Efficient and Secure Multiparty Computation from Fixed-Key Block Ciphers

Chun Guo
Université Catholique de Louvain
chun.guo.sc@gmail.com

Jonathan Katz
University of Maryland
jkatz@cs.umd.edu

Xiao Wang
Northwestern University
wangxiao@cs.northwestern.edu

Yu Yu
Shanghai Jiao Tong University
yuyu@cs.sjtu.edu.cn

***Abstract*—Many implementations of secure computation use fixed-key AES (modeled as a random permutation); this results in substantial performance benefits due to existing hardware support for AES and the ability to avoid recomputing the AES key schedule. Surveying these implementations, however, we find that most utilize AES in a heuristic fashion; in the best case this leaves a gap in the security proof, but in many cases we show it allows for explicit attacks.**

Motivated by this unsatisfactory state of affairs, we initiate a comprehensive study of how to use fixed-key block ciphers for secure computation—in particular for *OT extension* and *circuit garbling*—efficiently and securely. Specifically:

- **We consider several notions of pseudorandomness for hash functions (e.g., correlation robustness), and show provably secure schemes for OT extension, garbling, and other applications based on hash functions satisfying these notions.**
- **We provide provably secure constructions, in the (non-programmable) random-permutation model, of hash functions satisfying the different notions of pseudorandomness we consider.**

Taken together, our results provide end-to-end security proofs for implementations of secure-computation protocols based on fixed-key block ciphers (modeled as random permutations). Perhaps surprisingly, at the same time our work also results in noticeable performance improvements over the state-of-the-art.

I. Introduction

Over the past few years, secure computation has transitioned from the realm of pure theory to the point where it is implemented in multiple software libraries (see [26] for a recent survey), funded by various government agencies, marketed by many startup companies, and used in several real-world applications [10, 9, 32, 27]. This makes it critical to understand the security provided by *implementations*[1] of secure-computation protocols. Indeed, although published protocols typically come with proofs of security that can be independently verified by the community, published protocol descriptions often omit or ignore important low-level details, and researchers often apply performance optimizations in a haphazard way when implementing those protocols.

In this work we study the use of fixed-key AES (or fixed-key block ciphers more generally) in implementations of secure computation. Using fixed-key AES in this context can be traced back to the work of Bellare et al. [5], who consider fixed-key AES for circuit garbling as part of their JustGarble framework. Prior to that point, most implementations of garbled circuits used a hash function such as SHA-256, modeled as a random oracle. Bellare et al. design several provably secure garbling schemes based on a fixed-key block cipher, and in doing so demonstrated significant performance improvements; in particular, they show that using fixed-key AES can be up to 50× faster than using a cryptographic hash function due to the hardware support for AES provided by modern processors. (While it is also possible to garble circuits using AES with different keys at each gate, the added overhead of key scheduling is substantial.) Prior to their work *CPU time* was the main bottleneck for protocols based on circuit garbling; after the introduction of JustGarble, *network throughput* became the dominant factor. For this reason, fixed-key AES has been almost universally adopted in subsequent implementations of garbled circuits.

Bellare et al. prove security of their constructions when the fixed-key block cipher is modeled as a random permutation. This *random-permutation model (RPM)* is analogous to the random-oracle model, and assumes that all parties have oracle access to a public, random permutation $\pi : \{0,1\}^k \to \{0,1\}^k$ and its inverse. Modeling a fixed-key block cipher as a random permutation is weaker than modeling the block cipher as an ideal cipher (which is also common); in particular, related-key attacks do not apply in the fixed-key setting. Inspired by the work of Bellare et al., many subsequent works on efficient secure computation have relied on fixed-key AES for the purposes of both garbling (e.g., [53]) and *oblivious-transfer (OT) extension* (e.g., [46]), two important building blocks for secure-computation protocols. Unfortunately, however, end-to-end proofs of security are often missing. For example, published OT-extension protocols may be based on a hash function H and are proven secure by modeling H as a random oracle. When the protocols are implemented, however, H is *not* instantiated using a cryptographic hash function but is instead instantiated haphazardly from a fixed-key block cipher. At best this leaves a gap in the security proofs, but at worst—as we show in Section II—it leaves the implemented protocols vulnerable to explicit attacks.

Even the work of Bellare et al. has the drawback that it is not *modular*. That is, they do not prove security of their garbling schemes based on some assumption about the "encryption scheme" used for each gate; instead, they prove security directly in the random-permutation model. This makes it difficult to apply their ideas to new garbling schemes that are developed, as any changes in the scheme require re-doing the entire proof. This was done, for example, in the analysis of the subsequent half-gates garbling scheme [53]. In their paper introducing the scheme, Zahur et al. follow a more modular approach: they construct their garbling scheme based on a hash function H and then prove that their scheme is

[1] We stress that we are *not* referring to general software-security issues such as improper input handling or buffer-overflow attacks (though these are also important). Rather, the focus of this work is on *cryptographic* issues that arise in the course of implementation.

DOI 10.1109/SP40000.2020.00016

secure if H satisfies a certain property specific to their scheme. They also claim without proof that when H is instantiated in a particular way based on a fixed-key cipher, then H satisfies their definition. This is a step forward, but still leaves a gap in the overall security proof.

We believe this state of affairs is unsatisfactory. In a nutshell, the problem is that protocols are generally analyzed by cryptographers based on a hash function H viewed as a random oracle, but when these protocols are implemented then H is instantiated from a fixed-key block cipher in some unprincipled (and often insecure) way. In this paper, we attempt to resolve this mismatch as described next.

A. Our Contributions

Deficiencies of current implementations. As already hinted at above, our first contribution is merely identifying the problem. We examined all state-of-the-art platforms for secure computation[2] and found that most of those using fixed-key AES were using it incorrectly; of those, some are vulnerable to explicit attacks. We refer to Section II for details.

Faced with this chaotic status quo, we initiate a comprehensive study of how to securely and efficiently use fixed-key block ciphers for secure computation. We propose a modular approach: we first identify various properties that hash functions need to satisfy in order to prove different protocols secure, and then show how to efficiently construct hash functions provably satisfying those properties in the random-permutation model. In more detail:

Identifying abstract security properties. We consider several notions of pseudorandomness[3] for hash functions, some that were identified in previous work (e.g., correlation robustness [28] and circular correlation robustness [14]) and others that are new. We then show how hash functions realizing these different notions can be used in a provably secure way for various flavors of OT extension (Section IV) as well as for circuit garbling (Section V) and other applications (Section VI).

Realizations from fixed-key block ciphers. We show provably secure constructions, in the non-programmable RPM, of hash functions meeting the notions we consider (Section VII). Importantly, in our analyses we also provide concrete security bounds, something that is often lacking in prior work. We also discuss how to efficiently implement our constructions utilizing existing CPU instruction sets and pipelining (Section VIII).

Taken together, our work provides *end-to-end security proofs* for secure-computation protocols based on fixed-key block ciphers (in the OT-hybrid[4] model). Somewhat surprisingly, at the same time our work also results in performance improvements of up to 3–4× over the current state-of-the-art for OT extension and various other protocols; we refer to Section VIII for further discussion.

B. Alternate Approaches

Recall that the problem we are addressing is that protocols are proven secure assuming access to a random oracle H, but then implemented with H instantiated improperly from a fixed-key block cipher. The approach we advocate, outlined in the previous section, is to prove protocols secure using *weaker* assumptions on H (in particular, falsifiable assumptions) and then to provably realize these assumptions in the random-permutation model. But one could imagine other ways of trying to solve the problem; these are discussed briefly here.

One option is to simply instantiate H using a cryptographic hash function like SHA-256 or SHA-3, treating those as random oracles. The drawback of this approach is that such hash functions are 15–50× slower than using fixed-key AES; see Table III.

Alternately, one could hope to instantiate H based on a fixed-key block cipher such that H is indifferentiable [38] from a random oracle. This problem has attracted a lot of attention [35, 18, 37, 39, 16, 33, 6] but, as we now discuss, existing solutions are unsatisfactory.

We are aware of only two approaches to constructing a random oracle H in the random-permutation model: the first corresponds roughly to letting $H(x)$ be the $(k - O(k))$-bit truncation of $\pi(0^{O(k)}\|x)$ (where we view k as a security parameter), and the second—called the XORP approach—defines $H(x) = \bigoplus_i \pi_i(x)$ where the $\{\pi_i\}$ represent independent random permutations. The first approach results in a hash function whose domain and range are much shorter than the domain/range of π, and would result in impractically poor security bounds if applied to AES with a block length of 128 bits. The second approach can be used to construct a random oracle mapping k-bit inputs to k-bit outputs given multiple random permutations on $\{0,1\}^k$, and thus could in principle be used to realize some of the security definitions we consider. Drawbacks of this approach, however, include:

- Some of our definitions require compression, and it is unknown how to obtain a compressing random oracle that is both efficient and has acceptable concrete-security bounds starting from a random permutation on $\{0,1\}^k$.
- The XORP approach requires calls to (at least) two independent permutations; in some sense, this is inherent [8, 36]. The constructions we show here use only a single random permutation (and in some cases just a single call to that permutation), and are more efficient than XORP; we defer further discussion to Section VIII.

C. Outline of the Paper

In Section II we survey existing implementations of secure-computation protocols, focusing in particular on how they instantiate the underlying hash function based on fixed-key AES, and show that in some cases they allow for an explicit attack. We introduce in Section III various security definitions

[2] See `https://github.com/rdragos/awesome-mpc`.

[3] The properties we consider are incomparable to traditional security definitions for hash functions such as collision-resistance; in fact, in some cases we do not even need the hash function to be compressing. Nevertheless, we use the term "hash function" since the properties we need would traditionally be achieved using a hash function modeled as a random oracle.

[4] We cannot hope to construct the "base" oblivious transfers from fixed-key block ciphers, as it is known that this requires stronger assumptions.

Platform	Security	Implementation
APRICOT [46]	Malicious	$H(x,i) = \pi(x)$
APRICOT [46]	Semi-honest	$H(x,i) = \pi(x)$
SPDZ-2 [47]	Malicious	$H(x,i) = \pi(x) \oplus x$
libOTe [44]	Malicious	$H(x,i) = \pi(x) \oplus x$
libOTe [44]	Malicious	$H(x,i) = \text{SHA-256}(x)$
Unbound Tech [45]	Malicious	$H(x,i) = \pi(x \oplus i) \oplus x \oplus i$
EMP [49]	Malicious	$H(x,i) = \pi(2x \oplus i) \oplus 2x \oplus i$

TABLE I: **Insecure instantiations of the hash function in existing implementations of OT extension.** $\pi(\cdot)$ denotes AES-128 with a fixed key.

for hash functions, some of which have been considered before (e.g., correlation robustness) and some of which have not (e.g., tweakable correlation robustness). In the following sections we explore applications of hash functions satisfying these definitions to OT extension (Section IV), the half-gates garbling scheme (Section V), and other protocols (Section VI). We address in Section VII the question of constructing hash functions satisfying the various definitions from a fixed-key block cipher, when that block cipher is modeled as a random permutation. Finally, in Section VIII we evaluate the performance of our constructions and compare them to prior work.

II. The Current State of Affairs

As discussed in Section I, while many of the existing platforms for secure computation rely on fixed-key AES, very few of them utilize it properly. Here we discuss some of the problems we found. Before making our results public, we contacted the authors of the affected works to inform them of the problems we found, to confirm our analysis, and to allow them time to patch their implementations. (Some platforms [49, 45] adopted the constructions we propose later in the paper, while others [44] found alternate solutions.)

A. Oblivious-Transfer Extension

Oblivious transfer (OT), introduced by Rabin in 1981 [43], is a key component of protocols for secure two-party computation [52] and, more generally, secure computation without an honest majority [24]—even in the semi-honest setting. In standard (one-out-of-two) OT, there is a sender who begins holding two messages m_0, m_1, and a receiver who holds a bit b; by running the protocol, the receiver learns m_b. Security requires that the sender learns nothing about b while the receiver learns nothing about m_{1-b}.

Oblivious transfer requires public-key cryptography and thus was, for a long time, a bottleneck in secure-computation protocols. This changed with the introduction of efficient *OT extension* [4, 28], which enables a small number (say, 128) of "base" OTs to be leveraged to give an unbounded number of OTs using only symmetric-key techniques. At a high level, state-of-the-art OT-extension protocols work in two phases:

1) A receiver with choice bits $x_1, ..., x_m$ runs a protocol with a sender, after which the sender obtains a uniform k-bit string Δ and uniform k-bit strings $(\mathbf{a}_1, \ldots, \mathbf{a}_m)$ while the receiver obtains k-bit strings $(\mathbf{b}_1, \ldots, \mathbf{b}_m)$ such that $\mathbf{a}_i \oplus \mathbf{b}_i = x_i\Delta$. (Here, k is a security parameter.) A malicious receiver is able to influence the randomness and thus can potentially control the $\{\mathbf{b}_i\}$.
2) The sender, with input messages $\{(\mathbf{m}_i^0, \mathbf{m}_i^1)\}_{i=1}^m$, computes and sends $\mathbf{c}_i^b := H(\mathbf{a}_i \oplus b \cdot \Delta,\ i) \oplus \mathbf{m}_i^b$ to the receiver, for $i = 1, \ldots, m$ and $b \in \{0,1\}$. The receiver can "decrypt" only one value in each tuple; namely, it can recover $\mathbf{m}_i^{x_i} := H(\mathbf{b}_i, i) \oplus \mathbf{c}_i^{x_i}$ for all i.

Assuming the first phase is carried out securely, it can be shown that the above protocol for OT extension is secure in both the semi-honest and malicious settings when H is a random oracle. Ishai et al. [28] showed that in the semi-honest setting it suffices for H to be correlation robust for random inputs (and in this case the additional input i to H is not needed). Asharov et al. [2] proved that the protocol is secure in the malicious setting if H is strongly correlation robust (cf. Definition 1), i.e., even for adversarially chosen inputs.

Prior to our work it was not clear how to construct a (strongly) correlation robust hash function from a fixed-key block cipher, and existing implementations made seemingly arbitrary choices (see next and Table I). Below we show that these lead to attacks on the OT-extension protocols, though it may not always imply an explicit attack on the overall protocol implementation (depending on how the base OTs are implemented). In what follows the block cipher is always AES-128, but we write π for consistency with the rest of the paper.

APRICOT and libscapi. APRICOT [46], which is also internally used by libscapi [3], sets $H(x,i) = \pi(x)$. This is insecure, even in the semi-honest case, since this H is invertible. (In particular, assume the receiver knows $\mathbf{m}_i^0$ and $\mathbf{m}_i^1$ for some i. Then it can deduce $\mathbf{a}_i$ and $\mathbf{a}_i \oplus \Delta$; hence it can recover Δ and learn all the rest of the sender's inputs.)

We believe their intent was to set $H(x,i) = \pi(x) \oplus x$. This was confirmed by the authors of APRICOT, and the latest version of their implementation has been updated to reflect this. As described next, however, this is still insecure.

