, they would immediately imply new delegation schemes for NP.

Completeness: *For all* $(x, w) \in R$,

$$\Pr[\mathsf{Ver}_{\mathsf{sk}}(x, \pi) = 1 : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa, x), \pi \leftarrow \mathsf{Prove}_{\mathsf{pk}}(x, w)] = 1.$$

Selective Soundness: *For any PPT adversary* P*, any polynomial* $p(\cdot)$ *and any ensemble* $\{x_\kappa\}$ *with* $|x_\kappa| = p(\kappa)$ *and* $x_\kappa \notin R$ *we have:*

$$\Pr[\mathsf{Ver}_{\mathsf{sk}}(x, \pi) = 1 \ : \ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa, x), \pi \leftarrow P(1^\kappa, \mathsf{pk})] = \mathsf{negl}(\kappa).$$

Succinctness: *We say that the scheme is succinct if* $|\mathsf{pk}| + |\mathsf{sk}| + |\pi| = \mathsf{poly}(\kappa) \cdot \mathsf{polylog}(|x| + |w|)$.

The Aiello et al. construction transforms a succinct MIP Π for a relation R into a candidate succinct one-round argument $\hat{\Pi}$ for R using an FHE scheme $(\mathsf{FHE.Gen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval})$. The idea is to take the different queries that the MIP verifier sends to its prover, encrypt them under different FHE keys, and send to the prover. The prover can then execute the MIP provers homomorphically on the encrypted inputs, returning the encrypted answers to the verifier, who decrypts them and checks. As we show below, spooky-freeness is exactly the condition that we need to ensure that the single prover cannot relate the different ciphertexts in any meaningful way.

For the sake of improved efficiency and potential generality, we will allow several instances of the FHE schemes to use a common reference string $\mathsf{CRS} \leftarrow \mathsf{CRSGen}(1^\kappa)$. The transformation is given below:

- The verifier generates queries $(q_1, \ldots, q_n) \leftarrow V_1(1^\kappa, x)$, and remembers secret state α.
- The verifier generates a $\mathsf{CRS} \leftarrow \mathsf{CRSGen}(1^\kappa)$, and n independent (modulo shared CRS) key pairs $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{FHE.Gen}(\mathsf{CRS})$.
- The verifier computes $c_i \leftarrow \mathsf{FHE.Enc}(\mathsf{pk_i}, \mathsf{q_i})$ and sends $(\mathsf{CRS}, \mathsf{pk}_1, \ldots, \mathsf{pk}_n, c_1, \ldots, c_n)$ to the prover.
- Inside each ciphertext c_i, the prover implements the $\mathsf{FHE.Eval}$ homomorphic procedure using the i-th MIP prover's algorithm P_i (and possibly CRS), generating new ciphertext c'_i.
- The verifier gets n ciphertexts $c'_1 \ldots c'_n$, and decrypts them into n plaintexts $a_i \leftarrow \mathsf{Dec}_i(c'_i)$.
- The verifier runs the original MIP verifier V_2 on inputs $(q_1, \ldots, q_n, a_1, \ldots a_n, \alpha)$ and outputs its decision.

Theorem 14. *If the FHE scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is spooky-free and* Π *is a succinct* MIP *for a language* R*, then* $\hat{\Pi}$ *is a succinct one-round argument for* R*.*

Proof. Completeness is obvious, and succinctness follows immediately from the succinctness of the MIP argument. Hence, we turn to selective soundness.

Given a malicious prover P for the one-round argument, we define the message sampler $\mathcal{D}$ and an adversary $\mathcal{A}$ for the spooky-free game, as follows. $\mathcal{D}$ runs V_1 and sets $(m_1, \ldots, m_n) = (q_1, \ldots, q_n)$, where q_i are the questions prepared by V_1. It also keeps the secret side information α.

To define $\mathcal{A}$, we continue the execution above, and generate n ciphertexts c_i, as expected, sending them all to P (together with public keys and optional CRS). P mauls them to $c'_1, \ldots, c'_n$, and this is defined to be the output of $\mathcal{A}$. We also let $a_i = \mathsf{Dec}_i(c'_i)$.

By the spooky-freeness of FHE, we know that the joint distribution of $(q_1, \dots, q_n, a_1, \dots, a_n, \alpha)$ can be simulated by the spooky-free simulator $\mathcal{S}$, who generates a_i only as the function of q_i (and shared randomness). But then the execution with the simulator defines n valid non-communicating provers P_i for the MIP game, whose soundness implies that the soundness of the one-round succinct argument. □

Remark 7.5. *We note that the succinct one-prover argument that we construct inherits many extra properties that the* MIP *system may have. For example, if queries q_i can be generated independent of the instance x in the* MIP *proof, the same will be true for the one-round argument (which now becomes a "designated-verifier, non-adaptive SNARG"). Similarly, if the* MIP *is also a proof of knowledge, then so will be our argument system.*

7.2 Non-Malleable Encryption is Spooky-Free

We show that any non-malleable encryption is spooky-free. Actually, this is already true for one of the weakest notions of non-malleable encryption, where the adversary can access the decryption oracle just once.

