Garbling XOR Gates "For Free" in the Standard Model

Benny Applebaum*

Abstract

Yao's garbled circuit (GC) technique is a powerful cryptographic tool which allows to "encrypt" a circuit C by another circuit $\hat{C}$ in a way that hides all information except for the final output. Yao's original construction incurs a constant overhead in both computation and communication per gate of the circuit C (proportional to the complexity of symmetric encryption). Kolesnikov and Schneider (ICALP 2008) introduced an optimized variant that garbles XOR gates "for free" in a way that involves no cryptographic operations and no communication. This variant has become very popular and has lead to notable performance improvements.

The security of the free-XOR optimization was originally proved in the random oracle model. Despite some partial progress (Choi et al., TCC 2012), the question of replacing the random oracle with a standard cryptographic assumption has remained open.

We resolve this question by showing that the free-XOR approach can be realized in the standard model under the *learning parity with noise* (LPN) assumption. Our result is obtained in two steps:

1. We show that the random oracle can be replaced with a symmetric encryption which remains secure under a combined form of related-key (RK) and key-dependent message (KDM) attacks;
2. We show that such a symmetric encryption can be constructed based on the LPN assumption.

As an additional contribution, we prove that the combination of RK and KDM security is non-trivial in the following sense: There exists an encryption scheme which achieves RK security and KDM security separately, but breaks completely at the presence of combined RK-KDM attacks.

1 Introduction

Yao's *garbled circuit* (GC) construction [43] is an efficient transformation which maps any boolean circuit $C : \{0,1\}^n \to \{0,1\}^m$ together with secret randomness into a "garbled circuit" $\hat{C}$ along with n pairs of short k-bit keys (W_i^0, W_i^1) such that, for any (unknown) input x, the garbled circuit $\hat{C}$ together with the n keys $W_x = (W_1^{x_1}, \ldots, W_n^{x_n})$ reveal $C(x)$ but give no additional information about x. Yao's celebrated result shows that such a transformation can be based on the existence of any pseudorandom generator [13, 42], or equivalently a one-way function [21].

Originally motivated by the problem of secure multiparty computation [42, 20], the GC construction has found a diverse range of other applications to problems such as computing on encrypted data, parallel cryptography, verifiable computation, software protection, functional encryption, and key-dependent message security (see [5] for references). Despite its theoretical importance, GC was

*School of Electrical Engineering, Tel-Aviv University, bennyap@post.tau.ac.il. Supported by Alon Fellowship, ISF grant 1155/11, Israel Ministry of Science and Technology (grant 3-9094), and GIF grant 1152/2011.

typically considered to be impractical due to a large computational and communication overhead which is proportional to the circuit size. This belief was recently challenged by a fruitful line of works that optimizes the concrete efficiency of GC-based protocols up to a level that suits large-scale practical applications [37, 34, 31, 30, 39, 38, 22, 23, 41, 24, 29].

Among other improvements, all current implementations of GCs (e.g., [39, 22, 33, 41, 23]) employ the so-called *free-XOR* optimization of Kolesnikov and Schneider [28]. While in Yao's original construction every gate of the circuit C has a computational cost of few cryptographic operations (e.g., three or four applications of a symmetric primitive) and a communication cost of few ciphertexts, Kolesnikov and Schneider showed how to completely eliminate the communication and computational overhead of XOR-gates. Although this leads "only" to an efficiency improvement by a constant factor, the effect on the practical performance turns to be significant, especially for large or medium size circuits as demonstrated in [28, 27, 39].

As in many cases, this gain in efficiency requires stronger cryptographic assumptions. Unlike Yao's GC, which can be based on the existence of standard symmetric-key cryptography, the free-XOR optimization relies on a hash function H which is modeled as a random oracle [9]. Due to the known limitations of the random oracle model [16], it is natural to ask:

> Is it possible to realize the free-XOR optimization in the standard model?

This question was raised in the original work of Kolesnikov and Schneider [28] and was further studied in [3, 17]. In [28] it was conjectured that the full power of the random oracle is not really needed, and that the function H can be instantiated with a *correlation-robust hash function* [25], a strong (yet seemingly realizable) version of a hash function which remains pseudorandom even when it is applied to linearly related inputs. Choi et al. [17] showed that the picture is actually more complex: correlation robustness alone does not suffice for security (as demonstrated by an explicit counter-example in the random-oracle model). Instead, one has to employ a stronger form of hash function which, in addition to being correlation-robust, also satisfies some form of circular security [15, 10]. While the existence of circular correlation-robust hash functions (a new primitive introduced by Choi et al. [17]) seems to be a reasonable assumption (significantly weaker than the existence of a random oracle), it is still unknown how to realize it based on a standard cryptographic assumption. This leaves open the problem of implementing the free-XOR optimization in the standard model.

1.1 Our Contribution

We resolve the above feasibility question by showing that the free-XOR optimization can be realized in the standard model under the learning parity with noise (LPN) assumption [19, 11]. This assumption, which can also be formulated as the intractability of decoding a random linear code, is widely studied by the coding and learning communities and was extensively employed in cryptographic constructions during the last two decades.

Specifically, we make the following contributions:

1. We introduce a new combined form of Related Key (RK) and Key Dependent Message (KDM) attacks. Roughly speaking, in such an attack the adversary is allowed to see ciphertexts of the form $\mathsf{Enc}_{\phi(K)}(\psi(K))$ where K is the secret key and the functions ϕ and ψ are chosen by the adversary from some predefined function families. This notion of security, referred to as *RK-KDM security*, generalizes the previous definitions of semantic security under related key

attacks [3] and key-dependent message attacks [15, 10]. In fact, as shown in Section 5, this is a *strict* generalization as there exists an encryption scheme which satisfies both RK-security and KDM-security separately, but fails to achieve the combined form of RK-KDM security.

2. We prove that the free-XOR construction is secure when instantiated with a semantically-secure symmetric encryption scheme whose security is preserved under linear RK-KDM attacks. (Essentially, $\phi(K) = K \oplus \Delta_1$ and $\psi(K) = K \oplus \Delta_2$ for any fixed shift vectors Δ_1 and Δ_2.)

3. We show that the LPN-based symmetric encryption of [18] and its generalization [2] satisfies RK-KDM security with respect to linear functions. In fact, our proof provides a general template for proving RK-KDM security based on pseudorandomness and joint key/message homomorphism. This is similar to previous results along these lines [14, 2, 6, 3].

Altogether our proofs turn to be quite simple (which we consider as a virtue), short and modular. This is due to the following choices:

Encryption vs. Hashing. The key point in which we deviate from [28, 17] is the use of (randomized) *symmetric encryption*, as opposed to deterministic *hash function* (or some other pseudorandom primitive). Indeed, the GC construction essentially employs the hash function only as a "computational one-time pad", namely, as a mean to achieve secrecy. Therefore, in terms of functionality it seems best (i.e., more general) to abstract the underlying primitive as an encryption scheme. While this is true in general for the standard GC (cf. [31, 4] and the recent discussion in [7]), this distinction becomes even more important in the context of the free-XOR variant. In this case, the underlying primitive should satisfy stronger notions of security (RKA and KDM), and this turns to be much easier for randomized encryption than for pseudorandom objects such as hash functions. (See also [3].) As a secondary gain, the new security definition that arises for symmetric encryption (RKA-KDM semantic security) is natural and compatible with existing well-studied notions. In contrast, the analog definition of RKA-KDM security for hash functions (*circular correlation-robustness*) appears less natural as there is no obvious interpretation for the concepts of *message* and *key*.

GC as Randomized Encoding. It is important to distinguish between the garbled circuit transformation (i.e., the mapping from C to $\hat{C}$) and the secure function evaluation protocol which is based on it. The distinction between the two, which is sometimes blurred, can be formulated via the notion of *randomized encoding of functions* [26] as done in [4]. Our proofs follow this abstraction, and show that the free-XOR technique yields computationally private randomized encoding. At this point one can invoke, for example, the general theorem of [4] to derive a secure MPC protocol. Similarly, all other applications (cf. [1]) of randomized encoding can be obtained directly by invoking the reduction from RE to the desired task. This is the first modular treatment of the free XOR variant.

1.2 Discussion

The main goal of this work is to provide a solid theoretical justification for the free-XOR heuristic. This is part of an ongoing effort of the theory community to explain the security of "real world" protocols. Several such examples arise when trying to import random-oracle based protocols to

the standard model. In this context, [16] suggested a two-step methodology: (1) "identify useful special-purpose properties of the random oracle" and (2) show that these properties "can be also provided by a fully specified function (or function ensemble)". In the context of the free-XOR technique, the first step was essentially taken by [17] who identified the extra need of "circular security", while the current paper completes the second step which involves, in addition, some fine-tuning of step 1.

It should be emphasized that we do not suggest to replace the hash function with an LPN-based scheme in practical implementations (though we do not rule out such a possibility either). Still, we believe that the results of this work are useful even if one decides, due to efficiency considerations, to use a heuristic implementation. Specifically, viewing the primitive as an RKA-KDM secure encryption scheme allows to rely on other heuristic solutions such as block ciphers, for which RKA and KDM security are well studied.