SPDZ-2, MP-SPDZ, and MASCOT. The SPDZ-2 [47] implementation, which is also used by MP-SPDZ, sets $H(x,i) = \pi(x) \oplus x$ as proposed in the MASCOT paper [30]. This choice was justified there by noting that it is inspired by the Matyas-Meyer-Oseas (MMO) construction that is collision resistant in the ideal-cipher model. This reasoning is invalid since collision resistance does not imply correlation robustness. In any case, this instantiation admits a simple attack in the malicious setting that exploits the fact that H has no dependence on i: by using $x_1 = x_2 = 1$ and forcing $\mathbf{b}_1 = \mathbf{b}_2 = \mathbf{b}$, the receiver can learn $\mathbf{m}_1^1, \mathbf{m}_2^1$, and also

$$\mathbf{c}_1^0 \oplus \mathbf{c}_2^0 = H(\mathbf{b}, 1) \oplus H(\mathbf{b}, 2) = \mathbf{m}_1^0 \oplus \mathbf{m}_2^0,$$

which is disallowed.

Note that *any* instantiation of H that does not depend on i admits this attack in the malicious setting. In Appendix C, an attack in the $\mathcal{F}_{\Delta\text{-ROT}}$ hybrid model based on this idea that violates the privacy of the MASCOT protocol is presented.

libOTe. libOTe [44] provides two options for instantiating $H(x,i)$. The first option is identical to the one just discussed. The second option instantiates $H(x,i) = \text{SHA-256}(x)$. Besides the fact that this option no longer benefits from fixed-key AES, it also suffers from the same attack just described since there is no dependence on i.

Unbound Tech and EMP. The blockchain MPC implementation by Unbound Tech [45] sets $H(x,i) = \pi(x \oplus i) \oplus x \oplus i$. Although H now depends on i, a variant of the above attack still works if the malicious receiver chooses $\mathbf{b}_1, \mathbf{b}_2$ such that $\mathbf{b}_1 \oplus 1 = \mathbf{b}_2 \oplus 2$.

EMP [49] uses $H(x,i) = \pi(2x \oplus i) \oplus 2x \oplus i$, and a similar attack still applies.

The ABY framework. The ABY framework [17] also sets $H(x,i) = \pi(x \oplus i) \oplus x \oplus i$, but the above attack no longer applies since ABY targets semi-honest security. However, ABY implements correlated-OT extension and random-OT extension rather than standard-OT extension. Existing proofs of security for the former [1], even in the semi-honest setting, require H to be a random oracle (see further discussion in Section IV), while the ABY instantiation of H is clearly not indifferentiable from a random oracle since $H(x_1,i_1) = H(x_2,i_2)$ for any x_1, i_1, x_2, i_2 with $x_1 \oplus i_1 = x_2 \oplus i_2$.

B. Garbling

As noted in the Introduction, JustGarble [5] is a garbling scheme that is proven secure in the random-permutation model. The proof is non-modular, however, and so it is difficult to apply the techniques to newer garbling schemes. In analyzing their half-gates construction based on an abstract hash function H, Zahur et al. [53] introduce a definition called "circular correlation robustness for naturally derived keys" (see Section V) that is specific to their scheme, overly complicated, and difficult to work with. They then instantiate H as $H(x,i) = \pi(2x \oplus i) \oplus 2x \oplus i$, and claim without proof that this satisfies their definition.[5]

Zhu et al. [54] used a customized garbling scheme with the hash function instantiated as $H(x,i) = \pi(x \oplus i) \oplus x \oplus i$. Since the garbling scheme of Zhu et al. incorporates the free-XOR optimization [31], a proof of security requires H to satisfy a notion of *circular* correlation robustness [14]. However, we show in Section VII-C that the related hash function $H(x) = \pi(x) \oplus x$ is *not* circular secure (and the same applies to the hash function of Zhu et al. as well).

C. Other Protocols and Implementations

Although we focus primarily on OT extension and garbling in this paper, we observe that unprincipled reliance on fixed-key AES has come up in other scenarios as well.

TinyLEGO. Frederiksen et al. [22] showed that the wire-authentication protocol in TinyLEGO is secure if the hash function H being used is correlation robust for random inputs. In the implementation of TinyLEGO [41], they instantiated H as $H(x,i) = \pi(2x \oplus i) \oplus 2x \oplus i$, but there is no proof that this satisfies the required definition.

Free hash. Fan et al. [20] proposed a new way to "commit" to a garbled circuit. The proof of security for their construction assumes that the hash function H used is not only correlation robust but also *collision resistant*. Unfortunately, it is easy to see that the instantiation $H(x,i) = \pi(2x \oplus i) \oplus 2x \oplus i$ they use is not collision resistant (since $H(x_1,i_1) = H(x_2,i_2)$ when $2x_1 \oplus i_1 = 2x_2 \oplus i_2$), and this leads to an explicit attack on the binding property of their scheme.

III. Hash-Function Definitions

Here we define several notions of "pseudorandomness" for hash functions, some of which have been considered explicitly before. Our definitions are tailored for a concrete-security treatment, but asymptotic versions of our definitions can easily be obtained by suitable modifications. In what follows, we let $\mathsf{F}_{k,\ell}$ denote the set of all functions from $\{0,1\}^k$ to $\{0,1\}^\ell$, and write F_k for $\mathsf{F}_{k,k}$.

Our definitions are all phrased in the strongest sense possible—specifically, they allow the attacker to adaptively choose the inputs to its oracle—since our constructions satisfy them. For some applications, weaker notions (such as random inputs or non-adaptive choice of inputs) may suffice, and the definitions may be adapted appropriately for those cases.

Our definitions allow for non-uniform choice of the key R. This is useful in analyzing schemes like half-gates garbling [53], where the least-significant bit of R is set to 1.

Correlation robustness (cr). The notion of correlation robustness was first proposed by Ishai et al. [28] in the context of OT extension. Roughly, H is correlation robust if the keyed function $f_R(x) \stackrel{\text{def}}{=} H(x \oplus R)$ is pseudorandom. In the work of Ishai et al., this was only required to hold for random inputs, giving a definition analogous to a weak pseudorandom function; in other work [2], the attacker was allowed to choose arbitrary inputs but only in a non-adaptive manner. Here we consider the strongest notion where the attacker is free to adaptively choose its inputs to its oracle.

Definition 1. *Let $H : \{0,1\}^k \to \{0,1\}^k$ be a function, and let $\mathcal{R}$ be a distribution on $\{0,1\}^k$. For $R \in \{0,1\}^k$, define $\mathcal{O}^{\mathsf{cr}}_R(x) \stackrel{\text{def}}{=} H(x \oplus R)$. For a distinguisher D, define*

$$\mathbf{Adv}^{\mathsf{cr}}_{H,\mathcal{R}}(D) \stackrel{\text{def}}{=} \left| \Pr_{R \leftarrow \mathcal{R}}\left[D^{\mathcal{O}^{\mathsf{cr}}_R(\cdot)} = 1\right] - \Pr_{f \leftarrow \mathsf{F}_k}\left[D^{f(\cdot)} = 1\right] \right|.$$

*H is (t,q,ρ,ε)-*correlation robust *if for all D running in time at most t and making at most q queries to $\mathcal{O}^{\mathsf{cr}}_R(\cdot)$, and all $\mathcal{R}$ with min-entropy at least ρ, it holds that $\mathbf{Adv}^{\mathsf{cr}}_{H,\mathcal{R}}(D) \le \varepsilon$.*

Circular correlation robustness (ccr). Choi et al. [14] extended the notion of correlation robustness to allow for a form of "circularity" needed to prove security of the free-XOR technique [31] for circuit garbling. Zahur et al. [53] used a

[5] It is unclear to us whether H satisfies their definition or not. Nevertheless, we believe that half-gates garbling using their instantiation of H can be proven secure directly in the random-permutation model. That is, we do not claim that their scheme is insecure, only that their analysis is buggy.

weaker (but more complex) version of this definition, also in the context of garbling; more details about their definition are given in Section V.

Definition 2. *Let $H : \{0,1\}^k \to \{0,1\}^k$ be a function, and let $\mathcal{R}$ be a distribution on $\{0,1\}^k$. For $R \in \{0,1\}^k$, define $\mathcal{O}_R^{\mathsf{ccr}}(x,b) \stackrel{\text{def}}{=} H(x \oplus R) \oplus b \cdot R$. For a distinguisher D, define*

$$\mathbf{Adv}_{H,\mathcal{R}}^{\mathsf{ccr}}(D) \stackrel{\text{def}}{=} \left| \Pr_{R \leftarrow \mathcal{R}}\left[D^{\mathcal{O}_R^{\mathsf{ccr}}(\cdot)} = 1\right] - \Pr_{f \leftarrow \mathsf{F}_{k+1,k}}\left[D^{f(\cdot)} = 1\right] \right|,$$

*where we require that D never queries both $(x,0)$ and $(x,1)$ for any x. We say H is (t,q,ρ,ε)-*circular correlation robust *if for all D running in time at most t and making at most q queries to $\mathcal{O}_R^{\mathsf{ccr}}(\cdot)$, and all $\mathcal{R}$ with min-entropy at least ρ, it holds that $\mathbf{Adv}_{H,\mathcal{R}}^{\mathsf{ccr}}(D) \le \varepsilon$.*

Tweakable correlation robustness (tcr) and tweakable circular correlation robustness (tccr). By analogy with the notion of tweakable block ciphers [34], we extend the notion of (circular) correlation robustness to also incorporate a tweak. As we discuss in Section IV, the addition of a tweak is crucial for security of some protocols in the malicious setting.

Our definitions allow the attacker to repeat tweaks arbitrarily many times. For some applications, weaker notions (such as requiring non-repeating tweaks) may suffice, and the definitions may be modified appropriately for those cases.

Definition 3. *Let $H : \{0,1\}^{2k} \to \{0,1\}^k$ be a function, and let $\mathcal{R}$ be a distribution on $\{0,1\}^k$. For $R \in \{0,1\}^k$, define $\mathcal{O}_R^{\mathsf{tcr}}(x,i) \stackrel{\text{def}}{=} H(x \oplus R, i)$ and $\mathcal{O}_R^{\mathsf{tccr}}(x,i,b) \stackrel{\text{def}}{=} H(x \oplus R, i) \oplus b \cdot R$. For a distinguisher D, define*

$$\mathbf{Adv}_{H,\mathcal{R}}^{\mathsf{tcr}}(D) \stackrel{\text{def}}{=} \left| \Pr_{R \leftarrow \mathcal{R}}\left[D^{\mathcal{O}_R^{\mathsf{tcr}}(\cdot)} = 1\right] - \Pr_{f \leftarrow \mathsf{F}_{2k,k}}\left[D^{f(\cdot)} = 1\right] \right|$$

and

$$\mathbf{Adv}_{H,\mathcal{R}}^{\mathsf{tccr}}(D) \stackrel{\text{def}}{=} \left| \Pr_{R \leftarrow \mathcal{R}}\left[D^{\mathcal{O}_R^{\mathsf{tccr}}(\cdot)} = 1\right] - \Pr_{f \leftarrow \mathsf{F}_{2k+1,k}}\left[D^{f(\cdot)} = 1\right] \right|,$$

*where in the latter case we require that D never queries both $(x,i,0)$ and $(x,i,1)$ for any x,i. We say H is (t,q,ρ,ε)-*tweakable correlation robust *(resp., (t,q,ρ,ε)-*tweakable circular correlation robust*) if for all D running in time at most t and making at most q queries to $\mathcal{O}_R^{\mathsf{tcr}}(\cdot)$ (resp., $\mathcal{O}_R^{\mathsf{tccr}}(\cdot)$), and all $\mathcal{R}$ with min-entropy at least ρ, it holds that $\mathbf{Adv}_{H,\mathcal{R}}^{\mathsf{tcr}}(D) \le \varepsilon$ (resp., $\mathbf{Adv}_{H,\mathcal{R}}^{\mathsf{tccr}}(D) \le \varepsilon$).*

Definitions in the random-permutation model. In this work we construct hash functions H satisfying the above definitions in the random-permutation model. That is, we assume a public, random permutation $\pi : \{0,1\}^k \to \{0,1\}^k$ and show constructions of H given oracle access to π. The security definitions are then modified by (1) taking probabilities also over uniform choice of π and (2) giving the distinguisher D oracle access to both π and its inverse π^{-1}. In this case, we can prove security of our constructions unconditionally so long as we bound the number of queries that D makes to π/π^{-1} and its other oracle. Thus, e.g., we say that a construction H in the random-permutation model is (p,q,ρ,ε)-*correlation robust* if for all D making at most p queries to π/π^{-1} and q queries to $\mathcal{O}_R^{\mathsf{cr}}$, and all $\mathcal{R}$ with min-entropy at least ρ, it holds that $\mathbf{Adv}_{H,\mathcal{R}}^{\mathsf{cr}}(D) \le \varepsilon$. We remark that for applications to secure computation, q is typically fixed by the protocol but p can be as large as the adversary's running time.

Semi-honest security			
Reference	OT type	Prior work	This work
[28]	Standard OT	cr	cr
[1]	Correlated OT	random oracle	cr
[1]	Random OT	random oracle	cr
Malicious security			
[29, 2]	Standard OT	cr*	tcr
[2]	Correlated OT	random oracle	tcr
[2]	Random OT	random oracle	tcr

TABLE II: **Assumptions about H in protocols for OT extension.** cr* refers to correlation robustness with compression, for which no instantiation from a fixed-key cipher was known.

Relations between the definitions. It is easy to see that any H that is ccr (resp., tccr) is also cr (resp., tcr). It is also easy to see that any H that is tcr (resp., tccr) can be used to construct a hash function H' that is cr (resp., ccr).

The construction we give in Section VII-B is cr but not ccr. We are not aware of a generic transformation from cr (resp., ccr) to tcr (resp., tccr), however in Section V we show (implicitly) that any H that is ccr can be used to construct a hash function H' that satisfies tccr for random inputs and non-repeating tweaks. We show there that this weaker notion suffices for analyzing the half-gates garbling scheme.

IV. Oblivious-Transfer Extension

As discussed in Section II-A, many existing implementations of OT extension based on a fixed-key cipher are insecure or, at best, cannot be proven secure. Part of the problem is that some OT-extension protocols are proven secure in the random-oracle model, but (efficient) instantiations of a random oracle from a fixed-key cipher are not known. To address this gap, we present here various flavors of OT-extension protocols based on hash functions satisfying the definitions from the previous section. In doing so, we improve on the assumptions used in several prior works, as summarized in Table II. Specifically, for semi-honest security we show that it suffices for the hash function to be correlation robust. For malicious security, however, we need *tweakable* correlation robustness. The addition of a tweak is necessary in the malicious setting, intuitively, to prevent the attacks shown in Section II. Without a tweak, the hash function behaves "the same" across different executions of the OT, and this can be exploited by a malicious adversary. The attack can be prevented by incorporating an independent tweak for each execution. (We formally prove this intuition in the next section.)