Definition 7.6 (1-Bounded Non-Malleability)**.** *A public key encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* 1*-non-malleable (1-NM) if for every PPT $\mathcal{A}$ we have* $|\Pr[\mathbf{NMGame}_{\mathcal{A},\mathbf{0}}(\kappa) = 1] - \Pr[\mathbf{NMGame}_{\mathcal{A},\mathbf{1}}(\kappa) = 1]| \leq \mathsf{negl}(\kappa)$ *where we define the game* $\mathbf{NMGame}_{\mathcal{A},\mathbf{b}}(\kappa)$ *as follows:*

- *Generate* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa)$ *and give* pk *to* $\mathcal{A}$*.*
- $\mathcal{A}$ *chooses two messages m_0, m_1 with $|m_0| = |m_1|$.*
- *Compute $c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_b)$ and give c to $\mathcal{A}$.*
- $\mathcal{A}$ *chooses c' with $c' \neq c$ and gets $\mathsf{Dec}_{\mathsf{sk}}(c')$.*
- $\mathcal{A}$ *outputs a bit b' which is the output of the game.*

Theorem 15. *Any 1-non-malleable public-key encryption scheme is spooky-free.*

Proof. For any adversary $\mathcal{A}$ we define a simulator $\mathcal{S}(1^\kappa, 1^n, i, m; r)$ as follows:

- Use the randomness r to sample the following values: for $j = 1, \dots, n$ choose $(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{Gen}(1^\kappa)$, $c_j \leftarrow \mathsf{Enc}_{\mathsf{pk}_j}(0)$ and set $(c'_1, \dots, c'_n) \leftarrow \mathcal{A}(\mathsf{pk}_1, \dots, \mathsf{pk}_n, c_1, \dots, c_n)$.
- If $c'_i = c_i$, then output m, else output $\mathsf{Dec}_{\mathsf{sk}_i}(c_i)$.

Let $\mathcal{D}$ be any message sampler. We define a sequence of intermediate hybrids to show the indistinguishability of $\mathbf{REAL}_{\mathcal{D},\mathcal{A}}(\kappa)$ and $\mathbf{SIM}_{\mathcal{D},\mathcal{S}}(\kappa)$. Let Hybrid 0 be $\mathbf{REAL}_{\mathcal{D},\mathcal{A}}(\kappa)$. Let Hybrid i be the same as Hybrid 0 except that for $j \leq i$:

- Instead of setting $c_j \leftarrow \mathsf{Enc}_{\mathsf{pk}_j}(m_j)$ we set $c_j \leftarrow \mathsf{Enc}_{\mathsf{pk}_j}(0)$.
- If $c'_j = c_j$ then instead of setting $m'_j = \mathsf{Dec}_{\mathsf{sk}_j}(c_j)$ we set $m'_j = m_j$.

It's easy to show that Hybrid $i-1$ is indistinguishable from Hybrid i by the non-malleable security with public key pk_i. In particular, the only difference between these hybrids is whether $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}_i}(m_i)$ or $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}_i}(0)$ (in both hybrids, if $c'_i = c_i$, we set $m'_i = m_i$). Furthermore, the secret key sk_i is only used to decrypt a single ciphertext $c'_i \neq c_i$ during the course of the hybrids.

Therefore Hybrid 0, which is $\mathbf{REAL}_{\mathcal{D},\mathcal{A}}(\kappa)$, is indistinguishable from Hybrid n, which is $\mathbf{SIM}_{\mathcal{D},\mathcal{S}}(\kappa)$. This proves the theorem. □

It is also well known that 1-NM encryption can be generically built from any semantically secure encryption scheme [CHH$^+$07]. Thus,

Corollary 16. *Assuming the existence of semantically secure public-key encryption, there exists spooky-free encryption.*

7.3 Homomorphic Spooky-Free Encryption

We now show how to construct a spooky-free fully homomorphic encryption scheme, using three strong components: a fully homomorphic encryption scheme (FHE [RAD78, Gen09]), a true-simulation-extractable non-interactive zero-knowledge argument (tSE-NIZK [DHLW10]), and a succinct non-interactive argument of knowledge (SNARK [GW11, BSW12]).

As a high level, we use FHE to achieve the required homomorphism, tSE-NIZK arguments to ensure "non-malleability across keys" (which, as we saw, implies "spooky-freeness" in the non-homomorphic setting), and SNARKs to ensure that any ciphertext mauling due to the added homomorphism can be "traced back" to mauling the original ciphertexts, which we know is impossible due to the non-malleability mentioned above.

We formally state our construction below, but mention that a special instantiation of our scheme[13] is essentially identical to a scheme proposed by Boneh, Segev and Waters [BSW12] in the context of *targeted malleability*. In the latter problem, the goal is to have a (single key) homomorphic encryption scheme which only allows restricted types of homomorphic operations. Although this is a different goal than spooky free encryption, it turns out that essentially the same construction can serve both goals. Our (more general) construction also works for the application of [BSW12], and we use a slightly more modular presentation.[14]

For simplicity, we describe a construction which is only 1-hop and which only allows homomorphic evaluations over a single (multi-bit) ciphertext. However, it is easy to extend this to multiple hops and homomorphic evaluations over multiple ciphertexts, along the lines described by [BSW12]. For better efficiency, our spooky-free FHE shares the same CRS for both the tSE-NIZKs and SNARKs. This makes arguing security a bit harder, but still possible for our scheme. In particular, our scheme remains a spooky-free FHE if each encryption scheme chooses its own CRS for tSE-NIZKs and SNARKs.