Other related works. The notions of key-dependent message security (aka circular security) and related-key attacks were introduced by [15, 10] and [8]. Both notions were extensively studied (separately) during the last decade. Most relevant to this paper is our joint work with Harnik and Ishai [3]. This work introduces the notion of semantic security under related-key attacks, describes several constructions, and shows that protocols employing correlation-robust hash functions and their relatives (e.g., [36, 25]), can be securely instantiated with RKA-secure encryption schemes. In addition, [3] suggested to apply a similar modification to the free-XOR variant, which was believed to be secure when instantiated with correlation-robust hash functions [28]. As mentioned, the latter claim was found to be inaccurate, and therefore the results of [3] cannot be used in the context of the free-XOR technique. (The other applications mentioned in [3] remain valid.)

Organization. Following some preliminaries (Section 2), in Section 3 we define semantic security under RK-KDM attacks and describe an LPN-based implementation. Section 4 is devoted to the garbled circuit construction, including definitions (in terms of randomized encoding), a description of Yao's original construction and the free-XOR variant, and a proof of security that reduces the privacy of the free-XOR GC to the RK-KDM security of the underlying encryption. In Section 5, we describe an encryption scheme which is KDM secure and RKA secure but not RK-KDM secure, separating the latter notion from the formers. Finally, we end with a short conclusion in Section 6.

2 Preliminaries

We let $\circ$ denote string concatenation. Strings are often treated as vectors or matrices over the binary field $\mathbb{F}_2$, accordingly string *addition* is interpreted simply as bit-wise exclusive-or. When adding together two matrices $A_{n\times k}$ and $B_{N\times k}$ where $n < N$ we assume that the last $N-n$ missing rows of A are padded with zeroes. The same convention holds with respect to vectors (i.e., when $k = 1$).

2.1 Randomized functions

We extensively use the abstraction of randomized functions which can be seen as a special case of Maurer's Random Systems [35]. A *randomized function* is a two argument function $f : X \times R \to Y$ whose first input x is referred to as the *deterministic input* and the second input is referred to as

the *random input*. For every deterministic input x, we think of $f(x)$ as the random variable induced by sampling $r \xleftarrow{R} R$ and computing $f(x;r) \in Y$. When a (randomized) algorithm A gets an oracle access to a randomized function f, we assume that A has control only on the deterministic input; namely, if A queries f with x, it gets as a result a fresh sample from $f(x)$. Note that A^f itself defines a randomized function. We say that $\{f_s\}_{s\in\{0,1\}^*}$ is a *collection of randomized functions* if f_s is a randomized function for every key s. By default, all the collections are efficiently computable in the sense that $f_s(x)$ can be sampled in time $\mathrm{poly}(|s|+|x|)$.

Indistinguishability. A pair of randomized functions f, g is *equivalent* $f \equiv g$ if for every input x the random variables $f(x)$ and $g(x)$ are identically distributed. A pair $f = \{f_s\}$ and $g = \{g_s\}$ of collections of randomized functions is *computationally indistinguishable*, denoted by $f \overset{c}{\equiv} g$, if for every efficient adversary $\mathcal{A}$ it holds that

$$\left| \Pr_{s\xleftarrow{R}\{0,1\}^k}[\mathcal{A}^{f_s}(1^k)=1] - \Pr_{s\xleftarrow{R}\{0,1\}^k}[\mathcal{A}^{g_s}(1^k)=1] \right| < \mathrm{neg}(k).$$

We extend the above definition to the case of collections $f = \{f_{1^\kappa}\}$ and $g = \{g_{1^\kappa}\}$ which contain a single randomized function for every input length κ. In this case, we augment f (resp., g) by letting $f_s = f_{1^{|s|}}$ (resp., $g_s = g_{1^{|s|}}$) and use the previous definition.[1]

Let $\{f_s\}, \{g_s\}$ and $\{h_s\}$ be collections of randomized functions. We will need the following standard facts (cf. [35]).

Fact 2.1. *If for every $k \in \mathbb{N}$, $\Pr_{s\xleftarrow{R}\{0,1\}^k}[f_s \equiv g_s] > 1-\varepsilon(k)$ for some negligible function ε, then $\{f_s\} \overset{c}{\equiv} \{g_s\}$.*

Fact 2.2. *If $\{f_s\} \overset{c}{\equiv} \{g_s\}$ and A is an efficient function then $\left\{A^{f_s}\right\}_s \overset{c}{\equiv} \{A^{g_s}\}_s$.*

Fact 2.3. *If $\{f_s\} \overset{c}{\equiv} \{g_s\}$ and $\{g_s\} \overset{c}{\equiv} \{h_s\}$ then $\{f_s\} \overset{c}{\equiv} \{h_s\}$.*

3 RK-KDM Security

A pair of efficient probabilistic algorithms $(\mathsf{Enc}, \mathsf{Dec})$ is a *symmetric encryption scheme* over the message-space $\{0,1\}^*$ and key-space $\{0,1\}^k$ (where k serves as the security parameter) if for every message $M \in \{0,1\}^*$

$$\Pr_{s\xleftarrow{R}\{0,1\}^k}[\mathsf{Dec}_s(\mathsf{Enc}_s(M)) = M] = 1.$$

We also assume (WLOG) length-regularity, i.e., that messages of equal length M, M' are always encrypted by ciphertexts of equal length $|\mathsf{Enc}_s(M)| = |\mathsf{Enc}_s(M')|$.

Our security definitions are parameterized by a family of key-derivation and key-dependent-message functions (which are also indexed by the security parameter k)

$$\Phi_{\mathsf{RKA}} = \left\{\phi : \{0,1\}^k \to \{0,1\}^k\right\}, \qquad \Psi_{\mathsf{KDM}} = \left\{\psi : \{0,1\}^k \to \{0,1\}^*\right\}.$$

[1] More generally, one can define computational indistinguishability with respect to a pair of key sampling algorithm $\mathsf{KeyGen}_f(1^\kappa)$ and $\mathsf{KeyGen}_g(1^\kappa)$ which induce, for every security parameter κ, a probability distribution over the ensembles f and g. However, for this paper the simpler definition suffices.

These families determine the legal relations between the related-keys, and the key-related messages. RK-KDM Security is defined via the following pair of real/fake oracles Real_s and Fake_s which are indexed by a key $s \in \{0,1\}^k$. For a query $(\phi \in \Phi_{\mathsf{RKA}}, \psi \in \Psi_{\mathsf{KDM}})$, the oracle Real_s returns a sample from the distribution $\mathsf{Enc}_{\phi(s)}(\psi(s))$, whereas, the oracle Fake_s returns a sample from the distribution $\mathsf{Enc}_{\phi(s)}(0^{|\psi(s)|})$.

Definition 3.1 (RK-KDM-secure encryption). *A symmetric encryption scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *is* semantically-secure under Related-Key and Key-Dependent Message Attacks *(in short,* RK-KDM-*secure) with respect to* $\Phi_{\mathsf{RKA}}, \Psi_{\mathsf{KDM}}$ *if* $\mathsf{Real}_s \stackrel{c}{\equiv} \mathsf{Fake}_s$ *where* $s \stackrel{R}{\leftarrow} \{0,1\}^k$.

Remarks:

- **Relation to previous definitions.** We note that the above definition generalizes semantic security under related-key attacks [3] and semantic security under key-dependent message attacks [10]. Indeed, the former notion is obtained by restricting Ψ_{KDM} to contain only constant functions, and the latter is obtained by letting Φ_{RKA} contain only the identity function. If both restrictions are applied simultaneously, the definition becomes identical to standard semantic security under Chosen-Plaintext Attacks. On the other hand, as we show in Section 5, a scheme may satisfy both RKA security and KDM security (separately) without achieving the combined form of RKA-KDM security.

- **Non-Adaptivity.** Definition 3.1 allows the adversary to choose its queries in a fully adaptive way. One may define a seemingly weaker non-adaptive variant in which the adversary has to specify all its queries at the beginning of the game. We note that this weaker variant suffices for the free-XOR application.

- **LIN RK-KDM security.** We will be interested in linear functions over $\mathbb{F}_2$. Namely, both Φ_{RKA} and Ψ_{KDM} contain functions of the form $s \mapsto s + \Delta$ for every $\Delta \in \mathbb{F}_2^k$. To be compatible with standard semantic security, we require that Ψ_{KDM} also contains all fixed functions. Using a compact notation, we can describe each function in Ψ_{KDM} by a message M and a bit σ and let $g_{M,\sigma} : s \mapsto (M + (\sigma \cdot s))$. If the length of M is larger than k, we assume that $(\sigma \cdot s)$ is padded with zeroes at the end. Hence, the adversary may ask for an encryption of the shifted key concatenated with some fixed message. We refer to this notion as *LIN RK-KDM* security.[2]

3.1 LPN-based Construction

The *learning parity with noise* problem is parameterized by positive integers k, t, and noise parameter $\varepsilon \in (0, \frac{1}{2})$. The input to the problem is a random matrix $A \stackrel{R}{\leftarrow} \mathbb{F}_2^{t \times k}$ and a vector $y = As + e \in \mathbb{F}_2^t$ where $s \stackrel{R}{\leftarrow} \mathbb{F}_2^k$ and $e \stackrel{R}{\leftarrow} \mathsf{Ber}_\varepsilon^t$ is an "error" vector of t independent Bernoulli random variable which take the value 1 with probability ε. The goal is to recover the secret vector s. This can be considered to be a "decoding game" where A generates a random linear code and the goal is to recover