Here we consider standard-OT extension and correlated-OT extension; we defer the case of random-OT extension

Functionality $\mathcal{F}_{\Delta\text{-ROT}}$

Initialize: Upon receiving (Init, Δ) from $\mathsf{P_A}$ with $\Delta \in \{0,1\}^k$, and (Init) from $\mathsf{P_B}$, store Δ and ignore subsequent Init commands.
Set up correlations: Upon receiving $(x_1, \ldots, x_m)$ from $\mathsf{P_B}$ with $x_i \in \{0,1\}$ do:

1) For each $i \in [m]$, sample uniform $\mathbf{a}_i, \mathbf{b}_i \in \{0,1\}^k$ such that $\mathbf{a}_i \oplus \mathbf{b}_i = x_i \cdot \Delta$.
2) If $\mathsf{P_A}$ is corrupted, wait for $\mathcal{A}$ to send $\{\mathbf{a}_i\}$ and recompute $\{\mathbf{b}_i\}$ accordingly.
   If $\mathsf{P_B}$ is corrupted, wait for $\mathcal{A}$ to send $\{\mathbf{b}_i\}$ and recompute $\{\mathbf{a}_i\}$ accordingly.
3) Send $\{\mathbf{a}_i\}$ to $\mathsf{P_A}$ and $\{\mathbf{b}_i\}$ to $\mathsf{P_B}$.

Global key query: If $\mathsf{P_B}$ is corrupted then $\mathcal{A}$ can send a predicate $P : \{0,1\}^k \to \{0,1\}$ to the functionality after initialization but before sending its input. If $P(\Delta) = 1$, the functionality sends 1 to $\mathcal{A}$; otherwise, the functionality aborts and notifies $\mathsf{P_A}$.

Fig. 1. Functionality $\mathcal{F}_{\Delta\text{-ROT}}$.

Functionality $\mathcal{F}_{\mathsf{S\text{-}OT}}$

Upon receiving $((\mathbf{m}_1^0, \mathbf{m}_1^1), \ldots, (\mathbf{m}_m^0, \mathbf{m}_m^1))$ from $\mathsf{P_A}$ and $(x_1, \ldots, x_m)$ from $\mathsf{P_B}$ with $x_i \in \{0,1\}$, send $\{\mathbf{m}_i^{x_i}\}$ to $\mathsf{P_B}$.

Fig. 2. Functionality $\mathcal{F}_{\mathsf{S\text{-}OT}}$ for standard OT.

Protocol $\Pi_{\mathsf{S\text{-}OT}}$

Inputs: $\mathsf{P_A}$ has $(\mathbf{m}_1^0, \mathbf{m}_1^1), \ldots, (\mathbf{m}_m^0, \mathbf{m}_m^1)$ and $\mathsf{P_B}$ has $x_1, \ldots, x_m$ with $x_i \in \{0,1\}$.

Protocol:

1) $\mathsf{P_A}$ chooses uniform Δ and sends (Init, Δ) to $\mathcal{F}_{\Delta\text{-ROT}}$; $\mathsf{P_B}$ sends (Init) to $\mathcal{F}_{\Delta\text{-ROT}}$.
2) $\mathsf{P_B}$ sends $(x_1, \ldots, x_m)$ to $\mathcal{F}_{\Delta\text{-ROT}}$, which returns $\mathbf{a}_1, \ldots, \mathbf{a}_m$ to $\mathsf{P_A}$ and $\mathbf{b}_1, \ldots, \mathbf{b}_m$ to $\mathsf{P_B}$.
3) **Semi-honest security:** $\mathsf{P_A}$ sends $\mathbf{c}_i^0 := H(\mathbf{a}_i) \oplus \mathbf{m}_i^0$ and $\mathbf{c}_i^1 := H(\mathbf{a}_i \oplus \Delta) \oplus \mathbf{m}_i^1$ to $\mathsf{P_B}$, who can then compute $\mathbf{m}_i^{x_i} := \mathbf{c}_i^{x_i} \oplus H(\mathbf{b}_i)$.

   Malicious security: $\mathsf{P_A}$ sends $\mathbf{c}_i^0 := H(\mathbf{a}_i, i) \oplus \mathbf{m}_i^0$ and $\mathbf{c}_i^1 := H(\mathbf{a}_i \oplus \Delta, i) \oplus \mathbf{m}_i^1$ to $\mathsf{P_B}$, who can then compute $\mathbf{m}_i^{x_i} := \mathbf{c}_i^{x_i} \oplus H(\mathbf{b}_i, i)$.

Fig. 3. Protocol $\Pi_{\mathsf{S\text{-}OT}}$ in the $\mathcal{F}_{\Delta\text{-ROT}}$-hybrid model.

to Appendix A. Since our focus is on instantiating the hash function H used in the second phase of OT extension (cf. the beginning of Section II-A), we present all our protocols in the $\mathcal{F}_{\Delta\text{-ROT}}$-hybrid model (see Figure 1). This ideal functionality provides an abstraction of the first phase of OT extension, and efficient protocols realizing it are known in both the semi-honest [1] and malicious [29] settings.

A. Standard-OT Extension

Figure 2 describes the standard OT functionality $\mathcal{F}_{\mathsf{S\text{-}OT}}$. In Figure 3 we show a protocol realizing standard OT in the $\mathcal{F}_{\Delta\text{-ROT}}$-hybrid model, in both the semi-honest and malicious settings. We remark that the result for the case of semi-honest security already follows from the work of Ishai et al. [28].

Theorem 1 (Informal)**.** *If H is* cr *(resp.,* tcr*) then protocol $\Pi_{\mathsf{S\text{-}OT}}$ securely realizes $\mathcal{F}_{\mathsf{S\text{-}OT}}$ for semi-honest (resp., malicious) adversaries in the $\mathcal{F}_{\Delta\text{-ROT}}$-hybrid model.*

The theorem is somewhat informal since, e.g., we have not defined what it means for H to be cr but only what it means for it to be $(t, q, \rho, \varepsilon)$-cr. A formal statement incorporating concrete security bounds can be obtained from the proof.

Proof. Security in the $\mathcal{F}_{\Delta\text{-ROT}}$-hybrid model for a corrupted $\mathsf{P_A}$, whether semi-honest or malicious, holds perfectly and is trivial to show. We therefore focus on the case of an adversary $\mathcal{A}$ corrupting $\mathsf{P_B}$.

The case of a semi-honest $\mathsf{P_B}$ is straightforward. (As noted earlier, this is also implicit in [28].) The simulator in this case extracts $\mathsf{P_B}$'s inputs from its input to the $\mathcal{F}_{\Delta\text{-ROT}}$ functionality, sends these to the $\mathcal{F}_{\mathsf{S\text{-}OT}}$ functionality to obtain $\{\mathbf{m}_i^{x_i}\}$, and then for all i sets $\mathbf{c}_i^{x_i} := H(\mathbf{b}_i) \oplus \mathbf{m}_i^{x_i}$ and chooses uniform $\mathbf{c}_i^{1-x_i}$. It is immediate that correlation robustness of H implies that this results in a view for $\mathsf{P_B}$ that is indistinguishable from its view in a real execution.

In the malicious case the simulator is almost the same as before, but the proof is more involved. For completeness, we describe the simulator $\mathcal{S}$ in full:

1–2. $\mathcal{S}$ obtains the inputs $(x_1, \ldots, x_m)$, along with the values $\{\mathbf{b}_i\}_{i \in [m]}$, that $\mathcal{A}$ sends to $\mathcal{F}_{\Delta\text{-ROT}}$.
   $\mathcal{S}$ sends $(x_1, \ldots, x_m)$ to $\mathcal{F}_{\mathsf{S\text{-}OT}}$, which returns $(\mathbf{m}_0^{x_0}, \ldots, \mathbf{m}_m^{x_m})$.
   $\mathcal{S}$ chooses a uniform Δ and answers $\mathcal{A}$'s global key query (if any) using Δ.
3. For all i, $\mathcal{S}$ sets $\mathbf{c}_i^{x_i} := H(\mathbf{b}_i, i) \oplus \mathbf{m}_i^{x_i}$ and chooses uniform $\mathbf{c}_i^{1-x_i}$. It sends the $\{\mathbf{c}_i^b\}$ to $\mathsf{P_B}$.

If $\mathcal{A}$ makes no global key query then it is immediate that if H is (t, m, k, ε)-tweakable correlation robust then for any $\mathcal{A}$ running in time at most t the advantage of $\mathcal{A}$ in distinguishing the simulated view from the real view is at most $\varepsilon + 2^{-k}$ (with the second term accounting for the probability that $\Delta = 0^k$).

Assume that for all $0 \le \rho \le k$ it holds that H is $(t, m, \rho, \varepsilon(\rho))$-tweakable correlation robust. Say the global key query P of $\mathcal{A}$ is such that $|\{\Delta : P(\Delta) = 1\}| = 2^\rho$. Then with probability $2^{\rho-k}$ the attacker reduces the min-entropy of Δ to ρ, but with the remaining probability the functionality aborts. The maximum distinguishing advantage of $\mathcal{A}$ is thus $\max_\rho\{2^{\rho-k} \cdot \varepsilon(\rho)\} + 2^{-k}$. □

B. Correlated OT

Correlated OT, proposed by Asharov et al. [1], is a weaker form of OT in which the sender can only specify the XOR of its "messages" (which are otherwise chosen uniformly by the functionality); the relevant ideal functionality $\mathcal{F}_{\mathsf{C\text{-}OT}}$ is

Functionality $\mathcal{F}_{\mathsf{C\text{-}OT}}$

Upon receiving $(\Delta_1, \ldots, \Delta_m)$ from $\mathsf{P_A}$ and $(x_1, \ldots, x_m)$ from $\mathsf{P_B}$ with $x_i \in \{0,1\}$ do:

1) For each $i \in [m]$, sample uniform $\mathbf{m}_i^0 \in \{0,1\}^k$ and set $\mathbf{m}_i^1 := \mathbf{m}_i^0 \oplus \Delta_i$.
2) If $\mathsf{P_A}$ is corrupted, wait for $\mathcal{A}$ to send $\{\mathbf{m}_i^0\}$ and recompute $\{\mathbf{m}_i^1\}$ accordingly.
   If $\mathsf{P_B}$ is corrupted, wait for $\mathcal{A}$ to send $\{\mathbf{m}_i^{x_i}\}$ and recompute $\{\mathbf{m}_i^{1-x_i}\}$ accordingly.
3) Send $\{\mathbf{m}_i^0\}$ to $\mathsf{P_A}$ and $\{\mathbf{m}_i^{x_i}\}$ to $\mathsf{P_B}$.

Fig. 4. Functionality $\mathcal{F}_{\mathsf{C\text{-}OT}}$ for correlated OT.

Protocol $\Pi_{\mathsf{C\text{-}OT}}$

Inputs: $\mathsf{P_A}$ has $\Delta_1, \ldots, \Delta_m \in \{0,1\}^k$ and $\mathsf{P_B}$ has $x_1, \ldots, x_m$ with $x_i \in \{0,1\}$.

Protocol:

1) $\mathsf{P_A}$ chooses uniform Δ and sends (Init, Δ) to $\mathcal{F}_{\Delta\text{-}\mathsf{ROT}}$; $\mathsf{P_B}$ sends (Init) to $\mathcal{F}_{\Delta\text{-}\mathsf{ROT}}$.
2) $\mathsf{P_B}$ sends $(x_1, \ldots, x_m)$ to $\mathcal{F}_{\Delta\text{-}\mathsf{ROT}}$, which returns $\mathbf{a}_1, \ldots, \mathbf{a}_m$ to $\mathsf{P_A}$ and $\mathbf{b}_1, \ldots, \mathbf{b}_m$ to $\mathsf{P_B}$.
3) **Semi-honest security:** $\mathsf{P_A}$ computes $\mathbf{m}_i^0 := H(\mathbf{a}_i)$ and sends $\mathbf{c}_i := H(\mathbf{a}_i) \oplus H(\mathbf{a}_i \oplus \Delta) \oplus \Delta_i$ to $\mathsf{P_B}$, who can then compute $\mathbf{m}_i^{x_i} := H(\mathbf{b}_i) \oplus x_i \mathbf{c}_i$.

   Malicious security: $\mathsf{P_A}$ computes $\mathbf{m}_i^0 := H(\mathbf{a}_i, i)$ and sends $\mathbf{c}_i := H(\mathbf{a}_i, i) \oplus H(\mathbf{a}_i \oplus \Delta, i) \oplus \Delta_i$ to $\mathsf{P_B}$, who can then compute $\mathbf{m}_i^{x_i} := H(\mathbf{b}_i, i) \oplus x_i \mathbf{c}_i$.

Fig. 5. Protocol $\Pi_{\mathsf{C\text{-}OT}}$ in the $\mathcal{F}_{\Delta\text{-}\mathsf{ROT}}$-hybrid model.

given in Figure 4. Prior work showing correlated-OT extension protocols [1, 2] requires a programmable random oracle, even for semi-honest security, because the simulator needs to program the output of H to ensure consistency with the output from the ideal functionality. In fact, it appears difficult to efficiently realize the ideal functionality as defined by Asharov et al. without a programmable random oracle, and for this reason we weaken the ideal functionality to allow the adversary to choose its output. (Interestingly, we observe that prior work [2] does not actually realize the ideal functionality of Asharov et al. [1] but instead also realizes the weaker version we define here. See Appendix B for more details.) This ideal functionality still suffices for applications to secure computation. In Figure 5 we show a protocol that realizes this functionality in both the semi-honest and malicious settings.

Theorem 2 (Informal)**.** *If H is* cr *(resp.,* tcr*) then protocol $\Pi_{\mathsf{C\text{-}OT}}$ securely realizes $\mathcal{F}_{\mathsf{C\text{-}OT}}$ for semi-honest (resp., malicious) adversaries in the $\mathcal{F}_{\Delta\text{-}\mathsf{ROT}}$-hybrid model.*

The comment following Theorem 1 applies here as well.

Proof. As in the case of Theorem 1, security for an adversary $\mathcal{A}$ corrupting $\mathsf{P_A}$ is perfect and easy to show. We thus focus on the case of a corrupted $\mathsf{P_B}$. We consider the malicious setting; the semi-honest setting follows similarly.

The simulator $\mathcal{S}$ for a malicious $\mathsf{P_B}$ is as follows:

1–2. $\mathcal{S}$ obtains the inputs $(x_1, \ldots, x_m)$, as well as the values $\{\mathbf{b}_i\}_{i \in [m]}$, that $\mathcal{A}$ sends to $\mathcal{F}_{\Delta\text{-}\mathsf{ROT}}$.
   $\mathcal{S}$ also chooses a uniform Δ and answers $\mathcal{A}$'s global key query (if any) using Δ.
3. $\mathcal{S}$ chooses uniform $\{\mathbf{c}_i\}$ and sends them to $\mathsf{P_B}$. It sets $\mathbf{m}_i^{x_i} := H(\mathbf{b}_i, i) \oplus x_i \mathbf{c}_i$ and sends $(x_1, \ldots, x_m)$ and $\{\mathbf{m}_i^{x_i}\}$ to $\mathcal{F}_{\mathsf{C\text{-}OT}}$.

A proof of indistinguishability follows as in the proof of Theorem 1. □

V. Revisiting the Half-Gates Garbling Scheme

Zahur et al. [53] introduced the half-gates garbling scheme based on an abstract hash function H. To analyze the scheme, Zahur et al. introduce a definition called "circular correlation robustness for naturally derived keys," and prove security for their garbling scheme when H satisfies that definition. They then claim, without proof, that the hash function $H(x, i) = \pi(2x \oplus i) \oplus 2x \oplus i$ satisfies their definition. It is not clear to us that this is the case (but see footnote 5).

