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 been employed in several practical implementations leading to notable performance improvements.

The security of the free-XOR optimization was originally proven in the random oracle model. In the same paper, Kolesnikov and Schneider also addressed the question of replacing the random oracle with a standard cryptographic assumption and suggested to use a hash function which achieves some form of security under correlated inputs. This claim was revisited by Choi et al. (TCC 2012) who showed that a stronger form of security is required, and proved that the free-XOR optimization can be realized based on a new primitive called *circular 2-correlation hash function*. Unfortunately, it is currently unknown how to implement this primitive based on standard assumptions, and so the feasibility of realizing the free-XOR optimization in the standard model remains an open question.

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

1 Introduction

Yao's *garbled circuit* (GC) construction [51] 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 [16, 50], or equivalently a one-way function [27].

The GC construction was originally motivated by the problem of secure multiparty computation [50, 24]. Along the years, the GC construction has found a diverse range of other applications to problems such as computing on encrypted data [48, 17], parallel cryptography [5, 6], verifiable computation [21, 7], software protection [25], functional encryption [47, 26], and key-dependent message security [9, 1]. Despite its theoretical importance, GC was 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

*School of Electrical Engineering at Tel-Aviv University, Email: bennyap@post.tau.ac.il.

a fruitful line of works that optimizes the concrete efficiency of GC-based protocols up to a level that suits large-scale practical applications [43, 40, 37, 36, 45, 44, 28, 29, 49, 30, 35].

Among other things, all current implementations of GCs (e.g., [45, 28, 39, 49, 29]) employ the so-called *free-XOR* optimization of Kolesnikov and Schneider [34]. 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 [34, 33, 45].

As in many cases, this efficiency gain has a cost in terms of the underlying 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 [13]. Due to the known limitation of the random oracle model [19], 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 [34] and was further studied in [4, 20]. In [34] 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* [31], 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. [20] 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 [18, 14]. While the existence of circular correlation-robust hash functions (a new primitive introduced by Choi et al. [20]) 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 approach in the standard model.

1.1 Our Contribution

We resolve the above feasibility question by showing that the free-XOR approach can be realized in the standard model under the learning parity with noise (LPN) assumption [23, 15]. 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. Our main result is derived in two steps:

1. We prove that the free-XOR construction is secure when instantiated with a semantically-secure symmetric encryption scheme whose security is preserved under both related-key attacks and key-dependent message attacks. More precisely, we require security against a combined form of attack in which the adversary is allowed to see ciphertexts of the form $\mathsf{Enc}_{K\oplus\Delta_1}(K \oplus \Delta_2)$ where K is the secret key and the shifts Δ_1 and Δ_2 are chosen by the adversary. This notion, referred to as ***RK-KDM security*** (with respect to linear functions), strictly generalizes the previous definitions of semantic security under related key attacks [4] and key-dependent message attacks [18, 14].

2. We show that the LPN-based symmetric encryption of [22] and its generalization [3] 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 [3, 10, 4]

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

Encryption vs. Hashing. The key point in which we deviate from [34, 20] 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. [37, 6] and the recent discussion in [11]), 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 [4].) 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* [32] as done in [6]. Our proofs follows 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 [6] to derive a secure MPC protocol. Similarly, all other applications (cf. [2]) 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, [19] 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 optimization, the first step was essentially taken by [20] 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 [18, 14] and [12]. Both notions were extensively studied (separately) during

the last decade. Most relevant to this paper is our joint work with Harnik and Ishai [4]. 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., [42, 31]), can be securely instantiated with RKA-secure encryption schemes. In addition, [4] suggested to apply a similar modification to the free-XOR variant, which was believed to be secure when instantiated with correlation-robust hash functions [34]. As mentioned, the latter claim was found to be inaccurate, and therefore the results of [4] cannot be used in the context of the free-XOR approach. (The other applications mentioned in [4] 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. Finally, we end with a short conclusion in Section 5.

