Extending Oblivious Transfer with Low Communication via Key-Homomorphic PRFs

Peter Scholl

Aarhus University, Denmark
peter.scholl@cs.au.dk

Abstract. We present a new approach to extending oblivious transfer with communication complexity that is logarithmic in the security parameter. Our method only makes black-box use of the underlying cryptographic primitives, and can achieve security against an active adversary with almost no overhead on top of passive security. This results in the first oblivious transfer protocol with sublinear communication and active security, which does not require any non-black-box use of cryptographic primitives.
Our main technique is a novel twist on the classic OT extension of Ishai et al. (Crypto 2003), using an additively key-homomorphic PRF to reduce interaction. We first use this to construct a protocol for a large batch of 1-out-of-n OTs on random inputs, with amortized $o(1)$ communication. Converting these to 1-out-of-2 OTs on chosen strings requires logarithmic communication. The key-homomorphic PRF used in the protocol can be instantiated under the learning with errors assumption with exponential modulus-to-noise ratio.

1 Introduction

In an oblivious transfer protocol, a receiver wishes to learn a subset of some messages held by a sender, whilst hiding exactly which messages are received. A common type of oblivious transfer is 1-out-of-2 OT, where the sender holds messages x_0, x_1, while the receiver holds a bit b and wishes to learn x_b. The protocol should guarantee that the receiver learns no information on x_{1-b}, whilst the sender learns nothing about b. 1-out-of-2 OT is a key tool in building secure two-party and multi-party computation protocols, and most efficient protocols need to use a very large number of oblivious transfers that scales with the input size [Yao86], or the size of the circuit description of the function being computed [GMW87].

All known protocols for oblivious transfer are much more expensive than standard symmetric-key primitives, as they rely on public-key cryptography. This property seems to be inherent, since it is known that constructing OT from symmetric cryptographic primitives in a black-box manner is impossible [IR89]. An essential technique for reducing the cost of oblivious transfer is *OT extension*, which reduces the cost of carrying out many OTs with amortization. OT extension protocols proceed in two stages: in a setup phase, a small number of 'seed' OTs are created using standard public-key techniques; secondly, these are extended to create many more, independent OTs, with a lower cost than the seed OT protocols. Typically the second phase is based only on cheap, symmetric cryptography, so using OT extension allows many OTs to be created with only $O(k)$ public-key operations for security parameter k, greatly reducing the computational costs.

OT Extension: A Brief History. Classically, OT extension refers to evaluating OT using mainly symmetric key cryptography. In this paper we broaden the term to cover any protocol that generates $m = \mathsf{poly}(k)$ OTs in a way which is more efficient than executing m instances of an OT protocol. Reducing communication for the case of (1-out-of-2) *bit-OT*, where the sender's messages are bits, is of particular importance. This is exactly what is needed in the GMW protocol for secure multi-party computation [GMW87,Gol04], and a bit-OT protocol with $O(1)$ communication complexity implies secure computation with *constant overhead* using GMW, and even with active security [IPS08].

Beaver [Bea96] first showed how to convert $O(k)$ seed OTs into any polynomial number of OTs, using only one-way functions, for security parameter k. In this technique, the parties use a secure two-party computation protocol to evaluate the circuit that takes as input a random seed from each party, then applies a PRG and computes the OT functionality on the expanded, random inputs. With Yao's protocol [Yao86] this only needs $O(k)$ OTs, since the inputs are of size $O(k)$.

The 'IKNP' protocol, by Ishai, Kilian, Nissim and Petrank [IKNP03], lies at the core of all recent, practical OT extension protocols. IKNP was the first protocol to efficiently extend OT in a *black-box* manner, using only a hash function which satisfies a correlation robustness assumption (or a random oracle). The communication complexity of this protocol is $O(k + \ell)$ bits per extended OT with passive security, where ℓ is the bit length of the sender's strings. Harnik et al. [HIKN08] later showed how to obtain active security with the same asymptotic efficiency. The TinyOT protocol [NNOB12] introduced the first practical, actively secure OT extension, and more recently the overhead of active security has been reduced to almost nothing for both the 1-out-of-2 [ALSZ15,KOS15] and 1-out-of-n cases [OOS17,PSS17].

These protocols are essentially optimal for transferring messages of length $\ell = \Omega(k)$, but when ℓ is short (as in bit-OT where $\ell = 1$) there is still an overhead in $O(k)$. Kolesnikov and Kumaresan [KK13] presented a variant of the IKNP protocol based on 1-out-of-n OT, which can be used to perform 1-out-of-2 bit-OT with $O(k/\log k)$ communication. It is not known how to make this bit-OT protocol actively secure, because it relies on a passively secure reduction from 1-out-of-n to 1-out-of-2 OT [NP99].

Unfortunately, all known methods for achieving *constant-communication* bit-OT use very complex techniques, and often require non-black-box use of cryptographic primitives. Ishai et al. [IKOS08] combined Beaver's non-black-box technique [Bea96] for OT extension with a polynomial-stretch PRG in NC0 and randomized encodings, to obtain a passively secure protocol with amortized $O(1)$ computational overhead (implying $O(1)$ communication). As well as needing a strong assumption on the PRG, a major drawback is the use of non-black-box techniques, which lead to a very high constant. The same authors later gave an alternative, black-box approach with constant communication [IKOS09]. However, this still requires heavy machinery such as algebraic geometry codes, randomized encodings and low-communication PIR. Additionally, achieving active security would require generic use of zero-knowledge proofs [GMW86,IKOS07], again with non-black-box use of the underlying primitives. Recently, Boyle et al. [BGI17] showed how to obtain an amortized communication cost of just *4 bits per bit-OT* using homomorphic secret-sharing, which can be realised from either DDH [BGI16] or LWE [DHRW16]. As with the previous works, however, this construction makes non-black-box use of PRGs and would be extremely inefficient in practice.

Finally, we remark that using indistinguishability obfuscation and fully homomorphic encryption, it is possible to produce $\mathsf{poly}(k)$ OTs on random inputs with a communication complexity that is independent of the number of OTs, with a general method for reusable correlated randomness in secure computation [HW15,HIJ+16].