In this section, we revisit the assumption needed to prove security of the half-gates garbling scheme. (Everything we say here applies to the privacy-free version of the scheme as well.) We weaken the definition of circular correlation robustness to match exactly what is needed for the security proof of Zahur et al., and then show how to achieve the definition based on the notions introduced in Section III.

The notion of "circular correlation robustness for naturally derived keys" can be viewed as a form of tweakable circular correlation robustness where the attacker does not have full control over the queries made to the oracle $\mathcal{O}_R^{\mathsf{tccr}}$ (cf. Def. 3). We proceed to give the details. Let $H : \{0,1\}^{2k} \to \{0,1\}^k$ be a function, and let $\mathcal{R}$ be a distribution on $\{0,1\}^k$. We say a sequence of operations $\mathcal{Q} = (Q_1, \ldots, Q_q)$ is natural if each Q_i is one of the following:

1) $x_i \leftarrow \{0,1\}^k$.
2) $x_i := x_{i_1} \oplus x_{i_2}$, where $i_1 < i_2 < i$.
3) $x_i := H(x_{i_1}, i)$, where $i_1 < i$.
4) $x_i := \mathcal{O}(x_{i_1}, i, b)$, where $i_1 < i$.

Fix some natural sequence $\mathcal{Q}$ of length q. In the real-world experiment, denoted $\mathsf{Real}_{H,\mathcal{Q},\mathcal{R}}$, a key R is sampled from $\mathcal{R}$ and then the oracle $\mathcal{O}$ in step 4, above, is set to $\mathcal{O}_R^{\mathsf{tccr}}$. In the ideal-world experiment, denoted $\mathsf{Ideal}_{\mathcal{Q}}$, the oracle $\mathcal{O}$ is instead a function chosen uniformly from $\mathsf{F}_{2k+1,k}$. Either experiment defines a distribution (determined by executing the operations in $\mathcal{Q}$ in order) over values $x_1, \ldots, x_q$, which are output by the experiment.

Definition 4. *For $H, \mathcal{Q}, \mathcal{R}$ as above and a distinguisher D, define* $\mathbf{Adv}^{\mathsf{ccrnd}}_{H,\mathcal{Q},\mathcal{R}}(D)$ *as*

$$\left| \Pr_{\{x_i\} \leftarrow \mathsf{Real}_{H,\mathcal{Q},\mathcal{R}}} \left[D(\{x_i\}) = 1 \right] - \Pr_{\{x_i\} \leftarrow \mathsf{Ideal}_{\mathcal{Q}}} \left[D(\{x_i\}) = 1 \right] \right|.$$

*We say H is $(t, q, \rho, \varepsilon)$-*circular correlation robust for naturally derived keys *if for all D running in time at most t, all $\mathcal{Q}$ of length q, and all $\mathcal{R}$ with min-entropy at least ρ, it holds that $\mathbf{Adv}^{\mathsf{ccrnd}}_{H,\mathcal{Q},\mathcal{R}}(D) \le \varepsilon$.*

It is immediate that a tccr hash function satisfies the above definition. But this is overkill, and we show now that a family of hash functions satisfying the notion can be constructed from any H that is ccr. Specifically, we show that the keyed function $H'_S(x,i) = H(S \oplus x \oplus i)$ satisfies the above definition when S is uniform. (We stress, however, that S is public and so is also given to D.) Note that in the context of the half-gates scheme, the circuit garbler would choose S and send it (along with the garbled circuit) to the evaluator.

Theorem 3. *Let H be (t,q,ρ,ε)-ccr, and define $H'_S(x,i) = H(S \oplus x \oplus i)$. Then H'_S is (t,q,ρ,ε')-circular correlation robust for naturally derived keys (where the probabilities are also taken over uniform choice of $S \in \{0,1\}^\kappa$) with $\varepsilon' = 2\varepsilon + q^2/2^{k+1}$.*

Proof. Define $H_S(x) = H(S \oplus x)$, so that $H'_S(x,i) = H_S(x \oplus i)$. Fix some sequence $\mathcal{Q} = \{Q_1, \ldots, Q_q\}$. Consider the random variables $x_1, \ldots, x_q$ that are defined during the course of experiment $\mathsf{Real}_{H',\mathcal{Q},\mathcal{R}}$, and let Coll denote the event that there exist distinct i_1, i_2 with $x_{i_1} \oplus i_1 = x_{i_2} \oplus i_2$.

To bound the probability of Coll, note that all queries to H_S throughout the course of the experiment (which can occur either as "type 3" or "type 4" operations) are determined independently of S. Consider a modified experiment $\mathsf{Real}^*_{\mathcal{Q},\mathcal{R}}$ in which H_S is replaced with a function f chosen uniformly from F_k. Viewing S as the key, and using the fact that H is correlation robust, we must have

$$\left| \Pr_{\mathsf{Real}_{H',\mathcal{Q},\mathcal{R}}}[\mathsf{Coll}] - \Pr_{\mathsf{Real}^*_{\mathcal{Q},\mathcal{R}}}[\mathsf{Coll}] \right| \le \varepsilon.$$

Each operation Q_i in $\mathsf{Real}^*_{\mathcal{Q},\mathcal{R}}$ is one of the following:

1) $x_i \leftarrow \{0,1\}^k$.
2) $x_i := x_{i_1} \oplus x_{i_2}$, where $i_1 < i_2 < i$.
3) $x_i := f(x_{i_1} \oplus i)$, where $i_1 < i$.
4) $x_i := f(R \oplus x_{i_1} \oplus i) \oplus bR$, where $i_1 < i$.

Fix some distinct i_1, i_2. We have

$$x_{i_1} \oplus x_{i_2} = \bigoplus_{i \in \mathcal{I}} x_i \oplus \bigoplus_{j \in \mathcal{J}} f(x_j \oplus i_j) \oplus \bigoplus_{k \in \mathcal{K}} f(R \oplus x_k \oplus i_k) \oplus bR,$$

for some sets $\mathcal{I}, \mathcal{J}, \mathcal{K} \subset [q]$ and $b \in \{0,1\}$. Note that Coll occurs iff $x_{i_1} \oplus x_{i_2} = i_1 \oplus i_2$. If the above expression is syntactically 0 (i.e., $\mathcal{I} = \mathcal{J} = \mathcal{K} = \emptyset$ and $b = 0$), then Coll cannot occur. If that is not the case, then at least one of $\mathcal{I}, \mathcal{J}, \mathcal{K}$ must be nonempty (note that $b = 1$ implies that $\mathcal{K} \neq \emptyset$). But then the probability that $x_{i_1} \oplus x_{i_2} = i_1 \oplus i_2$ is at most 2^{-k}. So, by a union bound, we find that the probability of Coll in $\mathsf{Real}^*_{\mathcal{Q},\mathcal{R}}$ is at most $q^2/2^{k+1}$. Hence the probability of Coll in $\mathsf{Real}_{H',\mathcal{Q},\mathcal{R}}$ is at most $q^2/2^{k+1} + \varepsilon$.

Conditioned on that event that Coll does not occur in $\mathsf{Real}_{H',\mathcal{Q},\mathcal{R}}$, no two queries to $H(R \oplus \cdot)$ as part of evaluating a "type 4" operation $\mathcal{O}(x_i, i, b) = H(R \oplus (S \oplus x_i \oplus i)) \oplus bR$ ever repeat, and thus we can construct from D a legal distinguisher against H in the sense of circular correlation robustness. Viewing R as the key, this implies that the distinguishing advantage of D is at most

$$\Pr_{\mathsf{Real}_{H',\mathcal{Q},\mathcal{R}}}[\mathsf{Coll}] + \varepsilon.$$

This completes the proof. □

VI. Other Applications of Correlation Robustness

Here we describe two other applications of correlation robust hash functions to secure distributed computing. Our discussion here is brief only because the improvements, once described, are immediate. We defer discussion about concrete performance improvements to Section VIII, where we show a $3\times$ improvement for both applications.

A. Length Extension for OT

A well-known technique for performing OT of long messages is to first carry out OT for (short) keys, and then to encrypt each message with the corresponding key. Thus, at a high level, we need to encrypt each of ℓ messages $m_1, \ldots, m_\ell$ with the corresponding key from among $k_1, \ldots, k_\ell$. While this can of course be done using a block cipher, the natural approach to doing so would involve keying the cipher with each of the ℓ keys, thus imposing the cost of ℓ key-scheduling operations. We observe that it is possible to do better using a correlation robust hash function H.

Let $m_i = m_i^1, \ldots, m_i^t$, where each block m_i^j is k bits long. Then the encryption $c_i^1, \ldots, c_i^t$ of each message m_i can be done by setting

$$c_i^j = H(j \oplus k_i) \oplus m_i^j.$$

(We do not need randomized encryption here since each key is used to encrypt just one message.) Security follows directly from correlation robustness of H.

B. The GGM Tree and Distributed Point Functions

The GGM tree construction [23] involves a binary tree in which the label of a node is used to derive the label of its children using a length-double pseudorandom generator (PRG) G. If G is instantiated using AES in counter mode, then deriving the labels for a leaf of the tree will require multiple key-scheduling operations. We observe that G can instead be instantiated using a correlation robust hash via

$$G(k) = H(1 \oplus k) \,\|\, H(2 \oplus k).$$

The GGM tree has recently been used in the construction of distributed point functions [11], which in turn have found several applications including secure computation [19], private queries on public data [48], and anonymous messaging [15].

VII. Instantiating Hash Functions in the RPM

In this section, we show constructions of hash functions based on a random permutation π that satisfy the definitions from Section III. Our proofs all rely on the H-coefficient technique, which we review in Section VII-A.

A. The H-Coefficient Technique

We briefly recall the H-coefficient technique [42, 13], specialized for our proofs in the following three sections. In all cases we consider a deterministic distinguisher D given access to two oracles. The first oracle is always a random permutation π on $\{0,1\}^k$ (and its inverse). The second oracle $\mathcal{O}$ can take two forms: in the real world it is a function that depends on a key R sampled from a distribution $\mathcal{R}$, while in the ideal world it is a random function with range $\{0,1\}^k$. We are interested in bounding the maximum difference between the probabilities that D outputs 1 in the real world vs. the ideal world, where the maximum is taken over all D making p queries to its first oracle and q queries to its second oracle.

We define a transcript of D's interaction by

$$\mathcal{Q} = (\mathcal{Q}_\pi, \mathcal{Q}_\mathcal{O}, R),$$

where $\mathcal{Q}_\pi = \{(x_1, y_1), \ldots\}$ records D's queries/answers to/from π or π^{-1} (with $(x,y) \in \mathcal{Q}_\pi$ meaning $\pi(x) = y$, regardless of whether the query was to π or π^{-1}) and $\mathcal{Q}_\mathcal{O} = \{(w_1, z_1), \ldots\}$ records D's queries/answers to/from the second oracle. A key R is appended to the transcript as well (even though it is not part of the view of D) to facilitate the analysis: in the real world, this is the key used by the second oracle, whereas in the ideal world it is simply an independent key sampled from $\mathcal{R}$. A transcript $\mathcal{Q}$ is *attainable* for some fixed D if there exist some oracles such that the interaction of D with those oracles would lead to transcript $\mathcal{Q}$.

Fix some D. Let $\mathcal{T}$ denote the set of attainable transcripts, and let $\Pr_{\mathsf{real}}[\cdot]$ and $\Pr_{\mathsf{ideal}}[\cdot]$ denote the probabilities of events in the real and ideal worlds, respectively. The H-coefficient technique involves defining a partition of $\mathcal{T}$ into a "bad" set $\mathcal{T}_{\mathsf{bad}}$ and a "good" set $\mathcal{T}_{\mathsf{good}} = \mathcal{T} \setminus \mathcal{T}_{\mathsf{bad}}$, and then showing that

$$\Pr_{\mathsf{ideal}}[\mathcal{Q} \in \mathcal{T}_{\mathsf{bad}}] \le \varepsilon_1$$

and

$$\forall \mathcal{Q} \in \mathcal{T}_{\mathsf{good}} : \frac{\Pr_{\mathsf{real}}[\mathcal{Q}]}{\Pr_{\mathsf{ideal}}[\mathcal{Q}]} \ge 1 - \varepsilon_2.$$

It is then possible to show that the distinguishing advantage of D is at most $\varepsilon_1 + \varepsilon_2$.

One key insight of the H-coefficient technique is that the ratio $\Pr_{\mathsf{real}}[\mathcal{Q}] / \Pr_{\mathsf{ideal}}[\mathcal{Q}]$ is equal to the ratio between the probability that the real-world oracles are consistent with $\mathcal{Q}$ and the probability that the ideal-world oracles are consistent with $\mathcal{Q}$. Now, for any attainable transcript $\mathcal{Q} = (\mathcal{Q}_\pi, \mathcal{Q}_\mathcal{O}, R)$, the probability that the ideal world is consistent with $\mathcal{Q}$ is always exactly

$$\frac{\Pr_\mathcal{R}[R]}{(2^k)_p \cdot 2^{kq}}, \tag{1}$$

where for integers $1 \le b \le a$, we set

$$(a)_b = a \cdot (a-1) \cdots (a-b+1)$$

with $(a)_0 = 1$ by convention. This is so since the probability that a random permutation on $\{0,1\}^k$ is consistent with the p queries in $\mathcal{Q}_\pi$ is exactly $1/(2^k)_p$; the probability that a random function with range $\{0,1\}^k$ is consistent with the q queries to $\mathcal{Q}_\mathcal{O}$ is exactly $1/2^{kq}$; and the probability that the key is R is exactly $\Pr_\mathcal{R}[R]$. Bounding the distinguishing advantage of D thus reduces to bounding the probability that the real world is consistent with transcripts $\mathcal{Q} \in \mathcal{T}_{\mathsf{good}}$.

Let $\pi \vdash \mathcal{Q}_\pi$ denote the event that permutation π is consistent with the queries/answers in $\mathcal{Q}_\pi$, i.e., that $\pi(x) = y$ for all $(x,y) \in \mathcal{Q}_\pi$. Since, in the real world, the behavior of the second oracle is completely determined by π and R, we can also write $(\pi, R) \vdash \mathcal{Q}_\mathcal{O}$ to denote the event that permutation π and key R are consistent with the queries/answers in $\mathcal{Q}_\mathcal{O}$. For a (good) transcript $\mathcal{Q} = (\mathcal{Q}_\pi, \mathcal{Q}_\mathcal{O}, R)$, the probability that the real world is consistent with $\mathcal{Q}$ is exactly

$$\Pr[(\pi, R) \vdash \mathcal{Q}_\mathcal{O} \mid \pi \vdash \mathcal{Q}_\pi] \cdot \Pr[\pi \vdash \mathcal{Q}_\pi] \cdot \Pr_\mathcal{R}[R]$$

(using independence of R and π). We have $\Pr[\pi \vdash \mathcal{Q}_\pi] = 1/(2^k)_p$ exactly as before. The crux of the proof thus reduces to bounding $\Pr[(\pi, R) \vdash \mathcal{Q}_\mathcal{O} \mid \pi \vdash \mathcal{Q}_\pi]$. We can equivalently write this as $\Pr[\forall (w,z) \in \mathcal{Q}_\mathcal{O} : \mathcal{O}_R^\pi(w) = z \mid \pi \vdash \mathcal{Q}_\pi]$. Note that since the proof mainly uses the randomness in the RPM, we only need a non-programmable random permutation.