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 [41]. 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 $f \stackrel{c}{\equiv} g$, 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. [41]).

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

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

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

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)

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

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 strictly generalizes semantic security under related-key attacks [4] and semantic security under key-dependent message attacks [14]. 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, it seems that a scheme may satisfies both RKA security and KDM security without achieving the combined form of RKA-KDM security.

[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.

- **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 $M \in \mathbb{F}_2^*$ 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 $0 < \varepsilon < \frac{1}{2}$. The input to the problem is a random matrix $A \xleftarrow{R} \mathbb{F}_2^{t \times k}$ and a vector $y = As + e \in \mathbb{F}_2^t$ where $s \xleftarrow{R} \mathbb{F}_2^k$ and each entry of e is chosen independently according to the error distribution $\mathsf{Ber}_\varepsilon^m$ in which each entry is chosen to be 1 independently 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 a random information word s given a noisy codeword y.) For polynomially bounded integer 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 LPN_ε is *hard* if $\mathsf{LPN}_{t,\varepsilon}$ is hard for every polynomial $t(\cdot)$. We describe the symmetric encryption scheme of [3] which is a variant of the scheme of [22].

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*

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

[2] A seemingly weaker definition of LIN RK-KDM security restricts the KDM family to functions $g_{M,\sigma} : s \mapsto (M + (\sigma \cdot s))$. If M is longer than k where M and s are of the same length. 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 two 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 $(f \in \Phi_{\mathsf{RKA}}, g_{M_i,0})$.

- 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 [3] 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 LPN_ε problem [3]. Security against KDM and RKA attacks with respect to linear functions was further proven in [3] and [4]. We now generalize thses results and show that the scheme is LIN RK-KDM secure.

Lemma 3.3. *Assuming that LPN_ε is hard, the above construction is LIN RK-KDM secure.*

3.2 Proof of Lemma 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 claim (which is implicit in [3]) shows that the LPN encryption scheme is not only semantically secure but also pseudorandom in the following sense:

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

We will need the following key observation:

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

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

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

$$Z = (A+GH)\cdot S+E+G\cdot M+A\Delta = A\cdot(S+\Delta)+E+G\cdot(M+HS) = A\cdot(S+\Delta)+E+G\cdot(M+bS)$$

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 $S \xleftarrow{R} \mathbb{F}_2^{k\times N}$,

$$\mathsf{Real}_S \equiv F^{\mathsf{Enc}_S} \stackrel{c}{\equiv} F^{\mathcal{R}_S} \equiv \mathcal{R}_S. \tag{1}$$

Indeed, the first and third transitions are due to Claim 3.5, and the second transition is due to Claim 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:

$$\mathcal{R}_S \equiv F(\cdot, 0^{\ell\times N}, 0)^{\mathcal{R}_S} \stackrel{c}{\equiv} F(\cdot, 0^{\ell\times N}, 0)^{\mathsf{Enc}_S(0^{\ell\times N})} \equiv \mathsf{Real}_S(\cdot, 0^{\ell\times N}, 0) \equiv \mathsf{Fake}_S, \tag{2}$$

where the first and third transitions are due to Claim 3.5, and the second transition is due to Claim 3.4 and Fact 2.2. By combining Eq. 1 and Eq. 2 with Fact 2.3 we get that $\mathsf{Real}_S \stackrel{c}{\equiv} \mathsf{Fake}_S$, and Lemma 3.3 follows.

Remark 3.6 (Abstraction)**.** *The proof of Lemma 3.3 provides a general template for proving RKA-KDM security. Specifically, the properties needed are pseudorandomness (in the sense of Claim 3.4) and key/message homomorphism (in the sense of Claim 3.5). Indeed, observe that, apart from the proofs of Claims 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 [51] 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 [6], 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 our 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 a* compiler *that given a circuit C outputs the encoding $\hat{f}$, the recovery algorithm* Rec *and the simulator* Sim*, represented as circuits. (See [8] 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 [46, 43, 38, 6].