1.1 Contributions of This Work

We present a new approach to extending oblivious transfer with low communication. Our protocol, in the random oracle model, makes black-box use of the underlying cryptographic primitives and can achieve security against an active adversary with almost no overhead on top of passive security. This results in the first bit-OT protocol with sublinear communication and active security, making only black-box use of cryptographic primitives. Table 1 compares the characteristics of our protocol with some of the other OT extension protocols discussed earlier.

Our main technique is a novel twist on the classic IKNP OT extension, using an additively key-homomorphic PRF to reduce interaction. The main challenge here is to handle the homomorphism error present in known key-homomorphic PRF constructions, without compromising on correctness or security. We first present a protocol for a large batch of 1-out-of-p_i OTs on random inputs, for multiple, distinct primes p_i. The communication complexity of this protocol is *sublinear in the total number of random OTs*, with an amortized cost of $o(1)$ bits per OT. It was not known previously how to achieve this without using obfuscation,[1] and this primitive may be useful in wider applications. If we want to obtain 1-out-of-2 OT on chosen strings, each 1-out-of-p_i random OT can be converted to a 1-out-of-2 OT with $O(\log p_i)$ bits of communication using standard techniques, giving logarithmic communication overhead.

[1] Even with obfuscation, secure computation with complexity sublinear in the output size and active security is known to be impossible [HW15]. The obfuscation-based protocol of [HIJ+16] circumvents this using a CRS, whilst we use the random oracle model.

Protocol	Communication per OT (bits)	Security	Based on	Black-box
[Bea96]	$\mathsf{poly}(k)$	passive	OWF	✗
[IKNP03]	$O(k)$	passive	CRH/RO	✓
[KK13]	$O(k/\log k)$	passive	CRH/RO	✓
[ALSZ15,KOS15]	$O(k)$	active	CRH/RO	✓
[IKOS08]	$O(1)$	passive	poly-stretch local PRG	✗
[IKOS09]	$O(1)$	passive	Φ-hiding	✓
[BGI17]	$4 + o(1)$	passive	DDH	✗
[HIJ$^+$16] ($\binom{n}{1}$-ROT)	$o(1)$	active	iO + FHE	✗
This work ($\binom{p_i}{1}$-ROT[a])	$o(1)$	active	LWE + RO	✓
This work ($\binom{2}{1}$-OT)	$O(\log k)$	active	LWE + RO	✓

Table 1. Various protocols for extending 1-out-of-2 bit-OT (unless otherwise specified) with different assumptions. All passively secure protocols can be transformed to be actively secure using non-black-box zero-knowledge techniques. CRH is a correlation-robust hash function, RO is a random oracle; $\binom{n}{1}$-ROT is 1-out-of-n OT on random inputs.

[a] p_i are small distinct primes

The additively key-homomorphic PRF needed in our protocol can be instantiated based on the learning with errors assumption with an exponential modulus-to-noise ratio. This assumption has previously been used to construct attribute-based encryption [GVW13] and fully key-homomorphic encryption [BGG$^+$14], and is believed to be hard if the LWE dimension is chosen large enough to thwart known attacks. The downside of our approach is that this spectrum of LWE parameters results in fairly heavy computational costs for the parties, so it seems that the main uses of our protocol would be in low bandwidth environments where communication is much more expensive than computation.

As a contribution of independent interest, to implement the base OTs in our protocol we generalise the consistency check used for OT extension by Asharov et al. [ALSZ15], and adapt it for producing *correlated OTs* over any abelian group $\mathbb{G}$, instead of just XOR correlations over bit strings. These are 1-out-of-2 OTs where the sender's messages are all guaranteed to be of the form $(x_i, x_i + \Delta)$, for some fixed correlation $\Delta \in \mathbb{G}$. We also identify a crucial security flaw in the original protocol of Asharov et al. [ALSZ15,ALSZ17a], which leaks information on the receiver's inputs to a passively corrupted sender. After reporting this to the authors, their protocol has been modified to fix this [ALSZ17b], and we use the same fix for our protocol.

1.2 Overview of Techniques

IKNP OT Extension. We first recall the IKNP OT extension protocol [IKNP03] (with optimizations from [ALSZ13,KK13]), which uses k instances of oblivious transfer to construct $m = \mathsf{poly}(k)$ oblivious transfers with only a cryptographic hash function and a pseudorandom generator. The parties begin by performing k 1-out-of-2 OTs on random k-bit strings with their roles reversed. The receiver acts as sender in the base OTs, with input pairs of strings (k_i^0, k_i^1). The sender, acting as receiver, inputs a random choice bit s_i to the i-th OT and learns $k_i^{s_i}$, for $i = 1, \ldots, k$. The receiver then sends over the values

$$u_i = G(k_i^0) \oplus G(k_i^1) \oplus x$$

where $G : \{0,1\}^k \to \{0,1\}^m$ is a pseudorandom generator (PRG) and $x = (x_1, \ldots, x_m)$ are the receiver's m choice bits. After this step, the parties can obtain k *correlated* OTs on pairs of m-bit strings of the form $(t_i, t_i \oplus x)$, where $t_i = G(k_i^0)$: the receiver knows both t_i and $t_i \oplus x$, while the sender can define

$$\begin{aligned} q_i &= G(k_i^{s_i}) \oplus s_i \cdot u_i \\ &= t_i \oplus s_i \cdot x \\ &= \begin{cases} t_i & \text{if } s_i = 0 \\ t_i \oplus x & \text{if } s_i = 1 \end{cases} \end{aligned}$$

This is a 1-out-of-2 OT on m-bit strings because the other message, $t_i \oplus \overline{s_i} \cdot x$, is computationally hidden to the sender due to the use of a PRG.

Both parties then place these values into matrices $Q, T \in \{0,1\}^{k \times m}$ containing q_i and t_i (respectively) as rows, where the sender holds Q and the receiver holds T. If $q^j, t^j \in \{0,1\}^k$ are the *columns* of Q and T, and $s = (s_1, \ldots, s_k)$, then notice that we have

$$t^j = q^j \oplus (x_j \cdot s)$$

So, by transposing the matrix of OTs we obtain m sets of correlated OTs on k-bit strings, with x_j as the receiver's choice bits. Finally, the two parties can convert these correlated OTs into OTs on random strings using a hash function H that satisfies a notion of correlation robustness (or, modeled as a random oracle): the sender computes the two strings $H(q^j)$ and $H(q^j \oplus s)$, whilst the receiver can only compute one of these with $H(t^j)$; the other string remains unknown to the receiver since it does not know s. This means the parties have converted k initial OTs into $m = \mathsf{poly}(k)$ OTs on random strings, and these random OTs can be used to transfer the sender's chosen messages by encrypting them with a one-time pad.