B. Correlation Robustness

We begin by showing a construction that achieves correlation robustness. We refer to the resulting hash function as MMO since it is reminiscent of (though not identical to) the Matyas-Meyer-Oseas construction. Namely, we define

$$\mathsf{MMO}^\pi(x) \stackrel{\text{def}}{=} \pi(x) \oplus x.$$

Theorem 4. *If π is modeled as a random permutation then MMO^π is $(p, q, \rho, \varepsilon)$-correlation robust, where*

$$\varepsilon = \frac{2pq}{2^\rho} + \frac{q^2}{2^{k+1}}.$$

Proof. We consider a deterministic distinguisher D making queries to two oracles. The first is a random permutation π on $\{0,1\}^k$ (and its inverse); in the real world, the second oracle is $\mathcal{O}_R^{\mathsf{cr}}(\cdot) = \mathsf{MMO}^\pi(R \oplus \cdot)$ (for R sampled from a distribution $\mathcal{R}$), and in the ideal world it is an independent random function from $\{0,1\}^k$ to $\{0,1\}^k$. As in Section VII-A, we denote the transcript of D's interaction by $\mathcal{Q} = (\mathcal{Q}_\pi, \mathcal{Q}_\mathcal{O}, R)$. We only consider attainable transcripts from now on.

We say a transcript $\mathcal{Q} = (\mathcal{Q}_\pi, \mathcal{Q}_\mathcal{O}, R)$ is *bad* if either:

- (B-1) There is a query $(w,z) \in \mathcal{Q}_\mathcal{O}$ and a query of the form $(R \oplus w, \star)$ or of the form $(\star, R \oplus w \oplus z)$ in $\mathcal{Q}_\pi$.
- (B-2) There exist distinct queries $(w_i, z_i), (w_j, z_j) \in \mathcal{Q}_\mathcal{O}$ such that $w_i \oplus z_i = w_j \oplus z_j$.

In the ideal world, for some fixed queries $(w,z) \in \mathcal{Q}_\mathcal{O}$ and $(x,y) \in \mathcal{Q}_\pi$, we have

$$\Pr[R \oplus w = x] = \Pr[R = w \oplus x] \le \frac{1}{2^\rho}$$

as R has min-entropy ρ. Thus the probability of (B-1) is at most $2pq \cdot 2^{-\rho}$. Similarly, the probability of (B-2) is

$$\Pr[\exists i \ne j : w_i \oplus z_i = w_j \oplus z_j] = \binom{q}{2} \cdot \frac{1}{2^k} \le q^2/2^{k+1}$$

since both z_i and z_j are random.

Fix a good transcript $\mathcal{Q} = (\mathcal{Q}_\pi, \mathcal{Q}_\mathcal{O}, R)$. The probability that the ideal world is consistent with $\mathcal{Q}$ is exactly (1). The probability that the real world is consistent with $\mathcal{Q}$ is

$$\frac{\Pr[\forall (w,z) \in \mathcal{Q}_\mathcal{O} : \mathcal{O}_R^{\mathsf{cr}}(w) = z \mid \pi \vdash \mathcal{Q}_\pi]}{(2^k)_p} \cdot \Pr_\mathcal{R}[R].$$

We can express the numerator above as

$$\prod_{i=1}^{q} \Pr[\mathcal{O}_R^{\mathsf{cr}}(w_i) = z_i \mid \pi \vdash \mathcal{Q}_\pi \wedge \forall j < i : \mathcal{O}_R^{\mathsf{cr}}(w_j) = z_j].$$

Fix some i. Note that $\mathcal{O}_R^{\mathsf{cr}}(w_i) = z_i$ iff $\mathsf{MMO}^\pi(R \oplus w_i) = z_i$, i.e., $\pi(R \oplus w_i) = R \oplus w_i \oplus z_i$. Moreover, since the transcript is good there is no query of the form $(R \oplus w_i, \star)$ in $\mathcal{Q}_\pi$ (since (B-1) does not occur), nor is $\pi(R \oplus w_i)$ determined by the fact that $\mathcal{O}_R^{\mathsf{cr}}(w_j) = z_j$ for all $j < i$ (since all queries to $\mathcal{O}_R^{\mathsf{cr}}$ are distinct). Similarly, there is no query of the form $(\star, R \oplus w_i \oplus z_i)$ in $\mathcal{Q}_\pi$ (since (B-1) does not occur), nor is $\pi^{-1}(R \oplus w_i \oplus z_i)$ determined by the fact that $\mathcal{O}_R^{\mathsf{cr}}(w_j) = z_j$ for all $j < i$ (since (B-2) does not occur). Thus, we have

$$\begin{aligned}\Pr[\mathcal{O}_R^{\mathsf{cr}}(w_i) = z_i \mid \pi \vdash \mathcal{Q}_\pi \wedge \forall j < i : \mathcal{O}_R^{\mathsf{cr}}(w_j) = z_j] \\ = 1/(2^k - p - i + 1) \;\geq\; 1/2^k\end{aligned}$$

for all i. It follows that

$$\Pr[\forall (w,z) \in \mathcal{Q}_\mathcal{O} : \mathcal{O}_R^{\mathsf{cr}}(w) = z \mid \pi \vdash \mathcal{Q}_\pi] \geq 1/2^{kq},$$

and so the probability that the real world is consistent with the transcript is at least (1). This completes the proof. □

C. Circular Correlation Robustness

We begin by observing that the construction from the previous section is *not* circular correlation robust. (To the best of our knowledge, this gives the first explicit separation between correlation robustness and circular correlation robustness.) To see this, consider the following distinguisher D given oracle access to π and an oracle $\mathcal{O}$:

1) Query $z := \mathcal{O}(x, 1)$, where x is arbitrary.
2) Query $s := \pi^{-1}(x \oplus z)$, and set $R^* := x \oplus s$.
3) Query $z' := \mathcal{O}(x', 0)$, for any $x' \neq x$. Output 1 iff $z' = \mathsf{MMO}^\pi(x' \oplus R^*)$.

Note that if $\mathcal{O}(x, b) = \mathcal{O}_R^{\mathsf{ccr}}(x, b) \stackrel{\text{def}}{=} \mathsf{MMO}^\pi(x \oplus R) \oplus b \cdot R$ then

$$z = \pi(x \oplus R) \oplus (x \oplus R) \oplus R = \pi(x \oplus R) \oplus x.$$

Thus, $R^* = R$ and so D always outputs 1. On the other hand, if $\mathcal{O}$ is a random function then D outputs 1 only with probability 2^{-k}.

A small change to the previous construction, however, suffices to achieve circular correlation robustness. For a function $\sigma : \{0,1\}^k \to \{0,1\}^k$ that we will fix later, define

$$\widehat{\mathsf{MMO}}_\sigma^\pi(x) \stackrel{\text{def}}{=} \pi(\sigma(x)) \oplus \sigma(x).$$

We say σ is *linear* if $\sigma(x \oplus y) = \sigma(x) \oplus \sigma(y)$ for all $x, y \in \{0,1\}^k$. We say σ is an *orthomorphism* [12] if it is a permutation, and the function σ' given by $\sigma'(x) \stackrel{\text{def}}{=} \sigma(x) \oplus x$ is also a permutation.

Theorem 5. *Let σ be a linear orthomorphism. If π is modeled as a random permutation then $\widehat{\mathsf{MMO}}_\sigma^\pi$ is $(p, q, \rho, \varepsilon)$-circular correlation robust, where*

$$\varepsilon = \frac{2pq}{2^\rho} + \frac{q^2}{2^{k+1}}.$$

Proof. We prove a more general result. For some function $\sigma : \{0,1\}^k \to \{0,1\}^k$ and distribution $\mathcal{R}$ over $\{0,1\}^k$, set

$$\mathbf{H}_\infty(\sigma(\mathcal{R}) \oplus \mathcal{R}) \stackrel{\text{def}}{=} -\log\left(\max_{R^*} \Pr_{R \leftarrow \mathcal{R}}[\sigma(R) \oplus R = R^*]\right).$$

Clearly $\mathbf{H}_\infty(\sigma(\mathcal{R}) \oplus \mathcal{R}) \leq \mathbf{H}_\infty(\mathcal{R})$, with equality when σ is an orthomorphism. Assuming σ is linear permutation and fixing some distribution $\mathcal{R}$, we prove that the maximum advantage of a distinguisher making p queries to π/π^{-1} and q queries to its second oracle is at most

$$\varepsilon = \frac{pq}{2^\rho} + \frac{pq}{2^{\rho'}} + \frac{q^2}{2^{k+1}},$$

where $\rho = \mathbf{H}_\infty(\mathcal{R})$ and $\rho' = \mathbf{H}_\infty(\sigma(\mathcal{R}) \oplus \mathcal{R})$. This implies the theorem.

Fix a deterministic distinguisher D making queries to two oracles. The first is a random permutation on $\{0,1\}^k$ (and its inverse); the second oracle is $\mathcal{O}_R^{\mathsf{ccr}}(w, b) = \widehat{\mathsf{MMO}}_\sigma^\pi(R \oplus w) \oplus b \cdot R$ (for R sampled from $\mathcal{R}$) in the real world, but in the ideal world it is an independent random function from $\{0,1\}^{k+1}$ to $\{0,1\}^k$. Following the notation from Section VII-A, denote the transcript of D's interaction by $\mathcal{Q} = (\mathcal{Q}_\pi, \mathcal{Q}_\mathcal{O}, R)$. We only consider attainable transcripts from now on.

We say a transcript $(\mathcal{Q}_\pi, \mathcal{Q}_\mathcal{O}, R)$ is *bad* if either:

- (B-1) There is a query $(w, b, z) \in \mathcal{Q}_\mathcal{O}$ and a query of the form $(\sigma(R \oplus w), \star)$ or $(\star, \sigma(R \oplus w) \oplus bR \oplus z)$ in $\mathcal{Q}_\pi$.
- (B-2) There are distinct (w_i, b_i, z_i), $(w_j, b_j, z_j) \in \mathcal{Q}_\mathcal{O}$ such that $\sigma(w_i) \oplus b_i R \oplus z_i = \sigma(w_j) \oplus b_j R \oplus z_j$.

We now bound the probabilities of these events in the ideal world, beginning with (B-1). For some fixed queries $(w, b, z) \in \mathcal{Q}_\mathcal{O}$ and $(x, y) \in \mathcal{Q}_\pi$, we have

$$\Pr[\sigma(R \oplus w) = x] = \Pr[\sigma(R) = x \oplus \sigma(w)]$$

(where the probability is over choice of R), using the fact that σ is linear. Since σ is a permutation, this probability is at most $2^{-\rho}$. Similarly,

$$\Pr[\sigma(R \oplus w) \oplus bR \oplus z = y] = \Pr[\sigma(R) \oplus bR = y \oplus \sigma(w) \oplus z].$$

If $b = 0$, this probability is at most $2^{-\rho} \leq 2^{-\rho'}$ as before. If $b = 1$, this probability is at most $2^{-\rho'}$. Taking a union bound over all pairs of queries, we thus see that the probability of (B-1) is at most

$$\frac{pq}{2^\rho} + \frac{pq}{2^{\rho'}}.$$

For (B-2), consider distinct $(w_i, b_i, z_i), (w_j, b_j, z_j) \in \mathcal{Q}_\mathcal{O}$. Note that even if we condition on the value of R, the values z_i, z_j are uniform and independent. Thus,

$$\Pr[\sigma(w_i \oplus w_j) \oplus (b_i \oplus b_j) \cdot R = z_i \oplus z_j] = 2^{-k}.$$

Taking a union bound over all distinct pairs of queries, we see that the probability of (B-2) is at most $q^2/2^{k+1}$.

Fix a good transcript $(\mathcal{Q}_\pi, \mathcal{Q}_\mathcal{O}, R)$. The probability that the ideal world is consistent with this transcript is given by (1). The probability that the real world is consistent with this transcript is

$$\frac{\Pr[\forall (w,b,z) \in \mathcal{Q}_\mathcal{O} : \mathcal{O}_R^{\mathsf{ccr}}(w,b) = z \mid \pi \vdash \mathcal{Q}_\pi]}{(2^k)_p} \cdot \Pr_\mathcal{R}[R].$$

We can express the numerator of the above as

$$\prod_{i=1}^{q} \Pr[\mathcal{O}_R^{\mathsf{ccr}}(w_i,b_i) = z_i \mid \pi \vdash \mathcal{Q}_\pi \wedge \forall j < i : \mathcal{O}_R^{\mathsf{ccr}}(w_j,b_j) = z_j].$$

Note that $\mathcal{O}_R^{\mathsf{ccr}}(w_i,b_i) = z_i$ iff $\widehat{\mathsf{MMO}}_\sigma^\pi(R \oplus w_i) \oplus b_i R = z_i$, i.e., $\pi(\sigma(R \oplus w_i)) = \sigma(R \oplus w_i) \oplus b_i R \oplus z_i$. Since the transcript is good there is no query of the form $(\sigma(R \oplus w_i), \star)$ in $\mathcal{Q}_\pi$ (since (B-1) does not occur), nor is $\pi(\sigma(R \oplus w_i))$ determined by the fact that $\mathcal{O}_R^{\mathsf{ccr}}(w_j,b_j) = z_j$ for all $j < i$ (since D does not make two queries to $\mathcal{O}_R^{\mathsf{ccr}}$ with the same w_i). Similarly, there is no query of the form $(\star, \sigma(R \oplus w_i) \oplus b_i R \oplus z_i)$ in $\mathcal{Q}_\pi$ (since (B-1) does not occur), nor is $\pi^{-1}(\sigma(R \oplus w_i) \oplus b_i R \oplus z_i)$ determined by the fact that $\mathcal{O}_R^{\mathsf{ccr}}(w_j,b_j) = z_j$ for all $j < i$ (since (B-2) does not occur). Thus, for all i we have

$$\begin{aligned}\Pr[\mathcal{O}_R^{\mathsf{ccr}}(w_i,b_i) = z_i \mid \pi \vdash \mathcal{Q}_\pi \wedge \forall j < i : \mathcal{O}_R^{\mathsf{ccr}}(w_j,b_j) = z_j] \\ = 1/(2^k - p - i + 1) \;\geq\; 1/2^k.\end{aligned}$$

It follows that

$$\Pr[\forall (w,b,z) \in \mathcal{Q}_\mathcal{O} : \mathcal{O}_R^{\mathsf{ccr}}(w,b) = z \mid \pi \vdash \mathcal{Q}_\pi] \geq 1/2^{kq},$$

and so the probability that the real world is consistent with the transcript is at least (1). This completes the proof. □

Instantiating σ. There are various ways σ can be instantiated. Viewing $\{0,1\}^k$ as the field $\mathbb{F}_{2^k}$, it is easy to show that for $\alpha \neq 0, 1$ the map $\sigma(x) = \alpha \cdot x$ is a linear orthomorphism. (A common choice is $\alpha = 2$.) A more efficient solution, however, is given by $\sigma(x_L \| x_R) = x_R \oplus x_L \| x_L$ where x_L and x_R are the left and right halves of the input, respectively. This orthomorphism has received a lot attention in the context of symmetric-key cryptography [12], and we show in Section VIII that it can be implemented using a small number of instructions on modern CPUs.