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

$$E^2_{K_1,K_2}(M) := (\mathsf{Enc}_{K_1}(R), \mathsf{Enc}_{K_2}(R+M)), \quad D^2{}_{K_1,K_2}(C_1,C_2) := \mathsf{Dec}_{K_1}(C_1) + \mathsf{Dec}_{K_2}(C_2) \quad (3)$$

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 $r_i \stackrel{R}{\leftarrow} \{0,1\}$ 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 idea is to let the encoding $\hat{f}_n(x;(W,r))$ reveal only the value of the active keys $W_i^{b_i}$ and their colors c_i. This is done by traversing the circuit from inputs to outputs: first the encoding reveals the active keys of the inputs; in addition, for each gate, the encoding provides a mechanism that translates the active keys of the input wires into the active keys of the output wires. Specifically, for each Binary gate $g(\cdot,\cdot)$ (e.g., AND) the encoding outputs an encryption tables (or "gate labels") in which the keys of the outgoing wire W_ℓ^0, W_ℓ^1 are encrypted under the keys of the incoming wires i, j. Hence, one can propagate the values of $W_i^{b_i}$ from the inputs to the outputs. It is crucial to observe that the values of the active keys $W_i^{b_i}$ and their colors c_i reveal nothing on their semantics b_i. Only for the output wires, we reveal the coloring r_i, which makes it possible to recover the value of the i-th output wire 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* $\Pr[\mathsf{Rec}(1^n, \hat{f}_n(x; (r, W))) \neq f_n(x)] = 0$*.*

Proof. Let $\alpha = \hat{f}_n(x; (r, W))$ for some $x \in \{0,1\}^n$ and $(r, W) \in \{0,1\}^{\mu(n)}$. It suffices to show that, given α, it is possible to recover the active key $W_i^{b_i}$ of every wire i together with its color $c_i = (b_i(x) + r_i)$. Indeed, once these values are known we can easily recover all the outputs of $f_n(x)$: For every output wire j, we recover b_j by XOR-ing c_j with the mask r_j which is given explicitly as part of α.

The active keys and their colors are computed by scanning the circuit from bottom to top as follows. For an input wire i the desired value, $W_i^{b_i} \circ c_i$, is given as part of α. Next, consider a wire y that goes out of a gate t, and assume that we have already computed the desired values of the input wires i and j of this gate. 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 Equation (4); 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 gate t, which satisfies, by definition, the equality $b_y = g(b_i, b_j)$. □

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:

$$Q_t^{a_i,a_j} := E^2_{W_i^{a_i+r_i}, W_j^{a_j+r_j}}\Big(W_y^{g(a_i+r_i,a_j+r_j)} \circ (g(a_i + r_i, a_j + r_j) + r_y)\Big), \tag{4}$$

   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. 3.

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 $(\mathsf{Enc}, \mathsf{Dec})$. 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 $W_\ell^{b_\ell} \stackrel{R}{\leftarrow} \{0,1\}^k$ and a random color bit $c_\ell \stackrel{R}{\leftarrow} \{0,1\}$.
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 $r_\ell = c_\ell - b_\ell(x)$ (recall that x is known).
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
$$Q_t^{c_i,c_j} := E^2_{W_i^{b_i}, W_j^{b_j}}(W_y^{b_y} \circ c_y) \tag{5}$$
6. Compute the inactive labels as follows. For every $(\alpha, \beta) \neq (0,0)$ choose $R_{\alpha,\beta} \stackrel{R}{\leftarrow} \{0,1\}^{k+1}$ 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

$$\begin{aligned} Q_t^{c_i+1,c_j} &:= \left(\mathcal{O}(\phi_i, \psi_{1,0}), \mathsf{Enc}_{W_j^{b_j}}(R_{1,0})\right) \\ Q_t^{c_i+1,c_j+1} &:= \left(\mathcal{O}(\phi_i, \psi_{1,1}), \mathcal{O}(\phi_j, R_{1,1})\right) \\ Q_t^{c_i,c_j+1} &:= \left(\mathsf{Enc}_{W_i^{b_i}}(R_{0,1}), \mathcal{O}(\phi_j, \psi_{0,1})\right), \end{aligned} \tag{6}$$