Apart from the initial base OTs, the only interaction in this process is sending the u_i values at the beginning, which costs $O(mk)$ bits of communication. This gives an overhead of $O(k)$ when the sender's inputs are bit strings of constant size.

Using a Key-Homomorphic PRF. We observe that if the PRG, G, in the above protocol satisfies $G(x \oplus y) = G(x) \oplus G(y)$, and the *base OTs* are correlated so that $k_i^1 = k_i^0 \oplus r$ for some fixed string r, then the main step of interaction in IKNP can be removed. The homomorphic property of the PRG preserves the correlation, so the parties can obtain the OTs $(t_i, t_i \oplus x)$ (for random choice bits x) without any message from the receiver: the sender simply defines $q_i = G(k_i^{s_i})$ while the receiver defines $t_i = G(k_i^0)$ and $x = G(r)$. We then have

$$q_i = G(k_i^{s_i}) = G(k_i^0 \oplus (s_i \cdot r)) = G(k_i^0) \oplus (s_i \cdot G(r)) = t_i \oplus s_i \cdot x,$$

as previously.

Unfortunately, such XOR-homomorphic PRGs are not known to exist. Instead, we do know how to build *almost*-seed-homomorphic PRGs (and almost-key-homomorphic PRFs) $G : \mathbb{Z}_q^n \to \mathbb{Z}_p$, which satisfy

$$G(x + y) = G(x) + G(y) + e \pmod p,$$

where $q > p$ and $|e| \leq 1$ is an error term, based on the *learning with rounding* (LWR) or *learning with errors* (LWE) assumption [BLMR13,BP14]. We remark that it is possible to build an *error-free* key-homomorphic PRF in the random oracle model based on the decisional Diffie-Hellman assumption, with the simple construction $F(k, x) = H(x)^k$ [BLMR13]. However, here the output homomorphism is multiplicative instead of additive, which is more difficult to exploit in constructing OT extension.

Trying to apply these additively homomorphic PRGs (or PRFs) to the IKNP protocol brings about two main challenges. Firstly, since the homomorphism maps into $\mathbb{Z}_p$ and not $\mathbb{F}_2^k$, we obtain matrices Q and T containing $\mathbb{Z}_p$ elements instead of bits, which means there is no natural way of 'transposing' the OT matrix whilst preserving the correlated OT property. Secondly, the homomorphism error means that all of the OTs will be incorrect with high probability.

To handle the first problem, we choose p to be a *primorial modulus* which is the product of ℓ primes, and then decompose the correlated OTs via the Chinese Remainder Theorem.[2] This gives us an alternative to transposing the bit-matrix in IKNP; however, it means that instead of constructing 1-out-of-2 OT, we end up with 1-out-of-p_i random OTs, for each prime factor p_i in the modulus. The second problem of eliminating the error is more difficult. We observe that the receiver can always *compute* a homomorphism error e, such that the resulting error in the OT is given by $e' = e \cdot s_i$, where s_i is one of the sender's choice bits in the base OTs. It seems tempting to let the receiver just send over e so that the sender can correct the error, but this may not be secure: each error leaks information about the unknown PRG key, and a large number of errors could leak information on the secret PRG outputs. To securely correct the error, the receiver instead samples some uniform noise u, which is used to mask e.

[2] Ball, Malkin and Rosulek used a primorial modulus for a different application of constructing arithmetic garbled circuits [BMR16].

To ensure that both parties still obtain the correct result, we must *obliviously* transfer the masked error $u + s_i \cdot e'$ to the sender. Since s_i is a choice bit in the original base OTs, this can be done by without any additional OTs, by extending the base OTs once more with a (standard) PRG.

This last error-correction step introduces some interaction into the basic OT extension protocol, which is otherwise non-interactive. Importantly, the amount of data that needs sending only depends on the security parameter, and not the modulus. Since each distinct prime factor in the modulus produces one additional OT in the OT extension phase, choosing a sufficiently large modulus allows us to obtain an amortized communication cost of $o(1)$ bits per random OT. If we wish to construct 1-out-of-2 OTs, each random 1-out-of p_i OTs can be converted to a single 1-out-of-2 OT with $\log p_i = O(\log k)$ bits of communication (see Appendix A).

Active security. To obtain active security, the above protocol needs to be modified in two ways. Firstly, we need to ensure that the correlation in the base OTs is created correctly, and secondly, we need to ensure that a malicious receiver does not cheat in the error-correction stage, which would cause incorrect outputs.

We first consider the error-correction step. A common technique for dealing with this is to compute random linear combinations of all the correlated OTs, then open the result [KOS15] and check correctness. However, this only achieves negligible cheating probability when the correlation is over a *large field*. In our case we use a ring with many zero divisors, and this method cannot be applied in general. Nevertheless, our situation is slightly different because the *size* of the adversarial deviations can be bounded by some value B that is much smaller than the modulus. This means that for some error $d < B$ introduced by a cheating receiver, and random challenge $r \leftarrow \mathbb{Z}_p$, the product $dr \bmod p$ is *statistically close* to uniform in $\mathbb{Z}_p$ (for arbitrary p), which suffices to prove security when taking random linear combinations.

For the base OTs, the receiver can cheat in an arbitrary way when creating the correlations, which means the above check is not enough to prevent deviations here. It might be tempting to work around this problem by choosing a *prime* modulus q for the base OTs (before applying the PRG/PRF to convert mod p). The problem here is that we then wouldn't be able to *transpose* the base OTs, which is necessary for checking consistency via random linear combinations. Instead, we adopt a different approach used for OT extension in [ALSZ15], where the receiver sends hashes of every pair of base OTs, which are then checked for consistency by the sender. We show that this approach still works to prove that the OTs are correlated over an *arbitrary* abelian group, instead of just XOR correlations over bit strings. If the receiver cheats, they may guess a few bits of the sender's secret choice bits. This does not cause a problem for the OT extension phase, and we model this possibility in our setup functionality.