7.3.1 tSE-NIZK arguments and SNARKs

We use the definition of *true-Simulation-Extractable Non-Interactive Zero-Knowledge (*$\mathsf{tSE\text{-}NIZK}$*)* argument of Dodis et al. [DHLW10], which captures various notions of simulation-soundness [Sah99]

[13]Specifically, of the tSE-NIZK component using an FHE scheme and an adaptive simulation-sound NIZK. We note, though, that using our simpler construction, no FHE is needed to implement the tSE-NIZK, and more efficient instantiations are possible [DHLW10].

[14]E.g., we avoid explicitly using the Naor-Yung double encryption trick [NY90], burying it into low-level implementations of tSE-NIZKs.

and extractability [SP92, CLOS02, PR08a, PR08b, Gro06] and lets us abstract away some of the details of the construction. Intuitively, $\mathsf{tSE\text{-}NIZK}$ strengthens the soundness condition by requiring an *extractor* Ext, which (using a CRS-related trapdoor) is able to extracts a witness w^* from any any convincing proof, even if the prover previously saw many *simulated proofs* for arbitrary *true* statements.

Definition 7.7 (tSE-NIZK)**.** *A* $\mathsf{tSE\text{-}NIZK}$ argument *for an* NP *relation* R *consists of three algorithms* $(\mathsf{NIZK.CRSGen}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Ver})$ *with syntax:*

- $(\mathsf{CRS}, \mathsf{tk}, \mathsf{ek}) \leftarrow \mathsf{NIZK.CRSGen}(1^\kappa)$*: Creates a common reference string (CRS)* CRS*, an equivocation trapdoor key* tk*, and extraction key* ek *to the CRS.*
- $\pi \leftarrow \mathsf{NIZK.Prove}_{\mathsf{CRS}}(x, w)$*: Assuming* $R(x, w) = 1$*, creates a proof* π *certifying this fact.*
- $0/1 \leftarrow \mathsf{NIZK.Ver}_{\mathsf{CRS}}(x, \pi)$*: Verifies whether or not the proof* π *is correct.*

We require that the following three properties hold (we omit CRS *subscript for brevity):*

Completeness: *For any* $(x, w) \in R$*, if* $(\mathsf{CRS}, \mathsf{tk}, \mathsf{ek}) \leftarrow \mathsf{NIZK.CRSGen}(1^\kappa)$, $\pi \leftarrow \mathsf{NIZK}.\mathit{Prove}(x, w)$*, then* $\mathsf{NIZK.Ver}(x, \pi) = 1$*.*

Composable Zero-Knowledge: *There exists PPT simulator* $\mathsf{NIZK.Sim}$ *such that, for any PPT adversary* $\mathcal{A}$ *we have* $\left|\Pr[\mathcal{A} \textit{ wins }] - \frac{1}{2}\right| \leq \mathsf{negl}(\kappa)$ *in the following game:*

- *The challenger samples* $(\mathsf{CRS}, \mathsf{tk}, \mathsf{ek}) \leftarrow \mathsf{NIZK.CRSGen}(1^\kappa)$ *and gives* $(\mathsf{CRS}, \mathsf{tk})$ *to* $\mathcal{A}$*.*
- *The adversary* $\mathcal{A}$ *chooses* $(x, w) \in R$ *and gives these to the challenger.*
- *Challenger rejects if* $(x, w) \notin R$*; otherwise, it samples* $\pi_0 \leftarrow \mathsf{NIZK.Prove}(x, w), \pi_1 \leftarrow \mathsf{NIZK.Sim}(x, \mathsf{tk}), b \leftarrow \{0, 1\}$ *and gives* π_b *to* $\mathcal{A}$*.*
- *The adversary* $\mathcal{A}$ *outputs a bit* $\tilde{b}$ *and wins if* $\tilde{b} = b$*.*

True-Simulation-Extractability: *There exists a PPT algorithm* $\mathsf{NIZK.Ext}(x, \pi, \mathsf{ek})$ *such that for all* P^* *we have* $\Pr[P^* \textit{ wins}] \leq \mathsf{negl}(\kappa)$ *in the following game:*

1. **Key Generation:** *The challenger runs* $(\mathsf{CRS}, \mathsf{tk}, \mathsf{ek}) \leftarrow \mathsf{NIZK}.\mathit{CRSGen}(1^\kappa)$ *and gives* CRS *to* P^**.*
2. **Simulation queries:** P^* *is given access to the* true-simulation oracle*: a query to this oracle consists of a pair* (x, w)*; the oracle checks if* $(x, w) \in R$*; if true, it ignores* w*, and outputs a simulated argument* $\mathsf{NIZK.Sim}(x, \mathsf{tk})$*, and otherwise outputs* $\perp$*.*
3. **Adversary Output:** P^* *outputs a tuple* (x^*, π^*)*.*
4. **Extraction:** *The challenger runs* $w^* \leftarrow \mathsf{NIZK.Ext}(x^*, \pi^*, \mathsf{ek})$*.*
5. P^* *wins if* $\mathsf{NIZK.Ver}(x^*, \pi^*) = 1$ *and the value* x^* *was not part of a true-simulator query.*

Dodis et al. [DHLW10] also showed several constructions of $\mathsf{tSE\text{-}NIZK}$s for all of NP from traditional encryption and NIZK schemes.