D. Tweakable (Circular) Correlation Robustness

We show here a construction of a hash function that is tweakable circular correlation robust, and hence also tweakable correlation robust. (It is an interesting open question to come up with a more efficient construction satisfying the weaker notion only.) Define

$$\mathsf{TMMO}^\pi(x,i) = \pi(\pi(x) \oplus i) \oplus \pi(x).$$

Note that TMMO^π can be computed using only two calls to π.

Theorem 6. *Let $p < 2^k/2$. If π is modeled as a random permutation then TMMO^π is (p,q,ρ,ε)-tweakable circular correlation robust, where*

$$\varepsilon = \frac{4q(p+q)}{2^k} + \frac{5q^2}{2^{k+1}} + \frac{pq}{2^\rho} + \frac{q}{2^k}.$$

Proof. Fix a deterministic distinguisher D making queries to two oracles. The first is a random permutation on $\{0,1\}^k$ (and its inverse); in the real world, the second oracle is $\mathcal{O}_R^{\mathsf{tccr}}(w,i,b) = \mathsf{TMMO}^\pi(R \oplus w, i) \oplus b \cdot R$ (for R sampled from $\mathcal{R}$), but in the ideal world it is a random function from $\{0,1\}^{2k+1}$ to $\{0,1\}^k$. Following the notation from Section VII-A, denote the transcript of D's interaction by $\mathcal{Q} = (\mathcal{Q}_\pi, \mathcal{Q}_\mathcal{O}, R)$. We only consider attainable transcripts.

We say a transcript $(\mathcal{Q}_\pi, \mathcal{Q}_\mathcal{O}, R)$ is *bad* if:

- (B-1) There is a query $(w_j, i_j, b_j, z_j) \in \mathcal{Q}_\mathcal{O}$ and a query of the form $(R \oplus w_j, \star)$ in $\mathcal{Q}_\pi$.
- (B-2) There is a query $(w_j, i_j, b_j, z_j) \in \mathcal{Q}_\mathcal{O}$ such that $b_j R \oplus z_j = 0^k$.
- (B-3) There are distinct (w_j, i_j, b_j, z_j), $(w_\ell, i_\ell, b_\ell, z_\ell) \in \mathcal{Q}_\mathcal{O}$ such that $b_j R \oplus z_j = b_\ell R \oplus z_\ell$.

It is immediate that the probability of (B-1) in the ideal world is at most $pq/2^\rho$. Since each z_j is uniform and independent of R, it is similarly easy to see that the probability of (B-2) in the ideal world is at most $q/2^k$, and the probability of (B-3) in the ideal world is at most $q^2/2^{k+1}$.

Fix a good transcript $\mathcal{Q} = (\mathcal{Q}_\pi, \mathcal{Q}_\mathcal{O}, R)$. Letting $\mathcal{Q}_\mathcal{O} = \{(w_1, i_1, b_1, z_1), \ldots\}$ as above, define $u_j = R \oplus w_j$ for $1 \leq j \leq q$, and set $\mathcal{U} = \{u_1, \ldots, u_q\}$. Fixing some $\pi \vdash \mathcal{Q}_\pi$, we may define $v_j = \pi(u_j)$, $s_j = v_j \oplus i_j$, and $t_j = z_j \oplus v_j \oplus b_j R$; set $\mathcal{V} = \{v_1, \ldots, v_q\}$. Define a predicate $\mathsf{Bad}(\pi)$ on π, which is true if any of the following hold:

- (C-1) For some $1 \leq j \leq q$, there is a query of the form $(s_j, \star)$ in $\mathcal{Q}_\pi$, or $s_j \in \mathcal{U}$.
- (C-2) For some $1 \leq j \leq q$, there is a query of the form $(\star, t_j)$ in $\mathcal{Q}_\pi$, or $t_j \in \mathcal{V}$.
- (C-3) There are distinct i, j, with $1 \leq j < \ell \leq q$, such that $s_j = s_\ell$ or $t_j = t_\ell$.

We bound the probability of the above events when π is a uniform permutation, conditioned on $\pi \vdash \mathcal{Q}_\pi$.

Consider (C-1). Fixing some index j, recall that

$$s_j = v_j \oplus i_j = \pi(R \oplus w_j) \oplus i_j.$$

Since $\mathcal{Q}$ is good, $\pi(R \oplus w_j)$ is uniform in a set of size at least $2^k - p$ (and thus so is s_j). Therefore,

$$\Pr[\exists (x,y) \in \mathcal{Q}_\pi : s_j = x] \leq \frac{p}{2^k - p} \leq \frac{2p}{2^k},$$

using $p < 2^k/2$. Similarly,

$$\Pr[s_j \in \mathcal{U}] \leq \frac{|\mathcal{U}|}{2^k - p} \leq \frac{2q}{2^k}$$

(note that $\mathcal{U}$ is defined independent of π). Taking a union bound over all j, we see that the probability of (C-1) is at most $2q(p+q)/2^k$.

Next consider (C-2). Fixing some index j, recall that $t_j = z_j \oplus v_j \oplus b_j R = z_j \oplus \pi(R \oplus w_j) \oplus b_j R$ and so, arguing as above, we have

$$\Pr[\exists (x,y) \in \mathcal{Q}_\pi : t_j = y] \leq \frac{p}{2^k - p} < \frac{2p}{2^k}.$$

Fixing some $v_\ell \in \mathcal{V}$, we have $t_j = v_\ell$ iff

$$z_j \oplus \pi(R \oplus w_j) \oplus b_j R = \pi(R \oplus w_\ell).$$

The above can only possibly occur if $j \neq \ell$ since, if not, then $z_j \oplus b_j R = 0^k$ in contradiction to (B-2). But if $j \neq \ell$ then $\pi(R \oplus w_\ell)$ is uniform in a set of size at least $2^k - p - 1$ even conditioned on the value of $\pi(R \oplus w_j)$ and thus

$$\Pr[t_j = v_\ell] \leq \frac{1}{2^k - p - 1} \leq \frac{2}{2^k}$$

(using $p < 2^k/2$). Taking a union bound over all $v_\ell \in \mathcal{V}$ we see that the probability that $t_j \in \mathcal{V}$ is at most $2q/2^k$. Finally, taking a union bound over all j (and considering both sub-cases above) shows that the probability of (C-2) is at most $2q(p+q)/2^k$.

To analyze (C-3), fix distinct j, ℓ. Then $s_j = s_\ell$ iff $\pi(R \oplus w_j) \oplus i_j = \pi(R \oplus w_\ell) \oplus i_\ell$. If $w_j = w_\ell$ then $i_j \neq i_\ell$ and so $s_j = s_\ell$ is impossible. Otherwise, $\pi(R \oplus w_j)$ is uniform in $\geq 2^k - p - 1$ values even conditioned on the value of $\pi(R \oplus w_\ell)$, and thus

$$\Pr[s_j = s_\ell] \leq \frac{1}{2^k - p - 1} \leq \frac{2}{2^k}.$$

The event $t_j = t_\ell$ occurs iff

$$z_j \oplus \pi(R \oplus w_j) \oplus b_j R = z_\ell \oplus \pi(R \oplus w_\ell) \oplus b_\ell R.$$

The above can only possibly occur if $j \neq \ell$ since, if not, then $b_j R \oplus z_j = b_\ell R \oplus z_\ell$ in contradiction to (B-3). But if $w_j \neq w_\ell$ then $\pi(R \oplus w_j)$ is uniform in a set of at least $2^k - p - 1$ values, even conditioned on $\pi(R \oplus w_\ell)$, and so $\Pr[s_j = s_\ell] \leq \frac{2}{2^k}$. Taking a union bound over all distinct j, ℓ shows that the probability of (C-3) is at most $2q^2/2^k$. In summary, we have

$$\Pr[\mathsf{Bad}(\pi) \mid \pi \vdash \mathcal{Q}_\pi] \leq \frac{4q(p+q) + 2q^2}{2^k}. \tag{2}$$

The probability that the ideal world is consistent with the good transcript $\mathcal{Q}$ is exactly (1). The probability that the real world is consistent with the transcript is

$$\frac{\Pr[\forall (w,i,b,z) \in \mathcal{Q}_\mathcal{O} : \mathcal{O}_R^{\mathsf{tccr}}(w,i,b) = z \mid \pi \vdash \mathcal{Q}_\pi]}{(2^k)_p} \cdot \Pr_\mathcal{R}[R].$$

Write $\pi \vdash_j \mathcal{Q}$ if $\pi \vdash \mathcal{Q}_\pi$ and $\mathcal{O}_R^{\mathsf{tccr}}(w_\ell, i_\ell, b_\ell) = z_\ell$ for all $\ell \leq j$. The numerator above is at least

$$\begin{aligned} &\Pr[\pi \vdash_q \mathcal{Q} \wedge \neg\mathsf{Bad}(\pi) \mid \pi \vdash \mathcal{Q}_\pi] \\ &\geq \big(1 - \Pr[\mathsf{Bad}(\pi) \mid \pi \vdash \mathcal{Q}_\pi]\big) \\ &\quad \cdot \prod_{j=1}^{q} \Pr[\pi \vdash_j \mathcal{Q} \mid \neg\mathsf{Bad}(\pi) \wedge \pi \vdash_{j-1} \mathcal{Q}]. \end{aligned} \tag{3}$$

Consider any π such that $\pi \vdash \mathcal{Q}_\pi$ and $\neg\mathsf{Bad}(\pi)$. Note that $\mathcal{O}_R^{\mathsf{tccr}}(w_j, i_j, b_j) = z_j$ iff $\pi(s_j) = t_j$ (for s_j, t_j as defined before). If $\neg\mathsf{Bad}(\pi)$, there is no query of the form $(s_j, \star)$ or of the form $(\star, t_j)$ in $\mathcal{Q}_\pi$. Moreover, since neither (C-1) nor (C-2) occur, neither $\pi(s_j)$ nor $\pi^{-1}(t_i)$ is determined by the input/output relations $\{\pi(u_j) = v_j\}_{j=1,\ldots,q}$. Furthermore, since (C-3) does not occur, neither $\pi(s_j)$ nor $\pi^{-1}(t_j)$ is determined by the fact that $\pi \vdash_{j-i} \mathcal{Q}$ or, equivalently, the fact that $\pi(s_\ell) = t_\ell$ for all $\ell < j$. Thus, for all j we have

$$\Pr[\pi \vdash_j \mathcal{Q} \mid \neg\mathsf{Bad}(\pi) \wedge \pi \vdash_{j-1} \mathcal{Q}] \geq 1/2^k,$$

and therefore

$$\Pr[\pi \vdash_q \mathcal{Q} \mid \neg\mathsf{Bad}(\pi) \wedge \pi \vdash \mathcal{Q}_\pi] \geq 1/2^{kq}.$$

It follows from (3) that the ratio of the probability that the real world is consistent with $\mathcal{Q}$ to the probability that the ideal world is consistent with $\mathcal{Q}$ is at least

$$1 - \Pr[\mathsf{Bad}(\pi) \mid \pi \vdash \mathcal{Q}_\pi].$$

Using (2) completes the proof. □

VIII. Evaluation

In this section we evaluate the performance of our hash-function constructions from Section VII both on their own as well as when they are used in various protocols. The primary goal of our work is security, not efficiency. Nevertheless, we show that our work results in noticeable performance improvements over the state-of-the-art.

All timing results were obtained using an Intel(R) Xeon(R) Platinum 8124M CPU running at 3.00GHz. This CPU uses a Skylake architecture with hardware support for AES, where AES-NI has a latency of 4 cycles with a throughput of 1 cycle per instruction. All tests use only a single core. The implementations evaluated here are all publicly available [49].

A. Implementing the Hash Functions

Our constructions of correlation robust and tweakable (circular) correlation robust hash functions from Section VII only involve XORs and calls to AES. Our construction of a circular correlation robust hash function from Section VII-C, however, also requires a linear orthomorphism σ. We implement σ as described in that section using the `_mm_shuffle_epi32` instruction that is available since SSE2. This instruction allows for arbitrary permutations of the four 32-bit integers that comprise a 128-bit value. We can thus implement σ via $\sigma(a) = \texttt{\_mm\_shuffle\_epi32}(\texttt{a}, 78) \oplus \texttt{and\_si128}(\texttt{a}, \texttt{mask})$, where $\texttt{mask} = 1^{64} \| 0^{64}$ is a constant. In the CPU we used for our experiments, the `_mm_shuffle_epi32` instruction executes in 1 cycle, while the other two instructions need 0.33 cycles each, so our implementation of σ requires just 1.66 cycles. This is in contrast to prior instantiations of a linear orthomorphism based on finite-field doubling [5], which require 3.66 cycles on the same CPU.

In Table III we compare the performance of our hash functions with other symmetric-key primitives. These are:

- "SHA-256" (resp., "SHA-3") refers to computing SHA-256 (resp., SHA-3) on a 256-bit input. The implementations from openssl are used. (In Appendix D we discuss

Batch size	1	2	4	8
SHA-3	1375	1381	1374	1432
SHA-256	588	584	594	588
AES + key-sched.	81	55	37	37
XORP	48	27	22	22
doubling	51	28	16	13.5
Fixed-key AES	41	21	11	10
MMO^π	43	26	13	12
$\widehat{\mathsf{MMO}}_\sigma^\pi$	46	26	14	13
TMMO^π	81	45	25	21

TABLE III: **Performance of symmetric-key primitives.** Amortized cost per call, measured in CPU cycles, on an Intel Xeon processor.

why the concrete security of this approach is comparable to the concrete security of our constructions.)

- "AES + key-sched." refers to performing key scheduling for AES-128 (using optimizations of Gueron et al. [25]) followed by a single AES evaluation.
- "doubling" refers to computation of the function $H(x,i) = \pi(2x \oplus i) \oplus 2x \oplus i$, where π is AES-128 with a fixed key.
- "XORP" refers to computation of $H(x) = \pi_1(x) \oplus \pi_2(x)$, where π_1, π_2 are both AES-128 with fixed keys.
- "Fixed-key AES" is simply AES-128 with a fixed key.

The final three rows of the table refer to the hash functions we construct in Section VII, implemented as discussed. In each case, π is instantiated using AES-128 with a fixed key.

We evaluated the performance of the above primitives on different batch sizes (i.e., evaluating the primitive multiple times in parallel) to explore the benefit of instruction-level pipelining. We tested batch sizes as high as 32, but found little improvement above a batch size of 8. All numbers reported in the table are an average over 2^{25} experiments.

We also evaluated the performance of SHA-256 when using the SHA-NI instruction. (Note that SHA-NI is currently available only in AMD platforms and low-end Intel processors like Pentium and Celeron.) Our results (see Appendix E) show that SHA-NI is $30\times$ slower than AES-NI and hence roughly $22\times$ slower than MMO^π and $13\times$ slower than TMMO^π.

Discussion. We find that $\widehat{\mathsf{MMO}}_\sigma^\pi$ is almost as efficient as MMO^π, while TMMO^π is roughly $2\times$ slower. We can also see that $\widehat{\mathsf{MMO}}_\sigma^\pi$ is roughly $3\times$ faster than nonfixed-key AES, which directly translates to a $3\times$ improvement in the applications described in Section VI. For example, an implementation of the GGM tree using MMO^π takes roughly 6 ms to expand a 128-bit seed into one million 128-bit values, while the implementation using nonfixed-key AES takes about 21 ms for the same task. Finally, TMMO^π is $28\times$ faster than SHA-256; as we discuss further in the next section, this leads to a significant improvement in OT-extension protocols.