Instantiation based on DDH. It is possible to modify the above protocol to use a key-homomorphic PRF in the random oracle model based on the decisional Diffie-Hellman assumption, instead of using LWE or LWR. This has the advantage of avoiding the problems with homomorphism error, since the DDH-based PRF $F(k, x) = H(x)^k$ is noiseless. However, the drawback is that this protocol produces random 1-out-of-p OTs, where p is the order of a group in which DDH is hard. Since DDH is not hard if p has small factors, these cannot be decomposed into smaller random OTs using the CRT, so this does not lead to any improvements to 1-out-of-2 OT extension. Nevertheless, random 1-out-of-p OT for exponentially large p (sometimes referred to as batch related-key oblivious PRF evaluation) can be used to construct private equality test and private set intersection protocols [PSZ18,KKRT16,OOS17], so this variation could be useful in these applications to reduce interaction at the cost of requiring exponentiations instead of only symmetric-key operations. More details on this protocol are given in the full version of this work.

2 Preliminaries

2.1 Universally Composable Security

We present ideal functionalities and security proofs in the universal composability framework [Can01], and assume some familiarity with this.

Informally speaking, for a protocol Π which implements a functionality $\mathcal{F}$ in the $\mathcal{G}$-hybrid model, we let $\mathsf{HYBRID}^{\mathcal{G}}_{\Pi,\mathcal{A},\mathcal{Z}}$ denote the view of an environment $\mathcal{Z}$ in an execution of the real protocol with the adversary $\mathcal{A}$ controlling the corrupted parties, in a hybrid model where all parties have access to the ideal functionality $\mathcal{G}$. We let $\mathsf{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ denote the view of $\mathcal{Z}$ in the ideal execution, where the simulator $\mathcal{S}$ plays the role of the corrupted parties in Π and interacts with the functionality $\mathcal{F}$. When the context is clear, we sometimes abbreviate the two executions by HYBRID and IDEAL.

We say that the protocol Π securely realises the functionality $\mathcal{F}$ in the $\mathcal{G}$-hybrid model, if for every adversary $\mathcal{A}$ there exists a simulator $\mathcal{S}$, such that for every environment $\mathcal{Z}$,

$$\mathsf{HYBRID}^{\mathcal{G}}_{\Pi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \mathsf{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$$

where $\overset{c}{\approx}$ is the standard notion of computational indistinguishability.

As well as the standard, computational security parameter k, we often use a statistical security parameter λ. This means that the advantage of any probabilistic $\mathsf{poly}(k)$-time environment in distinguishing the two executions is at most $\mathsf{negl}(k) + O(2^{-\lambda})$.

2.2 Key-Homomorphic Pseudorandom Functions

We now recall the definitions of additively key-homomorphic pseudorandom functions [BLMR13,BP14], and discuss the distribution of the homomorphism error in LWE-based constructions. Let n, p and $q > p$ be integers.

Definition 1 (Key-homomorphic PRF). *A function $F : \mathbb{Z}_q^n \times \{0,1\}^\ell \to \mathbb{Z}_p$ is a* key-homomorphic PRF *if it is a PRF, and for all $k_1, k_2 \in \mathbb{Z}_q$ and $x \in \{0,1\}^\ell$ it holds that:*

$$F(k_1 + k_2, x) = F(k_1, x) + F(k_2, x) \in \mathbb{Z}_p$$

We do not know of any PRFs satisfying the above property, where the homomorphism is additive over both the inputs and the outputs, so instead use the following, weaker definition.

Definition 2 (Almost key-homomorphic PRF). *A function $F : \mathbb{Z}_q^n \times \{0,1\}^\ell \to \mathbb{Z}_p$ is an* almost key-homomorphic PRF *if it is a PRF, and for all $k_1, k_2 \in \mathbb{Z}_q$ and $x \in \{0,1\}^\ell$ it holds that:*

$$F(k_1 + k_2, x) = F(k_1, x) + F(k_2, x) + e \in \mathbb{Z}_p$$

where $|e| \leq 1$.

To realise this, we use the *rounding* function

$$\lfloor x \rceil_p = \lfloor x \cdot (p/q) \rceil$$

which scales $x \in \mathbb{Z}_q$ to lie in the interval $[0, p)$ and then rounds to the nearest integer. We now define the learning with rounding assumption [BPR12].

Definition 3 (Learning With Rounding). *Let $n \geq 1$ and $q \geq p \geq 2$ be integers. For a vector $\boldsymbol{s} \in \mathbb{Z}_q^n$, define the distribution $\mathsf{LWR}_{\boldsymbol{s}}$ to be the distribution over $\mathbb{Z}_q^n, \mathbb{Z}_p$ obtained by sampling $\boldsymbol{a} \leftarrow \mathbb{Z}_q^n$ uniformly at random, and outputting $(\boldsymbol{a}, b = \lfloor \langle \boldsymbol{a}, \boldsymbol{s} \rangle \rceil_p)$.*

The decisional-$\mathsf{LWR}_{n,q,p}$ problem is to distinguish any desired number of samples $(\boldsymbol{a}_i, b_i) \leftarrow \mathsf{LWR}_{\boldsymbol{s}}$ from the same number of samples taken from the uniform distribution on $(\mathbb{Z}_q^n, \mathbb{Z}_p)$, where the secret $\boldsymbol{s}$ is uniform in $\mathbb{Z}_q^n$.

With this we can easily construct a key-homomorphic PRF in the random oracle model, with the function $F : \mathbb{Z}_q^n \times \{0,1\}^\ell \to \mathbb{Z}_p$ defined by

$$F(k, x) = \lfloor \langle k, H(x) \rangle \rceil_p$$

where $H : \{0,1\}^\ell \to \mathbb{Z}_q^n$ is a random oracle. This is an almost-key-homomorphic PRF under the $\mathsf{LWR}_{n,q,p}$ assumption [BPR12].

There are also constructions in the standard model based on learning with rounding or learning with errors [BLMR13,BP14]. All these constructions inherit the same error term, which comes from first computing a function that is linear in the key k, and then rounding the result.