In contrast, SNARKs are a much stronger knowledge-based assumption, which cannot be black-box constructed from any falsifiable assumption [GW11] (with adaptive soundness).

Definition 7.8 (SNARK). *A Succinct Non-interactive ARgument or Knowledge (SNARK) for an* NP *relation* R *consists of PPT algorithms* $(\mathsf{SNARK.CRSGen}, \mathsf{SNARK.Prove}, \mathsf{SNARK.Ver})$ *where*

- $\mathsf{CRS} \leftarrow \mathsf{SNARK.CRSGen}(1^\kappa)$*: Creates a common reference string (CRS)* CRS*.*
- $\pi \leftarrow \mathsf{SNARK.Prove}_{\mathsf{CRS}}(x, w)$*: Assuming* $R(x, w) = 1$*, creates a proof* π *certifying this fact.*
- $0/1 \leftarrow \mathsf{SNARK.Ver}_{\mathsf{CRS}}(x, \pi)$*: Verifies whether or not the proof* π *is correct.*

We require that the following properties (we omit CRS *subscript for brevity):*

Completeness: *For any* $(x, w) \in R$*:*

$$\Pr[\mathsf{SNARK.Ver}(x, \pi) = 1 \ : \ \mathsf{CRS} \leftarrow \mathsf{Gen}(1^\kappa), \pi \leftarrow \mathsf{SNARK.Prove}(x, w)] = 1.$$

(Adaptive) Knowledge Extraction: *for every PPT algorithm* P *there here exists a PPT extractor* $\mathsf{SNARK.Ext}_P$ *such that*

$$\Pr\left[\begin{array}{c} \mathsf{SNARK.Ver}(x, \pi) = 1 \\ \textit{and } R(x, w) = 0 \end{array} \middle| \begin{array}{c} \mathsf{CRS} \leftarrow \mathsf{SNARK.CRSGen}(1^\kappa) \\ \pi \leftarrow P(1^\kappa, \mathsf{CRS}; r) \\ w \leftarrow \mathsf{SNARK.Ext}_P(1^\kappa, \mathsf{CRS}; r) \end{array}\right] \leq \mathsf{negl}(\kappa).$$

Succinctness: *The proof is short:* $|\pi| = \mathsf{poly}(\kappa) \cdot \mathsf{polylog}(|x| + |w|)$*.*

7.3.2 Constructing of Spooky-Free FHE

We set up tSE-NIZK for the relation $R_{NIZK} = \{(x = (\mathsf{pk}, c), w = (m, r)) \ : \ c = \mathsf{FHE.Enc}_{\mathsf{pk}}(m; r)\}$, and SNARK for the relation

$$R_{SNARK} = \left\{ \begin{pmatrix} x = (\mathsf{CRS}, \mathsf{pk}, c'), \\ w = (f, c, \pi) \end{pmatrix} \quad : \quad \begin{array}{c} c' = \mathsf{FHE.Eval}_{\mathsf{pk}}(f, c), \\ \mathsf{NIZK.Ver}(\mathsf{CRS}, (\mathsf{pk}, c), \pi) = 1 \end{array} \right\}.$$

We construct a spooky-free FHE scheme ($\mathsf{SF.CRSGen}$, $\mathsf{SF.Gen}$, $\mathsf{SF.Enc}$, $\mathsf{SF.Dec}$, $\mathsf{SF.Eval}$) as follows:

- $\mathsf{CRS} \leftarrow \mathsf{SF.CRSGen}(1^\kappa)$: Run $(\mathsf{CRS}_1, \mathsf{tk}, \mathsf{ek}) \leftarrow \mathsf{NIZK.CRSGen}(1^\kappa)$ and $\mathsf{CRS}_2 \leftarrow \mathsf{SNARK.CRSGen}(1^\kappa)$. Set $\mathsf{CRS} = (\mathsf{CRS}_1, \mathsf{CRS}_2)$.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{SF.Gen}(1^\kappa)$: Run $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.Gen}(1^\kappa)$.
- $C \leftarrow \mathsf{SF.Enc}_{\mathsf{pk}}(m)$: Run $c \leftarrow \mathsf{FHE.Enc}_{\mathsf{pk}}(\mathsf{m}; \mathsf{r})$ and $\pi_1 \leftarrow \mathsf{NIZK.Prove}(\mathsf{CRS}_1, (\mathsf{pk}, c), (m, r))$. Output $C = (0, c, \pi_1)$.
- $C' \leftarrow \mathsf{SF.Eval}(f, C)$: Parse $C = (0, c, \pi_1)$. Compute $c' = \mathsf{FHE.Eval}_{\mathsf{pk}}(\mathsf{f}, \mathsf{c})$, $\pi_2 \leftarrow \mathsf{SNARK.Prove}(\mathsf{CRS}_2, (\mathsf{CRS}_1, \mathsf{pk}, c'), (f, c, \pi_1))$. Output $C' = (1, c', \pi_2)$.
- $m \leftarrow \mathsf{SF.Dec}_{\mathsf{sk}}(C)$: Parse $C = (b, c, \pi)$.
  If $b = 0$ and $\mathsf{NIZK.Ver}(\mathsf{CRS}_1, (\mathsf{pk}, c), \pi) = 0$ then output $\perp$.
  If $b = 1$ and $\mathsf{SNARK.Ver}(\mathsf{CRS}_2, (\mathsf{CRS}_1, \mathsf{pk}, c), \pi) = 0$ then output $\perp$.
  Else output $\mathsf{Dec}_{\mathsf{sk}}(c)$.