[2] A seemingly weaker definition of LIN RK-KDM security restricts the KDM family to functions $g_{M,\sigma} : s \mapsto (M + (\sigma \cdot s))$ where M and s are of the same length k. We note that a scheme that satisfies this notion can be trivially converted into a scheme that satisfies our definition (which supports M longer than s). This can be done by partitioning the long message M into t blocks $M_1, \ldots, M_t$ of length k each, and concatenating the encryptions of these blocks. A query of the form $(f \in \Phi_{\mathsf{RKA}}, g_{M,\sigma})$ can then be emulated by a linear query $(f \in \Phi_{\mathsf{RKA}}, g_{M_1,1})$ and $t-1$ fixed-message query $(f \in \Phi_{\mathsf{RKA}}, g_{M_i,0})$.

a random information word s given a noisy codeword y. For integer-valued function $t = t(k)$ and a parameter ε, we say that the problem $\mathsf{LPN}_{t,\varepsilon}$ is *hard*, if there is no efficient adversary that can solve it with more than negligible success probability. We say that $\mathsf{LPN}_{\varepsilon}$ is *hard* if $\mathsf{LPN}_{t,\varepsilon}$ is hard for every polynomial $t(\cdot)$. It is widely believed that $\mathsf{LPN}_{\varepsilon}$ is hard for any constant $\varepsilon \in (0, \frac{1}{2})$, and the best known algorithm runs in time $2^{\Theta(n/\log n)}$ [12]. In the following we describe the LPN-based symmetric encryption scheme of [2] which is a variant of the scheme of [18].

Parameters. Let $\ell = \ell(k)$ be a message-length parameter which is set to be an arbitrary polynomial in the security parameter k. (Shorter messages are padded with zeroes.) Let $\varepsilon < \frac{1}{2}$ and $0 < \delta < \frac{1}{2}$ be constants. We will use a family of linear binary error-correcting codes with information words of length $\ell(k)$ and block length $t = t(n)$, that has an efficient decoding algorithm D that can correct up to $(\varepsilon + \delta) \cdot t$ errors. We let $G = G_\ell$ be the $t \times \ell$ binary generator matrix of this family and we assume that it can be efficiently constructed (given 1^k).

Construction 3.2 (LPN-construction)**.** *Let $N = N(k)$ be an arbitrary polynomial (which controls the tradeoff between the key-length and the time complexity of the scheme). The private key of the scheme is a matrix S which is chosen uniformly at random from $\mathbb{F}_2^{k \times N}$.*

- Encryption*: To encrypt a message $M \in \mathbb{F}_2^{\ell \times N}$, choose a random $A \xleftarrow{R} \mathbb{F}_2^{t \times k}$ and a random noise matrix $E \xleftarrow{R} \mathsf{Ber}_{\varepsilon}^{t \times N}$. Output the ciphertext*

$$(A, A \cdot S + E + G \cdot M).$$

- Decryption*: Given a ciphertext (A, Z) apply the decoding algorithm D to each of the columns of the matrix $Z - AS$ and output the result.*

Observe that the decryption algorithm errs only when there exists a column in E whose Hamming weight is larger than $(\varepsilon+\delta)m$, which, by Chernoff Bound, happens with negligible probability. (This error can be eliminated by rejecting noise vectors whose relative Hamming weight exceeds $(\varepsilon + \delta)$.) The scheme is also highly efficient. Encryption requires only cheap matrix operations and decryption requires in addition to decode the code G. It is shown in [2] that for proper choice of parameters both encryption and decryption can be done in quasilinear time in the message length (for sufficiently long message).

Construction 3.2 was proven to be semantically secure based on the intractability of the $\mathsf{LPN}_{\varepsilon}$ problem [2]. Security against KDM and RKA attacks with respect to linear functions was further proven in [2] and [3]. We now generalize these results and show that the scheme is LIN RK-KDM secure.

Theorem 3.3. *Assuming that $\mathsf{LPN}_{\varepsilon}$ is hard, the above construction is LIN RK-KDM secure.*

3.2 Proof of Theorem 3.3

Through this section we keep the convention that $S \in \mathbb{F}_2^{k \times N}$ is a key, $\Delta \in \mathbb{F}_2^{k \times N}$ is a key-shift vector, $M \in \mathbb{F}_2^{\ell \times N}$ is a message, $b \in \{0, 1\}$ is a bit, and the pair $(A, Z) \in \mathbb{F}_2^{t \times k} \times \mathbb{F}_2^{t \times N}$ is a potential ciphertext. In addition, we let Enc denote the LPN encryption defined in Construction 3.2.

Recall that our goal is to prove that for a random key $S \xleftarrow{R} \mathbb{F}_2^{k\times N}$ the randomized functions

$$\mathsf{Real}_S : (\Delta, M, b) \mapsto \mathsf{Enc}_{S+\Delta}(M + bS)$$
$$\mathsf{Fake}_S : (\Delta, M, b) \mapsto \mathsf{Enc}_{S+\Delta}(0^{\ell\times N}),$$

are indistinguishable. This will be proven via a sequence of hybrids.

Let $\mathcal{R}_S$ be a randomized function which ignores the key S and the given input, and outputs a fresh uniformly chosen matrices $A \xleftarrow{R} \mathbb{F}_2^{t\times k}$ and $Z \xleftarrow{R} \mathbb{F}_2^{t\times N}$. (If $\mathcal{R}_S$ is applied to the same input more than once it responds with independent answers.)

The following lemma shows that the LPN encryption scheme is not only semantically secure but also pseudorandom in the following sense:

Lemma 3.4. *Assuming that* LPN_ε *is hard,* $\{\mathsf{Enc}_S\} \stackrel{c}{\equiv} \{\mathcal{R}_S\}$*, where* $S \xleftarrow{R} \mathbb{F}_2^{k\times N}$.

The proof is implicit in [2], we include it here for completeness.

Proof. Fix some $\varepsilon \in (0, \frac{1}{2})$. For polynomials $N, t = \mathrm{poly}(\kappa)$, and $S \xleftarrow{R} \mathbb{F}_2^{k\times N}$ we define the randomized functions $\mathsf{LPN}_S^{t\times N}$ and $\mathcal{R}_S^{t\times N}$ which have no input (or equivalently ignore their input) as follows. In each call, $\mathsf{LPN}_S^{t\times N}$ samples a random matrix $A \xleftarrow{R} \mathbb{F}_2^{t\times k}$, a random noise matrix $E \xleftarrow{R} \mathsf{Ber}_\varepsilon^{t\times N}$, and outputs the pair $(A, A\cdot S+E)$. The function $\mathcal{R}_S^{t\times N}$ is defined similarly to $\mathcal{R}_S$, namely, in each call it simply outputs a fresh random pair $A \xleftarrow{R} \mathbb{F}_2^{t\times k}$ and $Z \xleftarrow{R} \mathbb{F}_2^{t\times N}$. The well-known search-to-decision reduction of [11] shows that, assuming the hardness of LPN_ε,

$$\left\{\mathsf{LPN}_S^{t\times N}\right\} \stackrel{c}{\equiv} \left\{\mathcal{R}_S^{t\times N}\right\}, \tag{1}$$

for $N = 1$ and any polynomial t. A standard hybrid argument allows to extend Eq. 1 to the case of an arbitrary polynomial N (and arbitrary polynomial t), as done in [2]. It remains to show that Eq. 1 implies the lemma.

Fix t, N to be the parameters from Construction 3.2, and let $G \in \mathbb{F}_2^{t\times\ell}$ be the generator matrix in use. Define an oracle aided function $\mathcal{A}^{(\cdot)}$ which given $M \in \mathbb{F}_2^{\ell\times N}$ calls its oracle $\mathcal{O}$ to obtains a pair (A, R) and outputs $(A, R + GM)$. For every S we have that

$$\mathsf{Enc}_S \equiv \mathcal{A}^{\mathsf{LPN}_S^{t\times N}} \qquad \text{and} \qquad \mathcal{A}^{\mathcal{R}_S^{t\times N}} \equiv \mathcal{R}_S,$$

and, therefore, by Eq. 1 and Facts 2.1,2.2 and 2.3, we have that $\{\mathsf{Enc}_S\} \stackrel{c}{\equiv} \{\mathcal{R}_S\}$ as required. □

We will need the following key observation:

Lemma 3.5. *There exists an efficient oracle machine* $F^{(\cdot)} : (\Delta, M, b) \mapsto (A, Z)$ *such that*

$$\mathsf{Real}_S \equiv F^{\mathsf{Enc}_S} \qquad \textit{and} \qquad F^{\mathcal{R}_S} \equiv \mathcal{R}_S,$$

for every $S \in \mathbb{F}_2^{k\times N}$.

Proof. We define F as follows: Given a query (Δ, M, b) the machine F calls the oracle with input M, gets back the answer (A', Z'), and outputs the pair $A = A' + GH$ and $Z = Z' + A\Delta$ where G is the generating matrix used in Construction 3.2 and $H \in \mathbb{F}_2^{\ell\times k}$ is the matrix $\left(\begin{smallmatrix} b\cdot I_{k\times k} \\ 0^{\ell-k\times k}\end{smallmatrix}\right)$.