Can we Remove the Homomorphism Error? Applications such as distributed PRFs can work around the error term by suitably rounding the output [BLMR13]. However, in some applications, particularly those in this work, it would be very useful to have a *noise-free* PRF satisfying Definition 1. Two previous works [BV15,BFP+15] claimed that their LWE-based constructions can achieve a slightly weaker notion, where it is *computationally hard* for an adversary to come up with a query x that violates key-homomorphism. Unfortunately, these claims are not correct,[3] and it seems difficult to modify these PRFs to satisfy this.

To see why the error seems to be inherent in these PRFs, consider an experiment where we sample random values $r_1, r_2 \leftarrow \mathbb{Z}_q$, and then test whether $\lfloor r_1 + r_2 \rceil_p = \lfloor r_1 \rceil_p + \lfloor r_2 \rceil_p$. This gives us the same result as testing for homomorphism error in F, since H is a random oracle and the two keys are random.[4] Let $x_1 = \lfloor r_1 \rceil_p, x_2 = \lfloor r_2 \rceil_p$ and define the relevant fractional components $e_1 = r_1 \cdot p/q \pmod 1$, $e_2 = r_2 \cdot p/q \pmod 1$, where the reduction modulo 1 is mapped to the interval $[-1/2, 1/2)$. If p divides q then it holds that e_1, e_2 are uniformly random in the set $[-1/2, 1/2) \cap (p/q)\mathbb{Z}$.

If there is no homomorphism error then we have

$$\lfloor e_1 + e_2 \rceil = \lfloor e_1 \rceil + \lfloor e_2 \rceil$$

Clearly when $e_1 \geq 0$ there will be no error as long as $e_2 \leq 0$. Similarly, when $e_2 \geq 0$ and $e_1 \leq 0$ there is no error. These two error-free possibilities cover approximately half of the space of possible choices of $(e_1, e_2) \in [-1/2, 1/2)^2$. For the remaining cases, if we condition on $e_1 \geq 0$ and $e_2 > 0$ then there will be an error whenever $e_1 + e_2 \geq 1/2$, which happens with probability around $1/2$. Symmetrically, in the remaining case of $e_2 \geq 0$ and $e_1 > 0$ the error probability is around $1/2$, and combining these cases we get an overall error probability of approximately $1/4$. The exact error rate depends on whether p divides q and if q/p is even or not, but is nevertheless always close to $1/4$.

3 OT Extension Protocol

We now describe our main protocol for extending oblivious transfer.

3.1 Setup Functionality

We use the setup functionality $\mathcal{F}_{\Delta\text{-ROT}}$, shown in Fig. 1, to implement the base OTs in our main protocol. This functionality creates k *random, correlated OTs* over an abelian group $\mathbb{G}$ (in our protocol we instantiate this with $\mathbb{G} = \mathbb{Z}_q$) where

[3] We have confirmed this through personal communication with an author of [BFP+15]. This does not affect the main results of that work or [BV15].

[4] This method also applies to known standard model KH-PRFs based on LWE, as these constructions all have the form $F(k, x) = \lfloor L_x(k) \rceil_p$ for some linear function $L_x : \mathbb{Z}_q^n \to \mathbb{Z}_q$.

the sender inputs $\Delta \in \mathbb{G}^n$ and obtains messages of the form $(b_i, b_i + \Delta)$ for randomly sampled $b_i \in \mathbb{G}^n$. The receiver inputs the choice bits $s_i \in \{0,1\}$ in a setup phase, and during the correlated OT phase it learns a_i, which is either b_i or $b_i + \Delta$, depending on s_i. This Δ-OT stage of the functionality also allows a corrupt sender to attempt to guess (a subset of) the bits s_i, but if the guess fails then the functionality aborts. This leakage is necessary so that we can efficiently implement $\mathcal{F}_{\Delta\text{-ROT}}$, using the protocol we give in Section 4.

$\mathcal{F}_{\Delta\text{-ROT}}$ also includes a **Chosen OTs** command, which further extends the base OTs on chosen (but not necessarily correlated) inputs from the sender, using the same choice bits from the receiver.

Functionality $\mathcal{F}^{n,k,\mathbb{G}}_{\Delta\text{-ROT}}$

Parameters: n, the length; k, the number of OTs; $\mathbb{G}$, a finite, additive abelian group.

Initialize: On receiving $(sid, \texttt{init}, s_1, \ldots, s_k)$ from P_A, where $s_i \in \{0,1\}$, and $(sid, \texttt{init})$ from P_B, store $\{s_i\}_i$ and ignore any subsequent $(sid, \texttt{init})$ messages.

Δ-OTs: On input $(sid, \texttt{correlated}, \Delta)$ from P_B, where $\Delta \in \mathbb{G}^n$, send $(sid, \texttt{correlated})$ to P_A and do the following.

1. Sample $b_i \leftarrow \mathbb{G}^n$, for $i \in [k]$. If P_B is corrupted, instead receive b_i from $\mathcal{A}$.
2. Compute $a_i = b_i + s_i \cdot \Delta$
3. If P_A is corrupted, receive $a_i \in \mathbb{G}^n$ from $\mathcal{A}$ and recompute $b_i = a_i - s_i \cdot \Delta$.
4. If P_B is corrupted, wait for $\mathcal{A}$ to input a message $(\texttt{guess}, S)$, where S efficiently describes a subset of $\{0,1\}^k$. If $(s_1, \ldots, s_k) \in S$ then send $(\texttt{success})$ to $\mathcal{A}$. Otherwise, send $\perp$ to both parties and terminate.
5. Output a_i to P_A and b_i to P_B.

Chosen OTs: On input $(sid, \texttt{chosen}, B, (u_i^0, u_i^1)_{i=1}^k)$ from P_B, and $(sid, \texttt{chosen}, B)$ from P_A, where $u_i^0, u_i^1 \in [0, B-1]$, send $(sid, \texttt{chosen}, (u_i^{s_i})_{i=1}^k)$ to P_A.

Fig. 1. Extended correlated random oblivious transfer functionality over a group $\mathbb{G}$

3.2 Random OT Protocol

The functionality we implement is shown in Fig. 2. This produces a batch of $m \cdot \ell$ random OTs at once, consisting of m sets of random 1-out-of-p_i OTs, for each $i = 1, \ldots, \ell$, where p_i is the i-th prime and ℓ is a parameter of the protocol. Let $P_\ell = 2 \cdot 3 \cdot 5 \cdots p_\ell$ be the product of the first ℓ primes.