Theorem 17. *Given a secure FHE, tSE-NIZK and SNARK schemes, the above construction is a spooky-free FHE scheme.*

We present a formal proof in Appendix A, here giving a high-level sketch. As we mentioned, spooky-freeness implies semantic security for $n \geq 2$, so we only need to check the compactness of the FHE and then show its spooky-freeness. For the former, it immediately follows from the succinctness of SNARK, which means that the size of the SNARK proof π_2 output by the Eval routine is indeed compact as compared to the size of f.

We next turn to spooky-freeness. The main challenge here is to use various security properties of tSE-NIZK and SNARK to slowly transform the real spooky-free game into an indistinguishable game, where no individual decryption keys sk_i are used, except either: (a) an attacker copied the challenge ciphertext c_i; or (b) the attacker honestly applied the FHE Eval procedure to a challenge ciohertext c_i. Once accomplished, we can use the semantic security of the FHE to replace real encryption of messages m_i with encryptions of 0, after which we can easily define the required simulator $\mathcal{S}$ who will use its message m_i only to cover the two obvious cases (a) and (b) mentioned above.

To do this transition from the real to the simulated game, we use the following high-level hybrids (each of which is usually composed of n smaller hybrids, one for each encryption component):

- **Hybrid** 0. Original real game where real witnesses are uses for all tSE-NIZKs and real decryption keys are used to decrypt.

- **Hybrid** 1. Replace all real proofs π_1 with simulated proofs, using the trapdoor tk. Notice, simulated proofs are used on true statement, and composable zero-knowledge of tSE-NIZKs is used to argue security.

- **Hybrid** 2. Recall, the attacker $\mathcal{A}$ can maul ciphertext $C = (0, c, \pi_1)$ into $C' = (b, c', \pi')$ two ways. (a) By keeping $b = 0$ (thus still using NIZKs), or (b) Making $c = 1$ and using SNARKs. In this second case, we use the knowledge extraction property of SNARKs to argue that the attacker must know a ciphertext c^*, a mauling function f^* and normal NIZK π^* s.t. the message m' encoded by c' is $f^*(m^*)$, where m^* is decryption of c^* and π^* is a valid tSE-NIZK. Thus, on a high level, we effectively reduced the mauling case (b) above to mauling case (a), except we need to apply the extracted function f^* to the result m^* to get m'_i.

- **Hybrid** 3. This formalized here, where instead of decrypting c' in case of mauling type (b) above, we return f^* applied to the decryption of c^* extracted from the attacker.

- **Hybrid** 4. Now that we only need to worry about maulings of type (a) (so we ignore discussing type (b) now, as it is handled similarly, except applying f^* to the result), we are ready to eliminate using the decryption key sk by using the tSE-NIZK extractor. Here we are allowed to use it since the attacker only sees simulated proofs of true statement (as done in **Hybrid 1**). This allows one to eliminate using the secret key sk, except in the obvious case when the attacker copied the actual ciphertext c_i (either in c' in case (a), or c^* in case (b)), which are handled specially.

- **Hybrid** 5. Now we can use the semantic security of FHE to replace the real encryptions of m_i by enryptions of 0.

- The resulting game no longer uses the decryption oracle, removes all information about the messages $(m_1, \ldots, m_n)$, except when the attacker copied the message, or honestly applied some (already extracted) homomorphic evaluation to the ciphertext. This leads to the obvious simulator $\mathcal{S}$, completing the proof.

Acknowledgments

This work was done in part while the authors were visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant #CNS-1523467.

The first author was partially supported by gifts from VMware Labs and Google, and NSF grants 1319051, 1314568, 1065288, 1017471.

The second author was supported in part by the Defense Advanced Research Projects Agency (DARPA) and Army Research Office(ARO) under Contract No. W911NF-15-C-0236.

The third author was partially supported by NSF Frontier "TWC: TTP Option: Frontier: Collaborative: MACS: A Modular Approach to Cloud Security" - CNS1413920 Simons Foundation - Agreement Dated 6-5-12.

The last author was supported in part by NSF grants CNS-1347350, CNS-1314722, CNS-1413964.