Fix a key S and a query (Δ, M, b), we will show that $F^{\mathsf{Enc}_S}(\Delta, M, b)$ is distributed identically to $\mathsf{Real}_S(\Delta, M, b)$. Let (A', Z') be a fresh sample from $\mathsf{Enc}_S(M)$. Clearly, $A = A' + GH$ is uniform in $\mathbb{F}_2^{t \times k}$ since A' is uniform. In addition, since $Z' = A' \cdot S + E + G \cdot M$ where $E \xleftarrow{R} \mathsf{Ber}_\varepsilon^{t \times N}$, and since $A' = A + GH$ we can write Z as

$$\begin{aligned}(A + GH) \cdot S + E + G \cdot M + A\Delta &= A \cdot (S + \Delta) + E + G \cdot (M + HS) \\ &= A \cdot (S + \Delta) + E + G \cdot (M + bS),\end{aligned}$$

where the first equality is due to linearity, and the second equality follows from the definition of H. It follows that (A, Z) is a fresh sample from $\mathsf{Enc}_{S+\Delta}(M + bS)$.

To prove that $F^{\mathcal{R}_S} \equiv \mathcal{R}_S$, it suffices to show that for any fixed query (Δ, M, b) the transformation from (A', Z') to (A, Z) is an affine invertible mapping. This follows immediately from the definition of F. □

We conclude that for $S \xleftarrow{R} \mathbb{F}_2^{k \times N}$,

$$\mathsf{Real}_S \equiv F^{\mathsf{Enc}_S} \stackrel{c}{\equiv} F^{\mathcal{R}_S} \equiv \mathcal{R}_S. \tag{2}$$

Indeed, the first and third transitions are due to Lemma 3.5, and the second transition is due to Lemma 3.4 and Fact 2.2.

To complete the argument we need two additional definitions. First we define an oracle machine which given an oracle $\mathcal{O}$ and an input (Δ, M, b) outputs a sample from $F^{\mathcal{O}}(\Delta, 0^{\ell \times N}, 0)$; namely, it replaces M, b with zeroes and proceeds as $F^{\mathcal{O}}$. By abuse of notation, we refer to this oracle as $F(\cdot, 0^{\ell \times N}, 0)$. Similarly, we let $\mathsf{Real}_S(\cdot, 0^{\ell \times N}, 0)$ denote the randomized function which maps (Δ, M, b) to $\mathsf{Real}_S(\Delta, 0^{\ell \times N}, 0)$. Note that the latter is just an equivalent formulation of Fake_S. Moreover, we can write:

$$\begin{aligned}\mathcal{R}_S &\equiv F(\cdot, 0^{\ell \times N}, 0)^{\mathcal{R}_S} \stackrel{c}{\equiv} F(\cdot, 0^{\ell \times N}, 0)^{\mathsf{Enc}_S(0^{\ell \times N})} \\ &\equiv \mathsf{Real}_S(\cdot, 0^{\ell \times N}, 0) \equiv \mathsf{Fake}_S,\end{aligned} \tag{3}$$

where the first and third transitions are due to Lemma 3.5, and the second transition is due to Lemma 3.4 and Fact 2.2. By combining Eq. 2 and Eq. 3 with Fact 2.3 we get that $\mathsf{Real}_S \stackrel{c}{\equiv} \mathsf{Fake}_S$, and Theorem 3.3 follows. □

Remark 3.6 (Abstraction)**.** *The proof of Theorem 3.3 provides a general template for proving RKA-KDM security. Specifically, the properties needed are pseudorandomness (in the sense of Lemma 3.4) and key/message homomorphism (in the sense of Lemma 3.5). Indeed, observe that, apart from the proofs of Lemmas 3.4 and 3.5, the overall proof can be written in a fully generic form with no specific references to the LPN construction.*

4 Yao's Garbled Circuit

4.1 Definition

Let $f = \{f_n\}_{n \in \mathbb{N}}$ be a polynomial-time computable function. In an abstract level, Yao's garbled circuit technique [43] constructs a randomized function $\hat{f} = \{\hat{f}_n\}_{n \in \mathbb{N}}$ which "encodes" f in the sense that for every x the distribution $\hat{f}(x)$ reveals the value of $f(x)$ but no other additional information.

We formalize this via the notion of *computationally private randomized encoding* from [4], while adopting the original definition from a non-uniform adversarial setting to the uniform setting (i.e., adversaries are modeled by probabilistic polynomial-time Turing machines).

Definition 4.1 (**Computational randomized encoding**). *Let $f = \{f_n : \{0,1\}^n \to \{0,1\}^{\ell(n)}\}_{n\in\mathbb{N}}$ be an efficiently computable function and let $\hat{f} = \{\hat{f}_n : \{0,1\}^n \times \{0,1\}^{m(n)} \to \{0,1\}^{s(n)}\}_{n\in\mathbb{N}}$ be an efficiently computable randomized function. We say that $\hat{f}$ is a* computational randomized encoding *of f (or encoding for short), if there exist an efficient recovery algorithm* Rec *and an efficient probabilistic simulator algorithm* Sim *that satisfy the following:*

- **Perfect correctness.** *For any n and any input $x \in \{0,1\}^n$,*

$$\Pr[\mathsf{Rec}(1^n, \hat{f}_n(x)) \neq f_n(x)] = 0.$$

- **Computational privacy.** *The randomized function $\hat{f}_n(\cdot)$ is computationally indistinguishable from the randomized function $\mathsf{Sim}(1^n, f_n(\cdot))$.*

Remark 4.2. *The above definition uses n both as an input length parameter and as a cryptographic "security parameter" quantifying computational privacy. When describing the construction, it will be convenient to use a separate parameter k for the latter, where computational privacy will be guaranteed as long as $k \geq n^\epsilon$ for some constant $\epsilon > 0$. Furthermore, while it is convenient to define randomized encoding for a single function f, Yao's construction (as well as the free-XOR variant) actually provides an efficient* compiler *that maps the function f (represented as a Boolean circuit) into (circuit representations of) the encoding $\hat{f}$, the recovery algorithm* Rec *and the simulator* Sim*. (See [5] for formal definition.) In this sense the encoding is fully constructive.*

4.2 Yao's Construction and the Free XOR Variant

Let $f = \{f_n : \{0,1\}^n \to \{0,1\}^{\ell(n)}\}_{n\in\mathbb{N}}$ be a polynomial-time computable function computed by the uniform circuit family $\{C_n\}_{n\in\mathbb{N}}$. In the following we describe Yao's construction and its free-XOR variant. Our notation and terminology borrow from previous presentations of Yao's construction in [40, 37, 32, 4].

Double-keyed Encryption. Let $k = k(n)$ be a security parameter (by default, $k = n^\varepsilon$ for some constant $\varepsilon > 0$). We will employ a symmetric encryption scheme (E^2, D^2) which is keyed by a *pair* of k-bit keys K_1, K_2. Intuitively, this corresponds to a double-locked chest in the sense that decryption is possible only if one knows both keys. There are several ways to implement such an encryption scheme based on standard single-key symmetric encryption $(\mathsf{Enc}, \mathsf{Dec})$ and, for simplicity, we choose to use

$$\begin{aligned} E^2_{K_1,K_2}(M) &:= (\mathsf{Enc}_{K_1}(R), \mathsf{Enc}_{K_2}(R+M)), \\ {D^2}_{K_1,K_2}(C_1, C_2) &:= \mathsf{Dec}_{K_1}(C_1) + \mathsf{Dec}_{K_2}(C_2) \end{aligned} \tag{4}$$

where R is a random string of length $|M|$. Other choices are also applicable under the LPN assumption.

The original construction. For each wire i of the circuit C_n we assign a pair of keys: a 0-key $W_i^0 \in \{0,1\}^k$ that represents the value 0, and a 1-key $W_i^1 \in \{0,1\}^k$ that represents the value 1. For each of these pairs we randomly "color" one key black and the other key white. This is done by choosing $r_i \xleftarrow{R} \{0,1\}$ and by letting $r_i + b$ be the color of W_i^b. Fix some input x for f_n, and let $b_i = b_i(x)$ be the value of the i-th wire induced by x. We refer to the key $W_i^{b_i}$ as the *active* key of the i-th wire.

The encoding $\hat{f}_n(x;(W,r))$ consists of three parts: (1) The active keys $W_i^{b_i}$ of the input wires together with their colors c_i; (2) For each gate a propagation mechanism which allows to translate the colored active keys of the incoming wires into the colored active keys of the outgoing wires. This mechanism is implemented via an encryption table (or "gate label") in which the keys of the outgoing wire are encrypted under the keys of the incoming wires. (3) For each output wire i, we also append the semantics of the coloring, i.e., the bit r_i. Altogether, one can propagate the values of the colored active keys $(W_i^{b_i}, c_i)$ from the inputs to the outputs, and at the end reveal the values of the output wires by unmasking the colors c_i with r_i. Intuitively, privacy holds as for non-output wires the values of the colored active keys reveal nothing on their semantics b_i.

Free XOR-gates. The "free-XOR" optimization modifies the above construction by making sure that the key W_ℓ^0 and coloring r_ℓ of a wire which outgoes a XOR gate is just the sum of the keys and coloring of the incoming wires i and j, namely,

$$W_\ell^0 = W_i^0 + W_j^0, \qquad\qquad r_\ell = r_i + r_j.$$

In addition, all key pairs W_ℓ^0, W_ℓ^1 have a fixed global (secret) difference $s = W_\ell^0 + W_\ell^1$. As a result, for every pair of values $(\alpha,\beta) \in \{0,1\}^2$ for the input wires of a XOR gate, we have that

$$W_\ell^{\alpha+\beta} = W_i^\alpha + W_j^\beta.$$

Hence, one can derive the colored active key $(W_\ell^{b_\ell(x)}, r_\ell + b_\ell(x))$ of the output wire by XOR-ing the colored active keys $(W_i^{b_i(x)}, r_i + b_i(x))$, $(W_j^{b_j(x)}, r_j + b_j(x))$ of the input wires, and so gate labels are not needed. XOR gates have, therefore, no effect on the communication complexity of the encoding, and only a minor effect on the computational complexity. A formal description of the encoding is given in Figure 1.