The protocol, shown in Fig. 3, starts with a setup phase where the parties perform k correlated OTs using $\mathcal{F}_{\Delta\text{-ROT}}$ over $\mathbb{Z}_q^n$, where n is the key length of the almost-key-homomorphic PRF $F : \mathbb{Z}_q^n \times \{0,1\}^k \to \mathbb{Z}_{P_\ell}$. After this phase, P_R holds random values $\boldsymbol{r}, \boldsymbol{b}_i \in \mathbb{Z}_q^n$, whilst P_S holds random $s_i \in \{0,1\}$ and $\boldsymbol{a}_i = \boldsymbol{b}_i + s_i \cdot \boldsymbol{r}$, for $i = 1, \ldots, k$.

Functionality $\mathcal{F}^{m,\ell,k}_{p_i\text{-ROT}}$

After receiving a message (`send`) from P_S and (`receive`) from P_R, ignore all other messages from both parties and do as follows:

1. Sample choices $c^i_1, \ldots, c^i_m \leftarrow \mathbb{Z}_{p_i}$, for $i \in [\ell]$. If P_R is corrupted, instead receive $c^i_j \in \mathbb{Z}_{p_i}$ from $\mathcal{A}$.
2. For each $j \in [m]$, sample $\sum_i p_i$ sets of strings $(y^{i,0}_j, \ldots, y^{i,p_i-1}_j)^{\ell}_{i=1}$, where each $y^{i,c}_j \leftarrow \{0,1\}^k$.
3. Output $(y^{i,0}_j, \ldots, y^{i,p_i-1}_j)$ to P_S and $(c^i_j, y^{i,c^j_i}_j)$ to P_R, for $i \in [\ell]$ and $j \in [m]$.

Fig. 2. Functionality for m sets of random $\{$1-out-of-$p_i\}^{\ell}_{i=1}$ OTs on k-bit strings

In the OT extension phase, the parties expand the base OTs using F, such that the key-homomorphic property of F preserves the correlation between $\boldsymbol{a}$ and $\boldsymbol{b}$, except for a small additive error. We have:

$$F(\boldsymbol{a}_i, j) = F(\boldsymbol{b}_i + s_i \cdot \boldsymbol{r}, j) = F(\boldsymbol{b}_i, j) + s_i \cdot (F(\boldsymbol{r}, j) + e_i)$$

where $|e_i| \leq 1$ (note that e_i depends on j, but we often omit the subscript-j to simplify notation).

Since P_R knows both $\boldsymbol{b}_i$ and $\boldsymbol{r}$, it can compute e_i, and then use the base OTs to obliviously transfer either $u_i + e_i$ (if $s_i = 1$) or u_i (if $s_i = 1$) to P_S, where u_i is a uniformly random value in $\{0, \ldots, B-1\}$, and B is superpolynomial in the security parameter so that u_i statistically masks e_i.

After step 2f, if $u_i + e_i \notin \{-1, B\}$ then we have:

$$\begin{aligned} q_i = q'_i - v_i &= t'_i + s_i \cdot (x_j + e_i) - u_i - s_i \cdot e_i \\ &= t'_i - u_i + s_i \cdot x_j \\ &= t_i + s_i \cdot x_j \pmod{P_\ell} \end{aligned}$$

For each $j \in [m+1]$, these k sets of correlated OTs are then placed into vectors $\boldsymbol{q}_j$ and $\boldsymbol{t}_j$, which satisfy $\boldsymbol{q}_j = \boldsymbol{t}_j + x_j \cdot \boldsymbol{s}$. To convert these into random 1-out-of-p_i OTs, for $i = 1, \ldots, \ell$, each $\boldsymbol{t}_j$ is reduced modulo p_i and then hashed with the random oracle to produce the receiver's output string. The c-th output of the sender, for $c \in \{0, \ldots, p_i - 1\}$, in the (i,j)-th OT is defined as:

$$H(\boldsymbol{q}_j - c \cdot \boldsymbol{s} \bmod p_i) = H(\boldsymbol{t}_j + (x_j - c) \cdot \boldsymbol{s} \bmod p_i)$$

which for $c = x_j$ is equal to the receiver's output, as required. The sender's other outputs are computationally hidden to the receiver, since to compute them it would have to query $\boldsymbol{q}_j - x' \cdot \boldsymbol{s} \mod p_i$ to the random oracle for some $x' \neq x_j$, but this requires guessing $\boldsymbol{s}$.

The only opportunity for a malicious receiver to misbehave in the protocol is when sending the $(u_i, u_i + e_i)$ values to the base OT functionality, to correct the errors. The consistency check phase prevents this, by opening a random linear combination of the correlated OTs and checking that the linear relation still

Protocol $\mathit{\Pi}^{m,\ell,k}_{p_i\text{-ROT}}$

Let $F : \mathbb{Z}_q^n \times \{0,1\}^k \to \mathbb{Z}_{P_\ell}$ be an almost-key-homomorphic PRF.
Let $H : \{0,1\}^* \to \{0,1\}^k$ be a random oracle and $G : \{0,1\}^k \to \mathbb{Z}_{P_\ell}^m$ be a PRG.

1. *Setup phase.*
   (a) P_S samples $s_1, \ldots, s_k \leftarrow \{0,1\}$, and P_R samples $\boldsymbol{r} \leftarrow \mathbb{Z}_q^n$
   (b) Both parties initialize $\mathcal{F}^{n,k,\mathbb{Z}_q}_{\Delta\text{-ROT}}$, where P_S inputs $(s_1, \ldots, s_k)$. P_R then calls the functionality again with input $(\texttt{correlated}, \boldsymbol{r})$.
   (c) P_R receives $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k$ and P_S receives $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k$ such that $\boldsymbol{a}_i = \boldsymbol{b}_i + s_i \cdot \boldsymbol{r}$.
2. *Extension phase.*[a] P_S initializes a zero matrix $\mathbf{Q} \in \mathbb{Z}_{P_\ell}^{k\times(m+1)}$. P_R initializes a zero matrix $\mathbf{T} \in \mathbb{Z}_{P_\ell}^{k\times(m+1)}$, and a vector $\boldsymbol{x} \in \mathbb{Z}_{P_\ell}^{m+1}$.