References

[ABOR00] William Aiello, Sandeep Bhatt, Rafail Ostrovsky, and S. Raj. Rajagopalan. Fast verification of any remote procedure call: Short witness-indistinguishable one-round proofs for NP. In *ICALP: Annual International Colloquium on Automata, Languages and Programming*, 2000.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 21–31, 1991.

[BGI15] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 337–367. Springer, 2015.

[BGKW88] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 113–131, 1988.

[BLP$^+$13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 575–584. ACM, 2013.

[BSW12] Dan Boneh, Gil Segev, and Brent Waters. Targeted malleability: homomorphic encryption for restricted computations. In Shafi Goldwasser, editor, *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 350–366. ACM, 2012.

[CHH$^+$07] Ronald Cramer, Goichiro Hanaoka, Dennis Hofheinz, Hideki Imai, Eike Kiltz, Rafael Pass, Abhi Shelat, and Vinod Vaikuntanathan. Bounded cca2-secure encryption. In Kaoru Kurosawa, editor, *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, volume 4833 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2007.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 494–503. ACM, 2002.

[CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 468–497, 2015.

[CM15] Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 630–656. Springer, 2015.

[DHLW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 613–631. Springer, 2010.

[DLN$^+$04] Cynthia Dwork, Michael Langberg, Moni Naor, Kobbi Nissim, and Omer Reingold. Succinct proofs for NP and spooky interactions. Unpublished manuscript, available at http://www.cs.bgu.ac.il/~kobbi/papers/spooky_sub_crypto.pdf, 2004.

[FRS94] Lance Fortnow, John Rompel, and Michael Sipser. On the power of multi-prover interactive protocols. *Theor. Comput. Sci.*, 134(2):545–557, 1994.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178. ACM, 2009.

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In *Theory of Cryptography - 11th Theory* *of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 74–94, 2014.

[GHV10] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. i-hop homomorphic encryption and rerandomizable Yao circuits. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 155–172. Springer, 2010. http://eprint.iacr.org/2010/145.

[GI14] Niv Gilboa and Yuval Ishai. Distributed point functions and their applications. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 640–658. Springer, 2014.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229, 1987.

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

[Gro06] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology - ASIACRYPT 2006, 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006, Proceedings*, volume 4284 of *Lecture Notes in Computer Science*, pages 444–459. Springer, 2006.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC*, pages 99–108, 2011.

[HL10] Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols - Techniques and Constructions.* Information Security and Cryptography. Springer, 2010.

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 294–304, 2000.

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology-Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 335–354, 2004.

[KRR13] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. Delegation for bounded space. In *STOC*, pages 565–574, 2013.

[KRR14] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to delegate computations: the power of no-signaling proofs. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 485–494, 2014.

[LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 1219–1234, 2012.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round mutliparty computation via multi-key FHE. In *Eurocrypt 2016, to appear*, 2016. http://eprint.iacr.org/2015/345, accessed Jan 2016.

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA.*, pages 448–457, 2001.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 427–437. ACM, 1990.

[OPP14] Rafail Ostrovsky, Anat Paskin-Cherniavsky, and Beni Paskin-Cherniavsky. Maliciously circuit-private FHE. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 536–553. Springer, 2014. Available from https://eprint.iacr.org/2013/307.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 333–342. ACM, 2009.

[Pet94] Erez Petrank. The hardness of approximation: Gap location. *Computational Complexity*, 4:133–157, 1994.

[PR08a] Rafael Pass and Alon Rosen. Concurrent nonmalleable commitments. *SIAM J. Comput.*, 37(6):1891–1925, 2008.

[PR08b] Rafael Pass and Alon Rosen. Concurrent nonmalleable commitments. *SIAM J. Comput.*, 37(6):1891–1925, 2008.

[RAD78] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–177. Academic Press, 1978.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 543–553. IEEE Computer Society, 1999.

[SP92] Alfredo De Santis and Giuseppe Persiano. Zero-knowledge proofs of knowledge without interaction (extended abstract). In *33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, Pennsylvania, USA, 24-27 October 1992*, pages 427–436. IEEE Computer Society, 1992.

A Proof of Theorem 17

Since spooky-freeness implies semantic security for $n \geq 2$, so we only need to check the compactness of the FHE and then show its spooky-freeness. For the former, it immediately follows from the succinctness of SNARK, which means that the size of the SNARK proof π_2 output by the Eval routine is indeed compact as compared to the size of f.

We next turn to spooky-freeness. For any adversary $\mathcal{A}$ and sampler $\mathcal{D}$, let **Hybrid** 0 denote the distribution $(m_1, \ldots, m_n, m'_1, \ldots, m'_n, \alpha)$ produced by the real game $\mathbf{REAL}_{\mathcal{D},\mathcal{A}}(\kappa)$. Instead of defining simulator $\mathcal{S}$ directly, we will produce a series of hybrids, each indistinguishable from the previous, until we arrive at a hybrid whose code defines a legitimate simulator $\mathcal{S}$.

Hybrids 1.1-1.n. Recall, in **Hybrid** 0 the tSE proofs π_i (for $i = 1$ to n) were generated using the honest tSE-proof $\mathsf{NIZK.Prove}(\mathsf{CRS}_1, (\mathsf{pk}_i, c_i), (m_i, r_i))$. In this hybrid we generate these i proofs running the simulator $\mathsf{Sim}((\mathsf{pk}_i, c_i), \mathsf{tk})$ instead.