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. 6 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 $\left\{H^{\mathsf{Real}_s}\right\}_s$ and $\left\{H^{\mathsf{Fake}_s}\right\}_s$ 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. 5. 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

$$\begin{aligned} Q_t^{c_i+1,c_j} &:= \left(\mathsf{Enc}_{W_i^{b_i+1}}(0^{k+1}), \mathsf{Enc}_{W_j^{b_j}}(R_{1,0})\right) \\ Q_t^{c_i+1,c_j+1} &:= \left(\mathsf{Enc}_{W_i^{b_i+1}}(0^{k+1}), \mathsf{Enc}_{W_j^{b_j+1}}(0^{k+1})\right) \\ Q_t^{c_i,c_j+1} &:= \left(\mathsf{Enc}_{W_i^{b_i}}(R_{0,1}), \mathsf{Enc}_{W_j^{b_j+1}}(0^{k+1})\right). \end{aligned} \tag{7}$$

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. 7) are computed exactly as they are computed by $H^{\mathsf{Fake}_s(\cdot)}(x)$ (i.e., as defined in Eq. 6 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 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 approach 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, and raises several interesting questions. Specifically, it will be interested to understand the exact power of combined RKA-KDM in comparison to RKA and KDM security, to discover constructions which are based on abstract assumptions (e.g., CPA-secure encryption scheme), and to find additional applications of RKA/KDM secure primitives.

References

[1] Benny Applebaum. Key-dependent message security: Generic amplification and completeness theorems. In *EUROCRYPT*, pages 527–546, 2011.

[2] Benny Applebaum. Randomly encoding functions: A new cryptographic paradigm - (invited talk). In *ICITS*, pages 25–31, 2011.

[3] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *Advances in Cryptology - CRYPTO 2009*, 2009.

[4] Benny Applebaum, Danny Harnik, and Yuval Ishai. Semantic security under related-key attacks and applications. In *ICS*, pages 45–60, 2011.

[5] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *Proc. 45th FOCS*, pages 166–175, 2004. To appear in SIAM J. Comput.

[6] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Computional Complexity*, 15(2):115–162, 2006. Preliminary version in Proc. 20th CCC, 2005.

[7] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In *ICALP (1)*, pages 152–163, 2010.

[8] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. In *FOCS*, pages 120–129, 2011.

[9] Boaz Barak, Iftach Haitner, Dennis Hofheinz, and Yuval Ishai. Bounded key-dependent message security. In *Advances in Cryptology - EUROCRYPT 2010*, pages 423–444, 2010.

[10] Mihir Bellare and David Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In *Advances in Cryptology - CRYPT0 2010*, 2010.

[11] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Garbling schemes. Cryptology ePrint Archive, Report 2012/265, 2012. `http://eprint.iacr.org/`.

[12] Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In *EUROCRYPT: Advances in Cryptology: Proceedings of EUROCRYPT*, 2003.

[13] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73, Fairfax, 1993. ACM.

[14] John Black, Phillip Rogaway, and Thomas Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In *SAC '02*, pages 62–75, 2002.

[15] Avrim Blum, Merrick Furst, Michael Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In *Advances in Cryptology: Proc. of CRYPTO '93*, volume 773 of *LNCS*, pages 278–291, 1994.

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

[17] Christian Cachin, Jan Camenisch, Joe Kilian, and Joy Müller. One-round secure computation and secure autonomous mobile agents. In *ICALP*, pages 512–523, 2000.

[18] Camenisch and Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT '01*, 2001.

[19] Canetti, Goldreich, and Halevi. The random oracle methodology, revisited. *JACM: Journal of the ACM*, 51, 2004.