XORP is competitive with MMO^π and $\widehat{\mathsf{MMO}}_\sigma^\pi$ for low batch sizes, though is roughly twice slower for a batch size of 8. We stress that XORP is non-compressing, and therefore does not achieve *tweakable* (circular) correlation robustness; for that reason, it should not be compared with TMMO^π.

Protocols	Malicious			Semi-honest		
	High	Med.	Low	High	Med.	Low
S-OT (prior)	2.03	1.74	0.49	9.10	2.46	0.50
S-OT (here)	8.50	2.35	0.49	9.10	2.46	0.50
C-OT (prior)	2.04	1.76	0.74	2.17	1.84	0.74
C-OT (here)	9.17	3.60	0.74	10.9	3.66	0.74
R-OT (prior)	3.7	2.85	1.13	3.7	2.91	1.14
R-OT (here)	12.1	6.27	1.43	14.6	6.61	1.46

TABLE IV: **Performance of OT-extension protocols.** Numbers reported are in millions of OTs per second, and include the time for 128 base OTs.

B. OT Extension

As discussed in Section IV, prior constructions of malicious OT extension either rely on a cryptographic hash function like SHA-256 (modeled as a random oracle), or are constructed from fixed-key AES in an unprincipled—and often insecure—way. We can use our hash-function constructions in place of SHA-256 to achieve provable security (if we model fixed-key AES as a random permutation) with better efficiency. (We stress that malicious OT extension requires *tweakable* correlation robustness, and so XORP is inapplicable here.)

The actual improvement depends on the network speed. We benchmark the performance of our OT protocols in three settings: "High" (a 5 Gbps network), "Medium" (a 1 Gbps network), and "Low" (a 200 Mbps network). Results are summarized in Table IV, and are averaged over 2^{25} executions. S-OT, C-OT, and R-OT refer to standard-OT extension, correlated-OT extension, and random-OT extension, respectively. The table shows that we obtain a 3–4$\times$ improvement, except for the case of semi-honest standard-OT extension (where the existing implementation by Zahur et al. based on fixed-key AES is secure). End-to-end performance evaluations are presented in Appendix E.

Acknowledgments

Chun Guo is a post-doc funded by Francois-Xavier Standaert via the ERC project SWORD (724725). Work of Jonathan Katz was supported in part by DARPA and SPAWAR under contract N66001-15-C-4065. Work of Xiao Wang was supported in part by the MACS NSF project, the RISCS institute at Boston University, and a gift from PlatON network. Work of Yu Yu was supported in part by the National Natural Science Foundation of China (Grant numbers 61872236 and 61572192), the National Cryptography Development Fund (Grant number MMJJ20170209), and the Anhui Initiative in Quantum Information Technologies (Grant number AHY150100).

References

[1] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner, "More efficient oblivious transfer and extensions for faster secure computation," in *ACM Conf. on Computer and Communications Security (CCS)*. ACM, 2013, pp. 535–548.

[2] ——, "More efficient oblivious transfer extensions with security for malicious adversaries," in *Advances in Cryptology—Eurocrypt 2015, Part I*, ser. LNCS, vol. 9056. Springer, 2015, pp. 673–701.

[3] Bar Ilan Cryptography Research Group, "libscapi: The Secure Computation API," 2016.

[4] D. Beaver, "Correlated pseudorandomness and the complexity of private computations," in *28th Annual ACM Symposium on Theory of Computing (STOC)*. ACM Press, 1996, pp. 479–488.

[5] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway, "Efficient garbling from a fixed-key blockcipher," in *IEEE Symposium on Security and Privacy*, 2013, pp. 478–492.

[6] S. Bhattacharya and M. Nandi, "Full indifferentiable security of the xor of two or more random permutations using the χ^2 method," in *Advances in Cryptology—Eurocrypt 2018, Part I*, ser. LNCS. Springer, 2018, pp. 387–412.

[7] A. Biryukov and D. Wagner, "Advanced slide attacks," in *Advances in Cryptology—Eurocrypt 2000*, ser. LNCS, vol. 1807. Springer, 2000, pp. 589–606.

[8] J. Black, M. Cochran, and T. Shrimpton, "On the impossibility of highly-efficient blockcipher-based hash functions," *Journal of Cryptology*, vol. 22, no. 3, pp. 311–329, 2009.

[9] D. Bogdanov, L. Kamm, B. Kubo, R. Rebane, V. Sokk, and R. Talviste, "Students and taxes: A privacy-preserving study using secure computation," *Proc. Privacy Enhancing Technologies*, no. 3, pp. 117–135, 2016.

[10] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. I. Schwartzbach, and T. Toft, "Secure multiparty computation goes live," in *Financial Cryptography and Data Security (FC)*, ser. LNCS. Springer, 2009, pp. 325–343.

[11] E. Boyle, N. Gilboa, and Y. Ishai, "Function secret sharing," in *Advances in Cryptology—Eurocrypt 2015, Part II*, ser. LNCS, vol. 9057. Springer, 2015, pp. 337–367.

[12] S. Chen, R. Lampe, J. Lee, Y. Seurin, and J. P. Steinberger, "Minimizing the two-round Even-Mansour cipher," in *Advances in Cryptology—Crypto 2014, Part I*, ser. LNCS, vol. 8616. Springer, 2014, pp. 39–56.

[13] S. Chen and J. P. Steinberger, "Tight security bounds for key-alternating ciphers," in *Advances in Cryptology—Eurocrypt 2014*, ser. LNCS, vol. 8441. Springer, 2014, pp. 327–350.

[14] S. G. Choi, J. Katz, R. Kumaresan, and H.-S. Zhou, "On the security of the "free-XOR" technique," in *Theory of Cryptography Conference 2012*, ser. LNCS, vol. 7194. Springer, 2012, pp. 39–53.

[15] H. Corrigan-Gibbs, D. Boneh, and D. Mazières, "Riposte: An anonymous messaging system handling millions of users," in *IEEE Symposium on Security and Privacy*, 2015, pp. 321–338.

[16] W. Dai, V. T. Hoang, and S. Tessaro, "Information-theoretic indistinguishability via the chi-squared method," in *Advances in Cryptology—Crypto 2017, Part III*, ser. LNCS. Springer, 2017, pp. 497–523.

[17] D. Demmler, T. Schneider, and M. Zohner, "ABY—A framework for efficient mixed-protocol secure two-party computation," in *Network and Distributed System Security Symposium*, 2015.

[18] Y. Dodis, L. Reyzin, R. L. Rivest, and E. Shen, "Indifferentiability of permutation-based compression functions and tree-based modes of operation, with applications to MD6," in *Fast Software Encryption (FSE) 2009*, ser. LNCS. Springer, 2009, pp. 104–121.

[19] J. Doerner and A. Shelat, "Scaling ORAM for secure computation," in *ACM Conf. on Computer and Communications Security (CCS)*. ACM, 2017, pp. 523–535.

[20] X. Fan, C. Ganesh, and V. Kolesnikov, "Hashing garbled circuits for free," in *Advances in Cryptology—Eurocrypt 2017, Part III*, ser. LNCS. Springer, 2017, pp. 456–485.

[21] A. Faz-Hernández, J. López, and A. K. D. S. de Oliveira, "SoK: A performance evaluation of cryptographic instruction sets on modern architectures," in *Proc. 5th ACM on ASIA Public-Key Cryptography Workshop*. ACM, 2018, pp. 9–18.

[22] T. K. Frederiksen, T. P. Jakobsen, J. B. Nielsen, and R. Trifiletti, "TinyLEGO: An interactive garbling scheme for maliciously secure two-party computation," Cryptology ePrint Archive, Report 2015/309, 2015, http://eprint.iacr.org/2015/309.

[23] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," *J. ACM*, vol. 33, no. 4, pp. 792–807, Oct. 1986.

[24] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or A completeness theorem for protocols with honest majority," in *19th Annual ACM Symposium on Theory of Computing (STOC)*. ACM Press, 1987, pp. 218–229.

[25] S. Gueron, Y. Lindell, A. Nof, and B. Pinkas, "Fast garbling of circuits under standard assumptions," in *ACM Conf. on Computer and Communications Security (CCS)*. ACM, 2015, pp. 567–578.

[26] M. Hastings, B. Hemenway, D. Noble, and S. Zdancewic, "SoK: General purpose compilers for secure multi-party computation," in *IEEE Symposium on Security and Privacy*, 2019.

[27] B. Hemenway, S. Lu, R. Ostrovsky, and W. Welser IV, "High-precision secure computation of satellite collision probabilities," in *Intl. Conf. on Security and Cryptography for Networks (SCN)*, ser. LNCS. Springer, 2016, pp. 169–187.

[28] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, "Extending oblivious transfers efficiently," in *Advances in Cryptology—Crypto 2003*, ser. LNCS, vol. 2729. Springer, 2003, pp. 145–161.

[29] M. Keller, E. Orsini, and P. Scholl, "Actively secure OT extension with optimal overhead," in *Advances in Cryptology—Crypto 2015, Part I*, ser. LNCS, vol. 9215. Springer, 2015, pp. 724–741.

[30] ——, "MASCOT: Faster malicious arithmetic secure computation with oblivious transfer," in *ACM Conf. on Computer and Communications Security (CCS)*. ACM, 2016, pp. 830–842.

[31] V. Kolesnikov and T. Schneider, "Improved garbled circuit: Free XOR gates and applications," in *Intl. Colloquium on Automata, Languages, and Programming (ICALP)*, ser. LNCS. Springer, 2008, pp. 486–498.

[32] A. Lapets, F. Jansen, K. D. Albab, R. Issa, L. Qin, M. Varia, and A. Bestavros, "Accessible privacy-preserving web-based data analysis for assessing and addressing economic inequalities," in *Proc. 1st ACM SIGCAS Conference on Computing and Sustainable Societies*. ACM, 2018, pp. 48:1–48:5.

[33] J. Lee, "Indifferentiability of the sum of random permutations toward optimal security," *IEEE Trans. Info. Theory*, vol. 63, no. 6, pp. 4050–4054, 2017.

[34] M. Liskov, R. L. Rivest, and D. Wagner, "Tweakable block ciphers," *Journal of Cryptology*, vol. 24, no. 3, pp. 588–613, Jul. 2011.

[35] S. Lucks, "The sum of PRPs is a secure PRF," in *Advances in Cryptology—Eurocrypt 2000*, ser. LNCS, vol. 1807. Springer, 2000, pp. 470–484.

[36] A. Luykx, B. Mennink, B. Preneel, and L. Winnen, "Two-permutation-based hashing with binary mixing," *J. Mathematical Cryptology*, pp. 139–150, 2015.

[37] A. Mandal, J. Patarin, and V. Nachef, "Indifferentiability beyond the birthday bound for the xor of two public random permutations," in *Progress in Cryptology—Indocrypt 2010*, ser. LNCS. Springer, 2010, pp. 69–81.

[38] U. M. Maurer, R. Renner, and C. Holenstein, "Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology," in *Theory of Cryptography Conference*, ser. LNCS, vol. 2951. Springer, 2004, pp. 21–39.

[39] B. Mennink and B. Preneel, "On the XOR of multiple random permutations," in *Intl. Conference on Applied Cryptography and Network Security (ACNS)*, ser. LNCS. Springer, 2015, pp. 619–634.

[40] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *IEEE Symposium on Security and Privacy*, 2017, pp. 19–38.

[41] J. B. Nielsen, T. Schneider, and R. Trifiletti, "Constant round maliciously secure 2PC with function-independent preprocessing using LEGO," in *Network and Distributed System Security Symposium*, 2017.

[42] J. Patarin, "The "coefficients H" technique (invited talk)," in *Annual International Workshop on Selected Areas in Cryptography (SAC)*, ser. LNCS. Springer, 2009, pp. 328–345.

[43] M. O. Rabin, "How to exchange secrets with oblivious transfer," Cryptology ePrint Archive, Report 2005/187, Aiken Computation Lab, Harvard University, Tech. Rep. TR-81, 1981, available at `http://eprint.iacr.org/2005/187`.

[44] P. Rindal, "libOTe: an efficient, portable, and easy to use Oblivious Transfer Library," https://github.com/osu-crypto/libOTe, 41c5505262708136336437 0ba7f7893b3c413951.

[45] Unbound Tech, "Protecting cryptographic signing keys and seed secrets with multi-party computation." https://github.com/unbound-tech/blockchain-crypto-mpc, 2018, 94d5b83dc83e920a668d6e737c0a720c3abca7dc.

[46] University of Bristol, "Apricot: Advanced protocols for real-world implementation of computational oblivious transfers." https://github.com/bristolcrypto/apricot, 2016, 3760dda51b0080ee0fb79c7184cbe2c00762c2b8.

[47] ——, "Multiparty computation with SPDZ, MASCOT, and overdrive offline phases (inactive)." https://github.com/bristolcrypto/SPDZ-2, 2016, 721abfae849625a02ea49aabc534f9cf41ca643f.
[48] F. Wang, C. Yun, S. Goldwasser, V. Vaikuntanathan, and M. Zaharia, "Splinter: Practical private queries on public data," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 299–313.
[49] X. Wang, A. J. Malozemoff, and J. Katz, "EMP-toolkit: Efficient MultiParty computation toolkit," https://github.com/emp-toolkit, 2016.
[50] X. Wang, S. Ranellucci, and J. Katz, "Authenticated garbling and efficient maliciously secure two-party computation," in *ACM Conf. on Computer and Communications Security (CCS)*. ACM, 2017, pp. 21–37.
[51] ——, "Global-scale secure multiparty computation," in *ACM Conf. on Computer and Communications Security (CCS)*. ACM, 2017, pp. 39–56.
[52] A. C.-C. Yao, "How to generate and exchange secrets (extended abstract)," in *27th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 1986, pp. 162–167.
[53] S. Zahur, M. Rosulek, and D. Evans, "Two halves make a whole—reducing data transfer in garbled circuits using half gates," in *Advances in Cryptology—Eurocrypt 2015, Part II*, ser. LNCS, vol. 9057. Springer, 2015, pp. 220–250.
[54] R. Zhu and Y. Huang, "JIMU: Faster LEGO-based secure computation using additive homomorphic hashes," in *Advances in Cryptology—Asiacrypt 2017, Part II*, ser. LNCS. Springer, 2017, pp. 529–572.

Appendix A
Random-OT Extension

Random OT is similar to correlated OT except that both "messages" of the sender are chosen uniformly by the ideal functionality; see Figure 6. It also has applications to secure computation [1]. In Figure 7 we show a protocol realizing this functionality in the $\mathcal{F}_{\Delta\text{-ROT}}$-hybrid model, in both the semi-honest and malicious settings.

Theorem 7. *If H is* cr *(resp.,* tcr*) then protocol $\Pi_{\text{R-OT}}$ securely realizes $\mathcal{F}_{\text{R-OT}}$ for semi-honest (resp., malicious) adversaries in the $\mathcal{F}_{\Delta\text{-ROT}}$-hybrid model.*

Proof. The proof is very similar to the proof of Theorem 2. As in that case, security for a corrupted $\mathsf{P_A}$ is perfect and easy to show, and we thus focus on a corrupted $\mathsf{P_B}$. We consider the malicious setting; the semi-honest setting follows similarly.