   For each $j \in [m+1]$:
   (a) P_S computes $q_i' = F(\boldsymbol{a}_i, j) \in \mathbb{Z}_{P_\ell}$, for $i \in [k]$
   (b) P_R computes $t_i' = F(\boldsymbol{b}_i, j), x_j = F(\boldsymbol{r}, j) \in \mathbb{Z}_{P_\ell}$
   (c) P_R computes the errors
   $$e_i = F(\boldsymbol{b}_i + \boldsymbol{r}, j) - t_i' - x_j \in \{0,1\}, \quad \text{for } i \in [k]$$
   and samples $u_i \leftarrow [0, B-1]$
   (d) P_R calls $\mathcal{F}^{n,k,\mathbb{Z}_q}_{\Delta\text{-ROT}}$ on input $(\texttt{chosen}, B, (u_i, u_i + e_i \bmod B))$
   (e) P_S receives $v_i = u_i + s_i \cdot e_i \bmod B$ from $\mathcal{F}^{n,k,\mathbb{Z}_q}_{\Delta\text{-ROT}}$
   (f) P_S defines $q_i = q_i' - v_i$, and P_R defines $t_i = t_i' - u_i$
   It should now hold that
   $$q_i = t_i + s_i \cdot x_j \mod P_\ell$$
   (g) P_S stores $\boldsymbol{q}_j = (q_1, \ldots, q_k)$ in column j of the matrix $\mathbf{Q}$.
   (h) P_R stores $\boldsymbol{t}_j = (t_1, \ldots, t_k)$ in column j of $\mathbf{T}$, and x_j in entry j of the vector $\boldsymbol{x}$.
3. *Consistency check.*
   (a) P_S samples a seed $\sigma \leftarrow \{0,1\}^k$ and sends this to P_R
   (b) Both parties compute $\boldsymbol{\alpha} = (G(\sigma)\|1)$.
   (c) P_R computes and sends
   $$\tilde{\boldsymbol{t}} = \mathbf{T}\boldsymbol{\alpha}, \quad \tilde{x} = \boldsymbol{x}^\top \boldsymbol{\alpha}$$
   (d) P_S checks that $\mathbf{Q}\boldsymbol{\alpha} = \tilde{\boldsymbol{t}} + \tilde{x}\boldsymbol{s}$
4. *Output:* P_S samples a seed $\rho \leftarrow \{0,1\}^k$ and sends this to P_R. The parties then compute their outputs as follows:
   – P_S outputs, for $j \in [m]$ and $i \in [\ell]$:
   $$H(i, \rho, \boldsymbol{q}_j^i), H(i, \rho, \boldsymbol{q}_j^i - \boldsymbol{s}), \ldots, H(i, \rho, \boldsymbol{q}_j^i - (p_i - 1)\cdot \boldsymbol{s}), \quad \text{where } \boldsymbol{q}_j^i = \boldsymbol{q}_j \bmod p_i$$
   – P_R outputs, for $j \in [m]$ and $i \in [\ell]$:
   $$x_j^i, H(i, \rho, \boldsymbol{t}_j^i), \qquad \text{where } x_j^i = x_j \bmod p_i, \quad \boldsymbol{t}_j^i = \boldsymbol{t}_j \bmod p_i$$

[a] Steps 2–4 can be iterated by maintaining j as a counter.

Fig. 3. Random 1-out-of-p_i OT extension protocol

holds. We must then discard the $(m+1)$-th set of OTs so that the opened values do not reveal anything about the receiver's outputs. This check allows a corrupt receiver to attempt to guess a few of the s_i bits by cheating in only a few OT instances. However, this is exactly the same behaviour that is already allowed by the $\mathcal{F}_{\Delta\text{-ROT}}$ functionality for the base OTs. It does not pose a problem for security because in the output phase $\boldsymbol{s}$ is put through a random oracle, and the whole of $\boldsymbol{s}$ must be guessed to break security.

3.3 Security

Theorem 1. *Let $B = \Theta(2^\lambda)$, $\ell = \Omega(k\lambda)$, F be an almost key-homomorphic PRF and H be a random oracle. Then protocol $\Pi^{m,\ell,k}_{p_i\text{-ROT}}$ securely realises the functionality $\mathcal{F}^{m,\ell,k}_{p_i\text{-ROT}}$ with static security in the $\mathcal{F}^{n,k,\mathbb{Z}_q}_{\Delta\text{-ROT}}$-hybrid model.*

We prove this by considering separately the cases of a corrupt sender and a corrupt receiver. Security when both parties are honest, or both corrupt, is straightforward.

Corrupt sender. This is the simpler of the two cases. We construct an ideal-world simulator, $\mathcal{S}_S$, shown in Fig. 4. The simulator uses random values to simulate the v_i messages sent to P_S during the OT extension phase, then samples $\tilde{x}$ at random to respond to the consistency check, computing $\tilde{\boldsymbol{t}}$ such that the check will always pass. The random oracle queries are responded to using knowledge of the sender's bits $\boldsymbol{s}$ from the setup phase, so as to be consistent with the random sender messages obtained from $\mathcal{F}_{p_i\text{-ROT}}$. All other queries are responded honestly, at random. The security of the protocol against a corrupt sender rests on two key points: (1) $B = \Theta(2^\lambda)$, so that $u_i + e_i$ statistically masks the errors e_i in the protocol, and (2) The security of the key-homomorphic PRF, which implies the x_j outputs of the honest receiver are pseudorandom, and also the simulated $\tilde{x}$ is indistinguishable from the real value in the protocol.

Lemma 1. *For every adversary $\mathcal{A}$ who corrupts P_S, and for every environment $\mathcal{Z}$, it holds that*

$$\mathsf{IDEAL}_{\mathcal{F}_{p_i\text{-ROT}},\mathcal{S}_S,\mathcal{Z}} \overset{c}{\approx} \mathsf{HYBRID}^{\mathcal{F}_{\Delta\text{-ROT}}}_{\Pi_{p_i\text{-ROT}},\mathcal{A},\mathcal{Z}}$$

Proof. Recall that as well as seeing the view of $\mathcal{A}$ during the protocol execution, the environment $\mathcal{Z}$ learns the outputs of both parties. We prove security by defining a sequence of hybrid executions, where H_0 is defined to be the ideal process and each successive hybrid modifies the previous execution in some way.