By the composable zero-knowledge property of tSE-NIZKs, all these hybrid are computationally indistinguishable from each other, since the composable zero-knowledge property holds even conditioned on the equivocation key tk used to generate simulated proofs. We denote **Hybrid** 1.n by **Hybrid** 1, noting that in this hybrid all the n tSE proofs $\pi_{1,1}, \ldots, \pi_{1,n}$ are generated by the simulator $\mathsf{Sim}(\cdot, \mathsf{tk})$.

Hybrids 2.1-2.n. Before defining **Hybrid** 2.i, we define a malicious prover P_i. This prover, on input CRS_2, simulates the complete run of **Hybrid** 1 (including running $\mathcal{D}$, generating all the public/secret keys $(\mathsf{pk}_i, \mathsf{sk}_i)$, generating $(\mathsf{CRS}_1, \mathsf{ek}, \mathsf{tk})$, etc.), up until the attacker $\mathcal{A}$ mauls n real ciphertexts $(C_1, \ldots, C_n)$ into $(C'_1, \ldots, C'_n)$. P_i then parses $C'_i = (b_i, c'_i, \pi'_i)$. If $b_i = 0$ (no Eval was run), or $b_i = 1$ but the SNARK proof π' does not verify, then P_i aborts. Otherwise, it outputs a valid statement-proof pair $(x_i = (\mathsf{CRS}_1, \mathsf{pk}_i, c'_i), \pi'_i)$ for R_{SNARK}.

By the knowledge extraction property of SNARKs, there is a PPT extractor $\mathsf{Ext}_i = \mathsf{Ext}_{P_i}$ which, when taking the randomness of P_i above, outputs a witness $w_i = (f^*_i, c^*_i, \pi^*_i)$ s.t. $c'_i = \mathsf{FHE.Eval}_{\mathsf{pk}_i}(f^*_i, c^*_i)$ (which implies $\mathsf{Dec}_i(c'_i) = f^*_i(\mathsf{Dec}_i(c^*_i))$) and $\mathsf{NIZK.Ver}(\mathsf{CRS}_1, (\mathsf{pk}_i, c^*_i), \pi^*_i) = 1$.

Having defined P_i and Ext_i above, we can now define **Hybrid** 2.i as follows. As in the definition of P_i, we simulate the complete run of **Hybrid** 1 (including running $\mathcal{D}$, generating all the public/secret keys $(\mathsf{pk}_i, \mathsf{sk}_i)$, generating $(\mathsf{CRS}_1, \mathsf{ek}, \mathsf{tk})$, etc.), up until the attacker $\mathcal{A}$ mauls n real ciphertexts $(C_1, \ldots, C_n)$ into $(C'_1, \ldots, C'_n)$. Also, for $j > i$, we do the same as in **Hybrid** 1. For $j \leq i$, however, we examine the ciphertext $C'_i = (b_i, c'_i, \pi'_i)$. If $b_i = 0$ (no Eval was run), or $b_i = 1$ but the SNARK proof π' does not verify, we again proceed as in **Hybrid** 1. However, if $b_i = 1$ and the SNARK proof π' verifies, we use the knowledge extractor $\mathsf{Ext}_i = \mathsf{Ext}_{P_i}$ which, when taking the randomness of P_i above, outputs a witness $w_i = (f_i^*, c_i^*, \pi_i^*)$ s.t. $c_i' = \mathsf{FHE.Eval}_{\mathsf{pk}_i}(f_i^*, c_i^*)$ (which implies $\mathsf{Dec}_i(c_i') = f_i^*(\mathsf{Dec}_i(c_i^*))$) and $\mathsf{NIZK.Ver}(\mathsf{CRS}_1, (\mathsf{pk}_i, c_i^*), \pi_i^*) = 1$.

Once again, all these hybrids are indistinguishable due to the knowledge extraction property of the SNARK. We denote **Hybrid** 2.n by **Hybrid** 2.

Hybrid 3. Notice, the decryption algorithm $\mathsf{Dec}_i(c_i')$ is only run in **Hybrid** 2 in two situations: (a) the mauled ciphertext $C_i' = (0, c_i', \pi_i')$ and π_i' tSE-verifies; or (b) $C_i' = (1, c_i', \pi_i')$, π_i' SNARK-verifies, and the extractors Ext_i has extracted values (f_i^*, c_i^*, π_i^*) s.t. $c_i' = \mathsf{FHE.Eval}_{\mathsf{pk}_i}(f_i^*, c_i^*)$ (which implies $\mathsf{Dec}_i(c_i') = f_i^*(\mathsf{Dec}_i(c_i^*))$) and $\mathsf{NIZK.Ver}(\mathsf{CRS}_1, (\mathsf{pk}_i, c_i^*), \pi_i^*) = 1$.

In **Hybrid** 3, we change rule (b) above, and instead will output $f_i^*(\mathsf{Dec}_i(c_i^*))$. As we note above, this produces the same output distribution as **Hybrid** 2.