Our main result shows that, assuming LIN RK-KDM security, the free XOR variant gives rise to a valid computational encoding:

Theorem 4.3 (Main)**.** *If the underlying symmetric encryption scheme* $(\mathsf{Enc},\mathsf{Dec})$ *is LIN RK-KDM secure, then the randomized function* $\hat{f}$*, as defined in Figure 1, is a randomized encoding of the function* f*.*

The proof of the theorem is deferred to Section 4.3 (correctness) and 4.4 (privacy).

4.3 Correctness

The following lemma shows that the encoding is correct.

Lemma 4.4 (Correctness). *There exists an efficient recovery algorithm* Rec *such that for every* $x \in \{0,1\}^n$ *it holds that*

$$\Pr[\mathsf{Rec}(1^n, \hat{f}_n(x;(r,W))) \neq f_n(x)] = 0.$$

The Encoding $\hat{f}_n$

Input: $x \in \{0,1\}^n$.

Randomness: Choose a random global shift vector $s \xleftarrow{R} \{0,1\}^k$.
For a wire ℓ that is not an output of a XOR gate let

$$r_\ell \xleftarrow{R} \{0,1\}, \qquad W_\ell^0 \xleftarrow{R} \{0,1\}^k, \qquad W_\ell^1 := W_\ell^0 + s.$$

For a wire ℓ that is an output of a XOR gate with inputs i, j let

$$r_\ell := r_i + r_j, \qquad W_\ell^0 := W_i^0 + W_j^0, \qquad W_\ell^1 := W_i^0 + s.$$

Outputs: The encoding consists of the following outputs:

1. For an input wire i, labeled by a literal χ (either some variable x_u or its negation) output $W_i^{\chi(x)} \circ (\chi(x) + r_i)$. If i is an output wire i, output the mask of this wire r_i.
2. For a non-XOR gate t that computes some binary function $g : \{0,1\}^2 \to \{0,1\}$ with input wires i, j and output wire[a] y. We associate with this gate 4 ordered outputs ("gate labels"). For every $(a_i, a_j) \in \{0,1\}^2$ we output:

$$Q_t^{a_i,a_j} := E^2_{W_i^{a_i+r_i}, W_j^{a_j+r_j}}\Big(W_y^{g(a_i+r_i,a_j+r_j)} \circ (g(a_i + r_i, a_j + r_j) + r_y)\Big), \tag{5}$$

   where $\circ$ denotes concatenation, and E^2 is a double-encryption algorithm whose randomness is omitted for simplicity.

[a] If the fan-out is larger than 1, all outgoing wires are treated as a single wire, i.e., with the same key and the same color.

Figure 1: The encoding $\hat{f}_n(x; (W, r, s))$ of the function $f_n(x)$. We assume that wires and gates of the circuit that computes f_n are numbered according to some topological order. The double-encryption algorithm $E^2_{K_1,K_2}(M)$ is defined based on a standard encryption $(\mathsf{Enc}, \mathsf{Dec})$ as in Eq. 4.

Proof. Let $\alpha = \hat{f}_n(x;(r,W))$ for some $x \in \{0,1\}^n$ and $(r,W) \in \{0,1\}^{\mu(n)}$. The decoder traverses the circuit in topological order from inputs to outputs, and for each wire y it recovers the active key $W_y^{b_y}$ together with its color $c_y = (b_y(x) + r_y)$ as follows.

If y is an input wire then the value $W_y^{b_y} \circ c_y$ is given as part of α. Otherwise, assume that the wire y outgoes a gate t whose incoming wires are i and j (for which we already computed the desired values). If t is a XOR gate then we let

$$W_y^{b_y} = W_y^{b_i+b_j} = W_i^{b_i} + W_i^{b_j}, \text{ and } c_y = (b_i + b_j) + r_y = (b_i + b_j) + (r_i + r_j) = c_i + c_j.$$

It t is not a XOR gate then we use the colors c_i, c_j of the active keys of the input wires to select the *active* label $Q_t^{c_i,c_j}$ of the gate t (and ignore the other 3 *inactive* labels of this gate). Consider this label as in Eq. (5); recall that this cipher was "double-encrypted" under the key $W_i^{c_i-r_i} = W_i^{b_i}$ and the key $W_j^{c_j-r_j} = W_j^{b_j}$. Since we have already computed the values $c_i, c_j, W_i^{b_i}$ and $W_j^{b_j}$, we can decrypt the label $Q_t^{c_i,c_j}$ (by applying the decryption algorithm D^2) and recover the value

$$W_y^{g(b_i,b_j)} \circ (g(b_i,b_j) + r_y) = W_y^{b_y} \circ (c_y),$$

where g is the function that the gate t computes, and therefore $b_y = g(b_i, b_j)$.

Finally, once we have the colors of an output wire y we can recover its value b_y by by XOR-ing c_y with the mask r_y which is given explicitly as part of α. □

4.4 Privacy

Computational privacy is slightly more subtle. The free-XOR optimization correlates the key pairs via the global shift s. This introduces two form of dependencies: (1) The four ciphertexts of every gate are encrypted under *related keys*; and (2) The keys (of the incoming wires) which are used to encrypt the gate-labels are correlated with the content of the labels (i.e., the keys of the outgoing wires). We show that if the underlying encryption $(\mathsf{Enc}, \mathsf{Dec})$ is RKA and KDM secure with respect to linear functions, then the encoding is indeed private.

Lemma 4.5 (Privacy). *There exists an efficient simulator* Sim *such that*

$$\hat{f}_n(\cdot) \stackrel{c}{\equiv} \mathsf{Sim}(1^n, f_n(\cdot)).$$

To prove the lemma we define an oracle-aided algorithm $H^{\mathcal{O}}(x)$ such that (1) when the oracle $\mathcal{O}$ is the real RK-KDM oracle (with respect to linear queries) the distribution of $H^{\mathcal{O}}(x)$ is identical to the distribution $\hat{f}_n(x)$, and (2) when the oracle $\mathcal{O}$ is the fake RK-KDM oracle, the distribution $H^{\mathcal{O}}(x)$ can be efficiently sampled based on the output $f_n(x)$, and therefore can be used as a simulator $\mathsf{Sim}(1^n, f_n(x))$. The indistinguishability of the two oracles implies that the simulator's output is computationally indistinguishable from the encoding's distribution $\hat{f}_n(x)$.

The algorithm $H^{(\cdot)}(x)$. Let $k = k(n)$, $x \in \{0,1\}^n$ be the input. We assume that H is given an oracle access to a randomized function $\mathcal{O}_s$ where $s \stackrel{R}{\leftarrow} \{0,1\}^k$ will play the role of the secret global shifts. We will assume that $\mathcal{O}_s$ has the same interface as Real_s and Fake_s, namely, given a pair of linear functions (ϕ, ψ) the oracle outputs a ciphertext of Enc. For every wire ℓ we define the following values:

1. If ℓ is not an output of a XOR gate, choose a random active key $W_\ell^{b_\ell} \xleftarrow{R} \{0,1\}^k$ and a random color bit $c_\ell \xleftarrow{R} \{0,1\}$.

2. If the wire ℓ is an output of a XOR gate, let $W_\ell^{b_\ell} := W_i^{b_i} + W_j^{b_j}$ and $c_\ell = c_i + c_j$ where i and j are the incoming wires.

3. If ℓ is an input wire, output $W_\ell^{b_\ell} \circ c_\ell$;
   If it is an output wire output $r_\ell = c_\ell - b_\ell(x)$.

4. The inactive key $W_\ell^{b_\ell+1}$ is unknown, but it can be written as a linear function of the master-key s, i.e., $\phi_\ell : s \mapsto s + W_\ell^{b_\ell}$.

For every (non-XOR) gate t with input wires i, j and output wire y we do the following:

5. Output the active label
$$Q_t^{c_i,c_j} := E^2_{W_i^{b_i}, W_j^{b_j}}(W_y^{b_y} \circ c_y) \tag{6}$$

6. Compute the inactive labels as follows. For every $(\alpha,\beta) \neq (0,0)$ choose $R_{\alpha,\beta} \xleftarrow{R} \{0,1\}^{k+1}$ and define the linear function $\psi_{\alpha,\beta}$ which maps s to the value
$$\Big((W_y^{b_y} + s \cdot g(b_i+\alpha, b_j+\beta) + b_y) \circ (g(c_i+\alpha+r_i, c_j+\beta+r_j) + r_y)\Big) + R_{\alpha,\beta},$$
   where g is the function that the gate computes, and $b_i = b_i(x)$, $r_i = b_i + c_i$, $b_j = b_j(x)$, $r_j = b_j + c_j$ and $b_y = b_y(x)$, $r_y = b_y + c_y$. Now, output
$$\begin{aligned} Q_t^{c_i+1,c_j} &:= \Big(\mathcal{O}(\phi_i, \psi_{1,0}), \mathsf{Enc}_{W_j^{b_j}}(R_{1,0})\Big) \\ Q_t^{c_i+1,c_j+1} &:= \Big(\mathcal{O}(\phi_i, \psi_{1,1}), \mathcal{O}(\phi_j, R_{1,1})\Big) \\ Q_t^{c_i,c_j+1} &:= \Big(\mathsf{Enc}_{W_i^{b_i}}(R_{0,1}), \mathcal{O}(\phi_j, \psi_{0,1})\Big), \end{aligned} \tag{7}$$
   where in the second equation, we let the string $R_{1,1}$ represent the constant function $s \mapsto R_{1,1}$.