The simulator $\mathcal{S}$ for a malicious $\mathsf{P_B}$ is as follows:

1-2. $\mathcal{S}$ obtains the inputs $(x_1, \ldots, x_m)$, as well as the values $\{\mathbf{b}_i\}_{i\in[m]}$, that $\mathcal{A}$ sends to $\mathcal{F}_{\Delta\text{-ROT}}$,
$\mathcal{S}$ also chooses a uniform Δ and answers $\mathcal{A}$'s global key query (if any) using Δ.
3. $\mathcal{S}$ sets $\mathbf{m}_i^{x_i} := H(\mathbf{b}_i, i)$ and sends $(x_1, \ldots, x_m)$ and $\{\mathbf{m}_i^{x_i}\}$ to $\mathcal{F}_{\text{R-OT}}$.

A proof of indistinguishability follows as in the proof of Theorem 1. □

Appendix B
On Defining Correlated OT

We give an explicit attack showing that the protocol for correlated-OT extension by Asharov et al. [2] does not realize their correlated-OT functionality in the malicious setting. A similar attack also works for random-OT extension.

In Figure 8 we show the original correlated-OT functionality as defined by Asharov et al. [2]. Compared to our version of the ideal functionality in Figure 4, this ideal functionality is stronger since it does not allow the adversary to specify the output it receives. In particular, a malicious $\mathsf{P_B}$ is no longer allowed to specify the values $\{\mathbf{m}_i^{x_i}\}$. Asharov et al. propose a protocol that is the same as ours (cf. Figure 5), and claim security when H is modeled as a random oracle.

Since a malicious $\mathsf{P_B}$ can fully determine the values of $\mathbf{b}_i$, however, it can clearly choose them in such a way that each of the $\{\mathbf{m}_i^{x_i}\}$ satisfy some predicate, e.g., so that the least-significant bit of $\mathbf{m}_i^{x_i}$ is equal to 0 for all i. This holds true regardless of how H is instantiated.

Functionality $\mathcal{F}_{\text{R-OT}}$

Upon receiving (Extend) from $\mathsf{P_A}$ and $(x_1, \ldots, x_m)$ from $\mathsf{P_B}$ with $x_i \in \{0,1\}$ do:

1) For each $i \in [m]$, sample uniform $\mathbf{m}_i^0, \mathbf{m}_i^1 \in \{0,1\}^k$.
2) If $\mathsf{P_A}$ is corrupted, wait for $\mathcal{A}$ to send $\{(\mathbf{m}_i^0, \mathbf{m}_i^1)\}$.
If $\mathsf{P_B}$ is corrupted, wait for $\mathcal{A}$ to send $\{\mathbf{m}_i^{x_i}\}$.
3) Send $\{(\mathbf{m}_i^0, \mathbf{m}_i^1)\}$ to $\mathsf{P_A}$ and $\{\mathbf{m}_i^{x_i}\}$ to $\mathsf{P_B}$.

Fig. 6. Functionality $\mathcal{F}_{\text{R-OT}}$.

Protocol $\Pi_{\text{R-OT}}$

Inputs: $\mathsf{P_B}$ has $x_1, \ldots, x_m$ with $x_i \in \{0,1\}$.

Protocol:

1) $\mathsf{P_A}$ chooses uniform Δ and sends (Init, Δ) to $\mathcal{F}_{\Delta\text{-ROT}}$; $\mathsf{P_B}$ sends (Init) to $\mathcal{F}_{\Delta\text{-ROT}}$.
2) $\mathsf{P_B}$ sends $(x_1, \ldots, x_m)$ to $\mathcal{F}_{\Delta\text{-ROT}}$, which returns $\mathbf{a}_1, \ldots, \mathbf{a}_m$ to $\mathsf{P_A}$ and $\mathbf{b}_1, \ldots, \mathbf{b}_m$ to $\mathsf{P_A}$.
3) **Semi-honest security:** For $i \in [m]$ and $b \in \{0,1\}$, $\mathsf{P_A}$ computes $\mathbf{m}_i^b := H(\mathbf{a}_i \oplus b\Delta)$; $\mathsf{P_B}$ sets $\mathbf{m}_i^{x_i} := H(\mathbf{b}_i)$.

Malicious security: For $i \in [m]$ and $b \in \{0,1\}$, $\mathsf{P_A}$ computes $\mathbf{m}_i^b := H(\mathbf{a}_i \oplus b\Delta, i)$; $\mathsf{P_B}$ sets $\mathbf{m}_i^{x_i} := H(\mathbf{b}_i, i)$.

Fig. 7. Protocol $\Pi_{\text{R-OT}}$.

Functionality $\mathcal{F}'_{\text{C-OT}}$

Upon receiving $(\Delta_1, \ldots, \Delta_m)$ from $\mathsf{P_A}$ and $(x_1, \ldots, x_m)$ from $\mathsf{P_B}$ with $x_i \in \{0,1\}$ do:

1) For each $i \in [m]$, sample uniform $\mathbf{m}_i^0$ and set $\mathbf{m}_i^1 := \mathbf{m}_i^0 \oplus \Delta_i$.
2) Send $\{(\mathbf{m}_i^0, \mathbf{m}_i^1)\}$ to $\mathsf{P_A}$ and $\{\mathbf{m}_i^{x_i}\}$ to $\mathsf{P_B}$.

Fig. 8. The original functionality $\mathcal{F}'_{\text{C-OT}}$ for correlated OT proposed by Asharov et al. [1, 2].

Appendix C
Attack on MASCOT

Recall that the MASCOT paper [30] recommends instantiating the hash function as $H(x, i) = \pi(x) \oplus x$, and this was implemented by SPDZ-2 [47]. In Section II, we showed an attack on the OT-extension protocol using this H. Here, we show that the same ideas can be used to attack the triple-generation protocol used in MASCOT in the $\mathcal{F}_{\Delta\text{-ROT}}$ hybrid model, which in turn can be used to violate privacy of the overall MASCOT protocol.

The attack on MASCOT discussed in Section II focuses on standard OT, but applies to random OT as well. In particular, a corrupted receiver can send the same choice bits (denoted as x) and messages (denoted as $\mathbf{b}$) to $\mathcal{F}_{\Delta\text{-ROT}}$ across all executions i. In this case, the $\mathbf{a}_i$ values obtained by the sender are also the same for all i since $\mathbf{a}_i = \mathbf{b}_i \oplus x_i\Delta = \mathbf{b} \oplus x\Delta$. Thus, the output of the sender (namely, m_i^0 and m_i^1) is also the same for all i, since they are computed as a deterministic function of $\mathbf{a}_i$ and Δ and are independent of i if the hash function H does not depend on i.

This can be used to attack the triple-generation protocol in MASCOT. Roughly, the MASCOT triple-generation protocol works by having each party P_i choose random $a_1^{(i)}, \ldots, a_\tau^{(i)}, b^{(i)}$ in a field $\mathbb{F}$, where $b^{(i)}$ will be the share of the (secret) b-value of the output triple, and some linear combination of $a_1^{(i)}, \ldots, a_\tau^{(i)}$ will be the share of the (secret) a-value of the output triple. Party P_j then obtains some random pads $q_{0,h}^{(j,i)}, q_{1,h}^{(j,i)}$ by playing the role of a sender in an execution of random OT, and later sends $q_{0,h}^{(j,i)} - q_{1,h}^{(j,i)} + b^{(j)}$ to P_i. If P_i is corrupted, and the adversary makes $q_{0,h}^{(j,i)}, q_{1,h}^{(j,i)}$ the same across two executions of the triple-generation protocol, then the adversary can easily learn the difference of the b-values used in two output triples. When such triples are used in the online phase, this allows the attacker to learn the difference of two secret values.

In detail, the attack works on Protocol 4 of the MASCOT paper, which is used to generate authenticated SPDZ triples. For simplicity, assume there are only two parties P_1 and P_2. (The attack can be easily generalized to any number of parties), and the adversary corrupts P_1. The attack proceeds as follows. The adversary first picks a bit a and a κ-bit string $\mathbf{b}$.

1) In step 2(a), the adversary (acting as the receiver in an execution of random OT) sends $(a, a, ..., a) \in \mathbb{F}_2^{\tau\kappa}$ to the $\mathcal{F}_{\Delta\text{-ROT}}$ ideal functionality, and requests $\mathcal{F}_{\Delta\text{-ROT}}$ to use the *same* string $\mathbf{b}$ for all h. By doing so, the adversary obtains the value $s_h^{(1,2)}$ that is the same across for all h, denoted as s_a.
2) The ideal functionality in step 2(b) sends $q_{0,h}^{(2,1)}, q_{1,h}^{(2,1)} \in \mathbb{F}$ to the sender. Due to the above attack, we know that $q_{a,h}^{(2,1)} = s_a$ for all h, and for some (random) value $s_{\bar{a}}$ it holds that $q_{\bar{a},h}^{(2,1)} = s_{\bar{a}}$ for all h.
3) In step 2(c), the adversary receives $d_h^{(2,1)}$ values from the honest party that are all equal (since $d_h^{(2,1)} = q_{0,h}^{(2,1)} - q_{1,h}^{(2,1)} + b^{(2)} = s_0 - s_1 + b^{(2)}$).
4) For each triple, the adversary uses the same values of a and $\mathbf{b}$, which forces all s_0, s_1 to be same same. So for generation of two triples, where the honest party uses $b^{(2)}$ and $b'^{(2)}$, the d values sent from the honest party are $d = s_0 - s_1 + b^{(2)}$ and $d' = s_0 - s_1 + b'^{(2)}$, where $b^{(2)} - b'^{(2)}$ can be computed by the adversary.

The above allows the adversary to learn all the SPDZ triples except for the first. Since these are used in the online phase to mask private inputs, this means that if the attacker knows one bit of the honest party's input then it can deduce the honest party's entire input. (Equivalently, without knowing anything a priori the attacker learns that the honest party's input is one of two possibilities.)

Batch size	1	2	4	8
SHA-3	1166	1188	1160	1163
SHA-256 (SHA-NI)	223	179	185	252
AES + key-sched.	78	58	34	34
XORP	45	29	18	12
doubling	51	29	16	10
Fixed-key AES	38	22	11	6
MMO^π	40	29	13	8
$\widehat{\mathsf{MMO}}_\sigma^\pi$	42	25	15	9
TMMO^π	79	46	25	14

TABLE V: **Performance of symmetric-key primitives.** Amortized cost per call, measured in CPU cycles, on an AMD processor with SHA-NI enabled.

APPENDIX D
CONCRETE SECURITY OF CORRELATION ROBUSTNESS

Our constructions in Section VII all achieve concrete security $O((pq+q^2)/2^k)$, assuming R is uniform for simplicity. We show that *no* hash function H can achieve concrete security better than $O(pq/2^k)$ for correlation robustness—even if H is a random oracle. For applications of correlation robustness to secure computation one would generally have $p \gg q$, and the concrete security of our constructions is asymptotically optimal when that holds.

We show a explicit attack with p queries to H and q queries to $\mathcal{O}_R^{\mathsf{cr}}(x) = H(R \oplus x)$ that has distinguishing advantage $O(pq/2^k)$:

1) Evaluate H on p uniform inputs, i.e., compute the values $z_1^* = H(v_1^*), \ldots, z_p^* = H(v_q^*)$.
2) Query $\mathcal{O}_R^{\mathsf{cr}}$ on q uniform inputs, i.e., obtain the results $z_1 = \mathcal{O}(x_1), \ldots, z_q = \mathcal{O}(x_q)$.
3) If $z_i^* = z_j$ for some i, j, then set $R^* = v_i^* \oplus x_j$ as a candidate guess for R. This guess can be verified using one additional query to each of H and $\mathcal{O}_R^{\mathsf{cr}}$.

(This is a "slide with a twist attack" [7] adapted to our setting.) The distinguishing advantage of this attack is $O(pq/2^k)$.

APPENDIX E
ADDITIONAL PERFORMANCE EVALUATIONS

A. Microbenchmarks on AMD CPUs

We measured the performance of various primitives using an AMD EPYC 7000 CPU available on an Amazon EC2 instance of type `m5ad.xlarge`. Here, the implementation of SHA-256 is from openssl and uses SHA-NI. When running multiple evaluations of SHA-256, we take advantage of pipelining using the implementation of Faz-Hernández et al. [21]. We observe a 20% improvement in running time when using a batch size of 2, but the performance is worse when the batch size is larger. This is consistent with the results reported by Faz-Hernández et al., who explained that this is due to the

	High	Med.	Low
MASCOT			
Prior work (SHA-256)	4.29 s	6.7 s	18.2 s
Here (AES)	1.92 s	4.3 s	16.1 s
Improvement	2.2×	1.6×	1.1×
SecureML			
Prior (SHA-256)	270 ms	320 ms	700 ms
This work (AES)	68 ms	150 ms	700 ms
Improvement	4.0×	2.1×	–

TABLE VI: **Performance of two OT-based triple-generation protocols on an Intel processor.**

limited CPU register size. The results of our experiments are summarized in Table V.

B. End-to-end Performance Improvement

To evaluate the performance improvement when using our improved design in end-to-end MPC protocols, we benchmarked the performance of MASCOT and SecureML [40] when using our optimized OT-extension protocols. MASCOT is a state-of-the-art protocol for multi-party computation with malicious security that relies on SPDZ-triple generation based on OT extension; SecureML uses OT extension for (semi-honest) generation of Beaver matrix triples. Since triple generation dominates the overall cost in either case, we focus on the performance of that step.

Similar to the performance of OT extension, the actual improvement depends on the network speed, and the performance is measured in three settings: "High" with a 5 Gbps network, "Medium" with a 1 Gbps network, and "Low" with a 200 Mbps network. In Table VI, we show the performance of these two triple-generation protocols in various network settings. The running time for MASCOT is for generating 10^4 128-bit SPDZ triples, and the running time for SecureML is for generating Beaver matrix triples of dimension 128×128, where each element is 64-bits long. Over a high-bandwidth network, we observe around $4.0\times$ improvement for SecureML and $2.2\times$ improvement for MASCOT; as for a medium-bandwidth network, we observe an $2.1\times$ improvement for SecureML and $1.6\times$ for MASCOT. We also performed the same benchmark on an AMD platform with SHA-NI, with results reported in Table VII.

We remark that the recent secure-computation protocols based on authenticated garbling [50, 51] do not use the same type of OT protocols described in this paper. Instead, they rely on the globally correlated OT-hybrid model [41], which does not need any (tweakable) correlation robust hash function. Therefore our improvements do not apply to their protocols.

	High	Med.	Low
MASCOT			
Prior work (SHA-256)	2.8 s	5.13 s	16.8 s
Here (AES)	1.64 s	3.99 s	15.6 s
Improvement	1.7×	1.3×	1.07×
SecureML			
Prior (SHA-256)	170 ms	220 ms	700 ms
This work (AES)	70 ms	144 ms	700 ms
Improvement	2.4×	1.5×	–

TABLE VII: **Performance of two OT-based triple-generation protocols on an AMD processor.**