Hybrids 4.1-4.n. Notice, we have the following invariants at the end of **Hybrid** 3. First, the attacker $\mathcal{A}$ only sees simulated proofs of true statements (see **Hybrid** 1). Second, whenever we decrypt some ciphertext using sk_i (either in case (a) or (b) above), we have a valid tSE-proof (either π_i' if $b_i = 0$, or π_i^* if $b_1 = 1$) w.r.t. R_{NIZK}). In **Hybrid** 4.i, instead of calling the actual decryption algorithm Dec_i of such a ciphertext, we will use the extractor Ext from the true-simulation-extractable NIZK, which, using the extraction key ek, will attempt to extract the message from the ciphertext.

One subtlety here is that the extraction procedure might not work when given a ciphertext on which a simulated proof was given. Fortunately, our simulator knows the plaintext m_i, so it will know what to do in such a case. More precisely, we apply the following modified decryption procedure in **Hybrid** 4 (where, for brevity, we ignore the second randomness output r when running Ext, instead only using the message output):

- When $b_i = 0$, NIZK π_i' verifies and $c_i' \neq c_i$, set $m_i' \leftarrow \mathsf{Ext}((\mathsf{pk}_i, c_i'), \mathsf{ek})$;
- When $b_i = 0$, NIZK π_i' verifies and $c_i' = c_i$, set $m_i' = m_i$;
- When $b_1 = 1$, NIZK π_i^* verifies and $c_i^* \neq c_i$, set $m_i' \leftarrow f_i^*(\mathsf{Ext}((\mathsf{pk}_i, c_i^*), \mathsf{ek}))$;
- When $b_i = 1$, NIZK π_i' verifies and the tuple $c_i' = c_i^*$, set $m_i' = f_i^*(m_i)$;
- otherwise set $m_i' = \perp$.

From the discussion above, we see that all the hybrids above are computationally indistinguishable because of the true simulation-extractability property, as the attacker only sees simulated proof of true statements, and the extractor Ext should succeed extracting on any proof not output for the tSE-oracle. We denote **Hybrid** 4.n by **Hybrid** 4.

Hybrids 5.1-5.n. Finally, we see that **Hybrid** 4 does not use the decryption algorithm at all, instead running various extractors Ext_i and Ext to output the correct messages. Hence, we can now use the semantic security of FHE, and start replacing real FHE-encryptions of $m_1, \ldots, m_n$ by encryptions of 0. Doing it for all n ciphertexts, we eventually arrive at the following final **Hybrid** 5:

- Sample $(m_1, \ldots, m_n, \alpha) \leftarrow \mathcal{D}(1^\kappa)$.

- Run $(\mathsf{CRS}_1, \mathsf{tk}, \mathsf{ek}) \leftarrow \mathsf{NIZK.CRSGen}(1^\kappa)$ and $\mathsf{CRS}_2 \leftarrow \mathsf{SNARK.CRSGen}(1^\kappa)$. Set $\mathsf{CRS} = (\mathsf{CRS}_1, \mathsf{CRS}_2)$.
- Run $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.Gen}(1^\kappa)$.
- Run $c_i \leftarrow \mathsf{FHE.Enc}_{\mathsf{pk}}(0; \mathsf{r_i})$ and $\pi_i \leftarrow \mathsf{NIZK.Sim}(\mathsf{CRS}_1, (\mathsf{pk}, c_i), \mathsf{tk})$. Set $C_i = (0, c_i, \pi_i)$.
- Let $(C'_1, \ldots, C'_n) \leftarrow \mathcal{A}(\mathsf{pk}_1, \ldots, \mathsf{pk}_n, c_1, \ldots, c_n)$.
- Parse $C'_i = (b_i, c'_i, \pi'_i)$.
- If $b_i = 0$ and π'_i does not tSE-verify, set $m_i = \perp$. Else
- If $b_i = 0$ and $c'_i \neq c_i$, set $m'_i = \mathsf{Ext}((\mathsf{pk}_i, \mathbf{c}'_i), \pi'_i, \mathsf{ek})$. Else
- If $b_i = 0$ and $c'_i = c_i$, set $m'_i = m_i$. Else
- If $b_i = 1$, and π'_i does not SNARK-verify, set $m_i = \perp$. Else
- Let $(f^*_i, c^*_i, \pi^*_i) \leftarrow \mathsf{Ext}_i(\mathsf{CRS}_2, (c'_i, \pi'_i); \textit{randomness})$.
- If $c^*_i \neq c_i$, set $m'_i = f^*_i(\mathsf{Ext}((\mathsf{pk}_i, \mathbf{c}^*_i), \mathsf{ek}))$. Else
- (If $c^*_i = c_i$,) set $m'_i = f^*_i(m_i)$.

From this description, we see that **Hybrid** 5 is equivalent to the simulated setting in the Definition of spooky-free encryption, where the simulator $\mathcal{S}$ can use shared randomness to sample all the common values (encryption/decryption key, various trapdoors, etc.), will run $\mathcal{A}$ on a bunch of encryptions of 0, but only needs to know the i-the message m_i to generate m'_i. This completes the proof.