Claim 4.6. *The randomized functions $\hat{f}_n$ and H^{Real_s} for $s \xleftarrow{R} \{0,1\}^k$ are identically distributed.*

Proof. We prove a stronger claim: for every $x \in \{0,1\}^n$ even if the encoding and the hybrid $H^{\mathsf{Real}_s}(x)$ output their internal coins (including the ones used by the oracle Real_s), the two experiments are identically distributed. First, it is not hard to verify that the values s, W_ℓ^0, r_ℓ and $W_\ell^1 = W_\ell^0 + s$ are identically distributed in both experiments. When these values are fixed, the active labels are also identically distributed. Finally, by substituting $\phi_i, \psi_{\alpha,\beta}$ in Eq. 7 it follows that the inactive labels are also distributed exactly as in $\hat{f}(x)$. □

Let us move to the case where the oracle $\mathcal{O}$ is instantiated with the oracle Fake_s for $s \xleftarrow{R} \{0,1\}^k$. By the RK-KDM security of the scheme $(\mathsf{Enc}, \mathsf{Dec})$ and Fact 2.2, we get that

Claim 4.7. *The randomized functions $\{H^{\mathsf{Real}_s}\}_s$ and $\{H^{\mathsf{Fake}_s}\}_s$ are computationally indistinguishable.*

Finally, we define the simulator which is just an equivalent description of $H^{\mathsf{Fake}_s}(x)$:

The simulator Sim. Given $z = f_n(x)$, for some $x \in \{0,1\}^n$, the simulator mimics the first three steps of H which can be computed based on the value of the output wires $f_n(x)$ (without knowing x itself). However, instead of virtually setting inactive keys in the forth step, the simulator chooses a random shift vector $s \xleftarrow{R} \{0,1\}^k$ and sets $W_\ell^{1+b_\ell} = W_\ell^{b_\ell} + s$ for every wire ℓ. Then, the simulator computes the active labels exactly as in Eq. 6. Note that all these computations can be done without knowing x (or $b_i(x)$). To compute the inactive labels the simulator mimics the distribution of $H^{\mathsf{Fake}_s}(x)$: It chooses $R_{1,0}, R_{1,1}, R_{0,1} \xleftarrow{R} \{0,1\}^{k+1}$ and computes

$$\begin{aligned} Q_t^{c_i+1,c_j} &:= \left(\mathsf{Enc}_{W_i^{b_i+1}}(0^{k+1}), \mathsf{Enc}_{W_j^{b_j}}(R_{1,0})\right) \\ Q_t^{c_i+1,c_j+1} &:= \left(\mathsf{Enc}_{W_i^{b_i+1}}(0^{k+1}), \mathsf{Enc}_{W_j^{b_j+1}}(0^{k+1})\right) \\ Q_t^{c_i,c_j+1} &:= \left(\mathsf{Enc}_{W_i^{b_i}}(R_{0,1}), \mathsf{Enc}_{W_j^{b_j+1}}(0^{k+1})\right). \end{aligned} \tag{8}$$

Indeed, all these ciphertexts can be computed directly since the inactive keys (and the global shift s) are known.

Claim 4.8. *The randomized functions $\mathsf{Sim}(f_n(\cdot))$ and $H^{\mathsf{Fake}_s}(\cdot)$ for $s \xleftarrow{R} \{0,1\}^k$ are identically distributed.*

Proof. Again, a stronger claim holds: for every $x \in \{0,1\}^n$ even if the simulator and the algorithm $H^{\mathsf{Fake}_s(\cdot)}(x)$ output their internal coins, the two experiments are identically distributed. First, it is not hard to verify that the values s, W_ℓ^0, r_ℓ and $W_\ell^1 = W_\ell^0 + s$ are identically distributed in both experiments. When these values are fixed, the active labels are also identically distributed. Finally, the inactive labels as defined by the simulator (Eq. 8) are computed exactly as they are computed by $H^{\mathsf{Fake}_s(\cdot)}(x)$ (i.e., as defined in Eq. 7 when the oracle $\mathsf{Fake}_s(\cdot)$ is being used). □

The proof of Lemma 4.5 follows from Claims 4.6–4.8 and Facts 2.1 and 2.3.

5 Separating RK-KDM from RKA & KDM

Recall that LIN RKA security corresponds to $(\Phi_{\mathsf{RKA}}, \Psi_{\mathsf{KDM}})$ RK-KDM security where Φ_{RKA} contains all linear functions (over the binary field) and Ψ_{KDM} contains the identity function. Similarly, LIN KDM security corresponds to the complementary case where Ψ_{KDM} contains all linear (and fixed) functions, and Φ_{RKA} contains the identity function.

We describe a symmetric encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ which is semantically secure under linear related-key attacks and semantically-secure under linear key-dependent message attacks but does not achieve linear RK-KDM security. In fact, one can fully recover the secret key via a combined LIN RK-KDM attack. Our counter-example is based on a pair of symmetric encryption schemes. The first scheme $(\mathsf{RE}, \mathsf{RD})$ is LIN RKA secure but can be completely broken via LIN KDM attacks, and the second scheme $(\mathsf{KE}, \mathsf{KD})$ is LIN KDM secure but can be broken via LIN RK attacks. Both schemes are based on the LPN-based encryption of Construction 3.2 instantiated with $N = 1$. Through this section we denote the LPN encryption scheme by $(\mathsf{PE}, \mathsf{PD})$ ("P" stands for parity).

5.1 Achieving RKA Security & KDM Insecurity

We define the scheme $(\mathsf{RE}, \mathsf{RD})$ identically to the LPN construction (Construction 3.2) except that if the prefix of a plaintext M is *equal* to the key S, then the corresponding ciphertext will be M itself (unencrypted). Formally[3],

$$\mathsf{RE}_S(M) := \begin{cases} M & \text{if } M_{[1:\kappa]} = S \\ \mathsf{PE}_S(M) & \text{otherwise.} \end{cases}, \qquad \mathsf{RD}_S(C) := \begin{cases} C & \text{if } C_{[1:\kappa]} = S \\ \mathsf{PD}_S(M) & \text{otherwise.} \end{cases}$$

It is not hard to prove that $(\mathsf{RE}, \mathsf{RD})$ is secure under linear related-key attacks, but is completely insecure at the presence of linear key-dependent message attacks.

Lemma 5.1. *Under the* LPN *assumption, the scheme* $(\mathsf{RE}, \mathsf{RD})$ *is secure against linear related-key attacks.*

Proof. Recall that in a LIN RK attack on an encryption algorithm E, the adversary makes queries of the form (Δ, M) and attempts to distinguish between the real oracle $E\mathsf{Real}_S$ which returns $E_{S+\Delta}(M)$ and the fake oracle $E\mathsf{Fake}_S$ which returns $E_{S+\Delta}(0^{|M|})$. The view of an adversary $\mathcal{A}$ that breaks the LIN RKA security of $(\mathsf{RE}, \mathsf{RD})$ is identical to the view of an adversary who breaks the LIN RKA security of the LPN-based scheme $(\mathsf{PE}, \mathsf{PD})$, as long as the adversary does not make a *revealing query* of the form (Δ, M) where $S + \Delta$ equals to the κ-bit prefix of M. Hence, it suffices to show that the probability of asking a revealing query is negligible. Indeed, this must be the case as a revealing query (Δ, M) can be used to recover the key by XOR-ing Δ with the κ-bit prefix of the message $M_{[1:\kappa]}$.

We proceed with a formal argument. Our goal is to prove that $\mathsf{REReal}_S \stackrel{c}{\equiv} \mathsf{REFake}_S$. First we show that REReal_S and PEReal_S are indistinguishable. Assume, towards a contradiction, that there exists some adversary $\mathcal{A}$ which distinguishes REReal_S from PEReal_S with noticeable advantage ε. We construct an adversary $\mathcal{B}^{\mathsf{PEReal}_S}$ which outputs S with noticeable probability ε/t where t is the number of queries that $\mathcal{A}$ makes. Clearly, such an adversary contradicts the LIN-RKA security of the LPN scheme. The adversary $\mathcal{B}$ simply chooses a random $i \in [t]$ and halts before making the i-th query (Δ, M) with the output $\Delta + M_{[1:\kappa]}$. To analyze the success probability of $\mathcal{B}$ we note that: (a) conditioned on not asking a revealing query the oracles REReal_S and PEReal_S are identically distributed; (b) Hence, under our assumption, $\mathcal{A}$ makes a revealing query with probability at least ε; (c) Therefore, with probability ε/t, the adversary $\mathcal{B}$ halts just before the first revealing query and, in this case, it outputs the key S.