[20] Seung Geol Choi, Jonathan Katz, Ranjit Kumaresan, and Hong-Sheng Zhou. On the security of the "free-XOR" technique. In *TCC*, 2012.

[21] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO*, 2010.

[22] Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. How to encrypt with the LPN problem. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008*, pages 679–690, 2008.

[23] Oded Goldreich, Hugo Krawczyk, and Michael Luby. On the existence of pseudorandom generators. *SIAM J. Comput.*, 22(6):1163–1175, 1993. Preliminary version in Proc. 29th FOCS, 1988.

[24] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play ANY mental game. In *STOC*, pages 218–229, 1987.

[25] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *STOC*, 2008.

[26] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *CRYPTO*, pages 162–179, 2012.

[27] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[28] Wilko Henecka, Stefan Kögl, Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. TASTY: tool for automating secure two-party computations. In *CCS*, pages 451–462, 2010.

[29] Yan Huang, David Evans, Jonathan Katz, and Lior Malka. Faster secure two-party computation using garbled circuits. In *USENIX Security Symposium*, 2011.

[30] Yan Huang, Chih-Hao Shen, David Evans, Jonathan Katz, and Abhi Shelat. Efficient secure computation with garbled circuits. In *ICISS*, pages 28–48, 2011.

[31] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *CRYPTO*, pages 145–161, 2003.

[32] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *Proc. 41st FOCS*, pages 294–304, 2000.

[33] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In *CANS*, pages 1–20, 2009.

[34] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008*, pages 486–498, 2008.

[35] Benjamin Kreuter, Abhi Shelat, and Chih-Hao Shen. Towards billion-gate secure computation with malicious adversaries. *IACR Cryptology ePrint Archive*, 2012:179, 2012.

[36] Y Lindell, B Pinkas, and Nigel Smart. Implementing two-party computation efficiently with security against malicious adversaries. In *SCN 2008*, pages 2–20, September 2008.

[37] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *Advances in Cryptology - EUROCRYPT 2007*, pages 52–78. Springer, 2007.

[38] Yehuda Lindell and Benny Pinkas. A proof of security of yao's protocol for two-party computation. *J. Cryptology*, 22(2):161–188, 2009.

[39] Lior Malka and Jonathan Katz. Vmcrypt - modular software architecture for scalable secure computation. Cryptology ePrint Archive, Report 2010/584, 2010. `http://eprint.iacr.org/`.

[40] Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay — A secure two-party computation system. In *Proc. of 13th USENIX Security Symposium*, 2004.

[41] Ueli M. Maurer. Indistinguishability of random systems. In *EUROCRYPT*, pages 110–132, 2002.

[42] M. Naor and B. Pinkas. Oblivious transfer with adaptive queries. In *CRYPTO: Proceedings of Crypto*, 1999.

[43] M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *Proc. 1st ACM Conference on Electronic Commerce*, pages 129–139, 1999.

[44] Jesper Buus Nielsen and Claudio Orlandi. LEGO for two-party secure computation. In *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009*, pages 368–386, 2009.

[45] Benny Pinkas, Thomas Schneider, Nigel Smart, and Stephen Williams. Secure two-party computation is practical. In *Advances in Cryptology – ASIACRYPT 2009*, pages 250–267, 2009.

[46] Phillip Rogaway. *The Round Complexity of Secure Protocols.* PhD thesis, MIT, June 1991.

[47] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *ACM Conference on Computer and Communications Security*, pages 463–472, 2010.

[48] Tomas Sander, Adam Young, and Moti Yung. Non-interactive cryptocomputing for NC^1. In *FOCS*, pages 554–567, 1999.

[49] Abhi Shelat and Chih-Hao Shen. Two-output secure computation with malicious adversaries. In *EUROCRYPT*, pages 386–405, 2011.

[50] A. C. Yao. Theory and application of trapdoor functions. In *Proc. 23rd FOCS*, pages 80–91, 1982.

[51] A. C. Yao. How to generate and exchange secrets. In *Proc. 27th FOCS*, pages 162–167, 1986.