A similar argument shows that REFake_S is indistinguishable from PEFake_S, and, since $\mathsf{PEReal}_S \stackrel{c}{\equiv} \mathsf{PEFake}_S$ we conclude, by Fact 2.3, that $\mathsf{REReal}_S \stackrel{c}{\equiv} \mathsf{REFake}_S$ and the scheme is LIN RKA secure. □

5.2 Achieving KDM Security & RKA Insecurity

The second scheme $(\mathsf{KE}, \mathsf{KD})$ is obtained by modifying the LPN construction $(\mathsf{PE}, \mathsf{PD})$ as follows. The key $S \in \{0,1\}^\kappa$ is augmented with an index $i \in \{1, \ldots, \kappa\}$. A plaintext M will be encrypted

[3]The decryption RD may err with negligible probability due to the possibility that some message M, whose prefix does not equal to the key S, will be mapped to a ciphertext $\mathsf{PE}_S(M)$ whose prefix equals to the key. This can be handled in several ways, e.g., by modifying the encryption algorithm so that such event never happens. We prefer the current version (with negligible error probability) for simplicity.

by the triple $(\mathsf{PE}_S(M), i, S_i)$, i.e., in addition to the ciphertext $\mathsf{PE}_S(M)$, we leak a single bit of the key S_i whose location i is determined by another (public) part of the key. Formally,

$$\mathsf{KE}_{S,i}(M) := (\mathsf{PE}_S(M), i, S_i), \qquad \mathsf{KD}_S(C_1, C_2, C_3) := \mathsf{PD}_S(C_1)$$

Below we show that the scheme is LIN KDM secure. In fact, it will be useful to prove KDM security with respect to a slightly richer family of "extended linear functions" which contains functions of the form $\psi_{M,T} : S \to M + TS$ for every $M \in \mathbb{F}_2^\ell$ and matrix $T \in \mathbb{F}_2^{\ell \times \kappa}$.

Lemma 5.2. *Under the* LPN *assumption, the scheme* $(\mathsf{KE}, \mathsf{KD})$ *is secure against extended linear key-dependent message attacks.*

Proof. Recall that in an extended LIN KDM attack on an encryption algorithm E, the adversary makes queries of the form (M, T) and attempts to distinguish between the real oracle $E\mathsf{Real}_S$ which returns $E_S(M + TS)$ and the fake oracle $E\mathsf{Fake}_S$ which returns $E_S(0^{|M|})$. Our goal is to show that the scheme $\mathsf{KE}_{S,i}$ is LIN KDM secure. Formally, this means that we should support functions which map the combined key $(S \circ i) \in \{0,1\}^{\kappa + \lceil \log(\kappa) \rceil}$ (viewed as a single long vector) into messages of the form $M + T \cdot (S \circ i)$. However, since i is public, any linear function in $(S \circ i)$ can be efficiently translated into a linear function in S of the form $M' + T'S$, and so it suffices to focus on such functions.

We will show how to reduce the extended LIN KDM security of $\mathsf{KE}_{S,i}$ with $S \xleftarrow{R} \{0,1\}^\kappa, i \xleftarrow{R} [\kappa]$ to the security of $\mathsf{PE}_{S'}$ with 1-bit shorter key $S' \xleftarrow{R} \{0,1\}^{\kappa-1}$. The extended LIN KDM security of the latter is proven in [2, Thm. 8]. For an index $i \in [\kappa]$ and a bit $\sigma \in \{0,1\}$, we define an oracle aided randomized function $\mathcal{A}_{i,\sigma}^{(\cdot)}$ such that

$$\mathsf{KEReal}_{S,i} \equiv \mathcal{A}_{i,\sigma}^{\mathsf{PEReal}_{S'}} \qquad \text{and} \qquad \mathcal{A}_{i,\sigma}^{\mathsf{PEFake}_{S'}} \equiv \mathsf{KEFake}_{S,i} \tag{9}$$

whenever $S = (S'_{[1:i-1]}, \sigma, S'_{[i:\kappa-1]})$. By the LIN RKA security of PE and Facts 2.1 and 2.2 we have that, for a random $S \in \{0,1\}^\kappa, i \in [\kappa]$,

$$\mathsf{KEReal}_{S,i} \equiv \mathcal{A}_{i,\sigma}^{\mathsf{PEReal}_{S'}} \stackrel{c}{\equiv} \mathcal{A}_{i,\sigma}^{\mathsf{PEFake}_{S'}} \equiv \mathsf{KEFake}_{S,i},$$

and the lemma follows by Fact 2.3.

It is left to describe the converter $\mathcal{A}$ and to prove Eq. 9. Given a KDM query (M, T) the algorithm $\mathcal{A}_{i,\sigma}$ queries his oracle with (M, T) and obtains a ciphertext $(A \in \mathbb{F}_2^{t \times \kappa - 1}, Z \in \mathbb{F}_2^t)$. Then, $\mathcal{A}_{i,\sigma}$ samples a random column $a_i \in \mathbb{F}_2^t$ and outputs the matrix $A' = (A_{[1:i-1]} | a_i | A_{[i:\kappa-1]})$, a vector $Z' = Z + \sigma \cdot a_i$ and the pair (i, σ). If the oracle returns a fresh sample $(A, Z = AS' + E + GM')$ from $\mathsf{PE}_{S'}(M')$ for some message M', then the pair $(A', Z' = A'S + E + GM')$ is a fresh sample from $\mathsf{PE}_S(M')$ where $S = (S'_{1:i-1}, \sigma, S'_{i:\kappa-1})$. This implies Eq. 9 and the lemma follows. □

On the other hand, one can fully recover the key S via an RKA by shifting the index i through all possible indices in $\{1, \ldots, \kappa\}$. Note that this attack is oblivious to the messages encrypted; In particular, all the attacker needs is the ability to obtain, for any choice of Δ, a ciphertext $\mathsf{KE}_{(S,i)+\Delta}(M)$ where the message M may be arbitrary and possibly unknown (e.g., chosen by the oracle).

5.3 Counter Example: RKA+KDM $\not\Rightarrow$ RK-KDM

Our counter-example is defined via the following double-encryption:

$$\mathsf{Enc}_{S_1,S_2}(M) := \mathsf{KE}_{S_2}(\mathsf{RE}_{S_1}(M)), \qquad \mathsf{Dec}_{S_1,S_2}(C) := \mathsf{RD}_{S_1}(\mathsf{KD}_{S_2}(C)),$$

where $S_1 \in \{0,1\}^\kappa$ and S_2 is the concatenation of a vector $S_2' \in \{0,1\}^\kappa$ and an index $i \in \{1,\ldots,\kappa\}$.

Lemma 5.3. *Under the* LPN *assumption, the scheme* $(\mathsf{Enc},\mathsf{Dec})$ *satisfies the followings:*

1. *Security under linear related-key attacks.*
2. *Security under linear key-dependent message attacks.*
3. *The secret key can be fully recovered via a LIN RK-KDM attack.*

Proof. (1) We show that any double encryption Enc, whose inner encryption RE is LIN RKA secure, is also LIN RKA secure. For an encryption E let $E\mathsf{Real}_S$ and $E\mathsf{Fake}_S$ be the real/fake RKA oracles as defined in Lemma 5.1. We define an oracle aided randomized function $\mathcal{A}^{\mathcal{O}}_{S_2}$ as follows: Given a LIN RKA query with shift vector $\Delta = (\Delta_1,\Delta_2)$ and message M, the function $\mathcal{A}^{\mathcal{O}}_{S_2}$ outputs a sample from $\mathsf{KE}_{S_2+\Delta_2}(\mathcal{O}(\Delta_1,M))$. It follows that, for random S_1 and every S_2,

$$\mathsf{EncReal}_{S_1,S_2} \equiv \mathcal{A}^{\mathsf{REReal}_{S_1}}_{S_2} \stackrel{c}{\equiv} \mathcal{A}^{\mathsf{REFake}_{S_1}}_{S_2} \equiv \mathsf{EncFake}_{S_1,S_2},$$

where the first and third transitions follow from the definition of $\mathcal{A}$ and the second transition is due to the LIN RKA security of RE.

(2) We will need the following observation which follows from the linear structure of the LPN based encryption PE. For every key S_1 and internal randomness r the inner encryption $\mathsf{RE}_{S_1}(X;r)$ can be written as an (extended) linear mapping $\psi_{M,T} : X \to M + TX$ where M and T can be computed based on S_1 and r via some efficiently computable mapping ρ. Using this observation, we show that the double encryption Enc inherits (extended) LIN KDM security from the outer encryption KE.

Formally, let $E\mathsf{Real}_S$ and $E\mathsf{Fake}_S$ be the real/fake KDM oracles for an encryption E defined in Lemma 5.2. Let $\mathcal{A}^{\mathcal{O}}_{S_1}$ be an oracle-aided randomized function which, given an extended LIN KDM query $\psi_{M,T}$, samples randomness r for the inner encryption RE, computes $(M',T') = \rho(S_1,r)$, and queries the oracle $\mathcal{O}$ with the composed linear function $\psi : S_2 \to \psi_{M',T'}(\psi_{M,T}(S_1,S_2))$. It is not hard to see that ψ is indeed an extended linear function, and for random (S_1,S_2)

$$\mathsf{EncReal}_{S_1,S_2} \equiv \mathcal{A}^{\mathsf{KEReal}_{S_2}}_{S_1} \stackrel{c}{\equiv} \mathcal{A}^{\mathsf{KEFake}_{S_2}}_{S_1},$$

where the first transition is due to the definition of $\mathcal{A}$ (and holds for every (S_1,S_2)) and the second transition follows from the security of KE.

To complete the proof, define an oracle-aided randomized function $\mathcal{B}^{\mathcal{O}}_{S_2}$ which given a LIN KDM query $\psi_{M,T}$ outputs $\mathcal{O}(\mathsf{Enc}_{S_2}(0^{|M|}))$. For random (S_1,S_2) we have that

$$\mathcal{A}^{\mathsf{KEFake}_{S_2}}_{S_1} \equiv \mathcal{B}^{\mathsf{KEFake}_{S_2}}_{S_1} \stackrel{c}{\equiv} \mathcal{B}^{\mathsf{KEReal}_{S_2}}_{S_1} \equiv \mathsf{EncFake}_{S_1,S_2},$$

and item (2) follows.

(3) We show that, given an access to the real LIN RK-KDM oracle $\mathsf{EncReal}_{S_1,S_2}$, it is possible to fully recover the key (S_1, S_2). First use RKA queries to fully recover the key S_2 via the attack described in Section 5.2. Second, in order to recover S_1, apply a KDM query to obtain an encryption C of (S_1, S_2), and use the decryption algorithm KD_{S_2} to decrypt the ciphertext C. We claim that the resulting value is simply (S_1, S_2). Indeed, by the definition of RE we have that

$$C = \mathsf{Enc}_{S_1,S_2}(S_1, S_2) = \mathsf{KE}_{S_2}(\mathsf{RE}_{S_1}(S_1, S_2)) = \mathsf{KE}_{S_2}(S_1, S_2)$$

and therefore $\mathsf{KD}_{S_2}(C) = (S_1, S_2)$ and the lemma follows. □

6 Conclusion

We defined a new combined form of RKA-KDM security, proved that such an encryption scheme can be realized based on the LPN assumption, and showed that the free-XOR technique can be securely instantiated with it. Altogether, our results enable a realization of the free-XOR optimization in the standard model under a well-studied cryptographic assumption.

The new definition of RKA-KDM security further motivates the study of security under related-key and key-dependent attacks. Specifically, in light of our counter-example, it is is natural to ask whether LIN RKA-KDM security can be constructed based on some combination of an RKA-secure scheme and a KDM-secure scheme, or better yet, based on more general assumptions (e.g., CPA-secure encryption scheme). It will also be interesting to find additional applications of RKA/KDM secure primitives.

References

[1] Applebaum, B.: Randomly encoding functions: A new cryptographic paradigm - (invited talk). In: ICITS. pp. 25–31 (2011)

[2] Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Advances in Cryptology - CRYPTO 2009 (2009)

[3] Applebaum, B., Harnik, D., Ishai, Y.: Semantic security under related-key attacks and applications. In: ICS. pp. 45–60 (2011)

[4] Applebaum, B., Ishai, Y., Kushilevitz, E.: Computationally private randomizing polynomials and their applications. Computional Complexity 15(2), 115–162 (2006), preliminary version in Proc. 20th CCC, 2005.

[5] Applebaum, B., Ishai, Y., Kushilevitz, E.: How to garble arithmetic circuits. In: FOCS '11. pp. 120–129 (2011)

[6] Bellare, M., Cash, D.: Pseudorandom functions and permutations provably secure against related-key attacks. In: Advances in Cryptology - CRYPT0 '10 (2010)

[7] Bellare, M., Hoang, V.T., Rogaway, P.: Garbling schemes. Cryptology ePrint Archive, Report 2012/265 (2012), `http://eprint.iacr.org/`

[8] Bellare, M., Kohno, T.: A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In: Advances in Cryptology - EUROCRYPT '03 (2003)

[9] Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: First ACM Conference on Computer and Communications Security. pp. 62–73 (1993)

[10] Black, J., Rogaway, P., Shrimpton, T.: Encryption-scheme security in the presence of key-dependent messages. In: SAC '02. pp. 62–75 (2002)

[11] Blum, A., Furst, M., Kearns, M., Lipton, R.J.: Cryptographic primitives based on hard learning problems. In: Advances in Cryptology - CRYPTO '93, pp. 278–291 (1993)

[12] Blum, Kalai, Wasserman: Noise-tolerant learning, the parity problem, and the statistical query model. JACM: Journal of the ACM 50(4), 506–519 (2003)

[13] Blum, M., Micali, S.: How to generate cryptographically strong sequences of pseudo-random bits. SIAM J. Comput. 13, 850–864 (1984), preliminary version in Proc. 23rd FOCS, 1982

[14] Boneh, D., Halevi, S., Hamburg, M., Ostrovsky, R.: Circular-secure encryption from decision diffie-hellman. In: Cryptographic primitives based on hard learning problems - CRYPTO '08. pp. 108–125 (2008)

[15] Camenisch, Lysyanskaya: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Cryptographic primitives based on hard learning problems - EUROCRYPT '01 (2001)

[16] Canetti, Goldreich, Halevi: The random oracle methodology, revisited. JACM: Journal of the ACM 51 (2004)

[17] Choi, S.G., Katz, J., Kumaresan, R., Zhou, H.S.: On the security of the "free-XOR" technique. In: TCC '12 (2012)

[18] Gilbert, H., Robshaw, M.J.B., Seurin, Y.: How to encrypt with the LPN problem. In: Automata, Languages and Programming, 35th International Colloquium, ICALP '08. pp. 679–690 (2008)

[19] Goldreich, O., Krawczyk, H., Luby, M.: On the existence of pseudorandom generators. SIAM J. Comput. 22(6), 1163–1175 (1993), preliminary version in Proc. 29th FOCS, 1988

[20] Goldreich, O., Micali, S., Wigderson, A.: How to play ANY mental game. In: STOC '87. pp. 218–229 (1987)

[21] Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM J. Comput. 28(4), 1364–1396 (1999)

[22] Henecka, W., Kögl, S., Sadeghi, A.R., Schneider, T., Wehrenberg, I.: TASTY: tool for automating secure two-party computations. In: CCS 10'. pp. 451–462 (2010)

[23] Huang, Y., Evans, D., Katz, J., Malka, L.: Faster secure two-party computation using garbled circuits. In: USENIX Security Symposium (2011)

[24] Huang, Y., Shen, C.H., Evans, D., Katz, J., Shelat, A.: Efficient secure computation with garbled circuits. In: ICISS '11. pp. 28–48 (2011)

[25] Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: CRYPTO '03. pp. 145–161 (2003)

[26] Ishai, Y., Kushilevitz, E.: Randomizing polynomials: A new representation with applications to round-efficient secure computation. In: Proc. 41st FOCS. pp. 294–304 (2000), `citeseer.nj.nec.com/ishai00randomizing.html`

[27] Kolesnikov, V., Sadeghi, A.R., Schneider, T.: Improved garbled circuit building blocks and applications to auctions and computing minima. In: CANS. pp. 1–20 (2009)

[28] Kolesnikov, V., Schneider, T.: Improved garbled circuit: Free XOR gates and applications. In: Automata, Languages and Programming, 35th International Colloquium, ICALP '08. pp. 486–498 (2008)

[29] Kreuter, B., Shelat, A., Shen, C.H.: Billion-gate secure computation with malicious adversaries. In: Security'12: Proceedings of the 21st USENIX conference on Security symposium. pp. 14-14 (2012)

[30] Lindell, Y., Pinkas, B., Smart, N.: Implementing two-party computation efficiently with security against malicious adversaries. In: SCN '08. pp. 2–20 (Sep 2008)

[31] Lindell, Y., Pinkas, B.: An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: Advances in Cryptology - EUROCRYPT '07. pp. 52–78. Springer (2007)

[32] Lindell, Y., Pinkas, B.: A proof of security of yao's protocol for two-party computation. J. Cryptology 22(2), 161–188 (2009)

[33] Malka, L., Katz, J.: Vmcrypt - modular software architecture for scalable secure computation. Cryptology ePrint Archive, Report 2010/584 (2010), `http://eprint.iacr.org/`

[34] Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay — A secure two-party computation system. In: Proc. of 13th USENIX Security Symposium (2004)

[35] Maurer, U.M.: Indistinguishability of random systems. In: EUROCRYPT. pp. 110–132 (2002)

[36] Naor, M., Pinkas, B.: Oblivious transfer with adaptive queries. In: CRYPTO: Proceedings of Crypto (1999)

[37] Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: Proc. 1st ACM Conference on Electronic Commerce. pp. 129–139 (1999)

[38] Nielsen, J.B., Orlandi, C.: LEGO for two-party secure computation. In: Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009. pp. 368–386 (2009)

[39] Pinkas, B., Schneider, T., Smart, N., Williams, S.: Secure two-party computation is practical. In: Advances in Cryptology – ASIACRYPT 2009. pp. 250–267 (2009)

[40] Rogaway, P.: The Round Complexity of Secure Protocols. Ph.D. thesis, MIT (June 1991)

[41] Shelat, A., Shen, C.H.: Two-output secure computation with malicious adversaries. In: EUROCRYPT. pp. 386–405 (2011)

[42] Yao, A.C.: Theory and application of trapdoor functions. In: Proc. 23rd FOCS. pp. 80–91 (1982)

[43] Yao, A.C.: How to generate and exchange secrets. In: Proc. 27th FOCS. pp. 162–167 (1986)