Hybrid H_1: Instead of sampling v_i at random, $\mathcal{S}_S$ sends $v_i = u_i + s_i \cdot e_i \bmod B$, where $u_i \leftarrow [0, B-1]$ and e_i is computed as in the protocol, using randomly sampled $\boldsymbol{r} \in \mathbb{Z}_q^n$ and $\boldsymbol{b}_i := \boldsymbol{a}_i - s_i \cdot \boldsymbol{r}$.

Simulator $\mathcal{S}_S$

1. *Setup phase.* $\mathcal{S}_S$ receives $\boldsymbol{s} = (s_1, \ldots, s_k) \in \{0,1\}^k$ and $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k \in \mathbb{Z}_q^n$ from the corrupt sender, as input to $\mathcal{F}_{\Delta\text{-ROT}}^{n,k,\mathbb{Z}_q}$.
2. *Extension phase.*
   (a) For each $j \in [m{+}1]$ and $i \in [k]$, whenever $\mathcal{S}_S$ receives a message $(\texttt{chosen}, B)$ from $\mathcal{A}$ (as input to $\mathcal{F}_{\Delta\text{-ROT}}^{n,k,\mathbb{Z}_q}$), sample a random value $v_{i,j} \in [0, B-1]$ and return this to $\mathcal{A}$.
   (b) Compute the matrix $\mathbf{Q}$ as P_S should in the protocol, using knowledge of $\boldsymbol{a}_i$ and $v_{i,j}$.
   (c) Call the functionality $\mathcal{F}_{p_i\text{-ROT}}$ to obtain the sender's outputs $\boldsymbol{y}_j^i = (y_j^i[1], \ldots, y_j^i[p_i])$, for $i \in [\ell]$ and $j \in [m]$.
3. *Consistency check.* After receiving the seed σ from $\mathcal{A}$, compute $\boldsymbol{\alpha} = (G(\sigma)\|1)$, then send $\tilde{x} \leftarrow \mathbb{Z}_{P_\ell}$ and $\tilde{\boldsymbol{t}} = \mathbf{Q}\boldsymbol{\alpha} - \hat{x}\boldsymbol{s}$ to $\mathcal{A}$.
4. *Output/Random Oracle Queries.* Receive the seed ρ from $\mathcal{A}$, and whenever a query of the form $(i, \rho, \boldsymbol{q}_j^i - c\cdot\boldsymbol{s})$ is sent to the random oracle H, for $c \in [0, p_i-1]$, respond with $y_j^i[c]$.

Fig. 4. Simulator for a corrupted sender

Hybrid H_2: Instead of sampling $\tilde{x}$ at random, let $x_1, \ldots, x_m$ be the outputs of the honest receiver (from $\mathcal{F}_{p_i\text{-ROT}}$), and sample $x_{m+1} \leftarrow \mathbb{Z}_{P_\ell}$. $\mathcal{S}_S$ then sends $\tilde{x} = \boldsymbol{x}^\top \boldsymbol{\alpha}$.

Hybrid H_3: This is defined the same as H_2, except the random choices $x_1, \ldots, x_{m+1}$ are replaced with values computed from the key-homomorphic PRF as $x_j = F(\boldsymbol{r}, j)$, using the previously sampled $\boldsymbol{r}$.

Note that all of the simulated messages in the final hybrid, H_3, are identically distributed to messages sent in the real execution, and the outputs of the sender and receiver are computed exactly as in the protocol as outputs of the random oracle H and PRF F, respectively. Therefore, $\mathsf{H}_3 \equiv \mathsf{HYBRID}$.

Hybrids H_0 and H_1 are identically distributed as long as $u_i + s_i \cdot e_i \notin \{-1, B\}$, since $e_i \in \{0, \pm 1\}$. If the reduction modulo B overflows then $\mathcal{Z}$ could distinguish because the outputs in H_1 will be incorrect. This occurs with probability at most $2/B$ for each i, so when $B = \Omega(2^\lambda)$ we have $\mathsf{H}_0 \overset{s}{\approx} \mathsf{H}_1$.

In hybrid H_2, the value $\tilde{x}$ is masked by x_{m+1} so is uniformly random in the view of $\mathcal{Z}$. Therefore, $\mathsf{H}_1 \equiv \mathsf{H}_2$.

Hybrids H_2 and H_3 are computationally indistinguishable, by a standard reduction to the key-homomorphic PRF, since $\boldsymbol{r}$ is uniformly random and not seen by the environment, so the x_j values are pseudorandom. This completes the claim that $\mathsf{IDEAL} \overset{c}{\approx} \mathsf{HYBRID}$. □

Corrupt receiver. The simulator $\mathcal{S}_R$, given in Fig. 5, essentially runs an internal copy of the sender and honestly generates messages as P_S would. The

Simulator $\mathcal{S}_R$

1. *Setup phase.* $\mathcal{S}_R$ receives $\boldsymbol{r}$ and $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k$ from the corrupt receiver, as input to $\mathcal{F}^{n,k,\mathbb{Z}_q}_{\Delta\text{-ROT}}$. It then samples $\boldsymbol{s} \leftarrow \{0,1\}^k$ and defines $\boldsymbol{a}_i = \boldsymbol{b}_i + s_i \boldsymbol{r}$.
   For any key query guess submitted by $\mathcal{A}$ to $\mathcal{F}^{n,k,\mathbb{Z}_q}_{\Delta\text{-ROT}}$, $\mathcal{S}$ responds according to the secret $\boldsymbol{s}$. If any guess is unsuccessful, $\mathcal{S}$ aborts.
2. *Extension phase.*
   (a) $\mathcal{S}$ waits for P_R to send a message of the form $(\texttt{chosen}, B, u_i, u'_i)$ as input to $\mathcal{F}^{n,k,\mathbb{Z}_q}_{\Delta\text{-ROT}}$. $\mathcal{S}$ then defines $e'_i = u'_i - u_i \bmod B$ and $v_i = u_i + s_i \cdot e'_i$.
   (b) $\mathcal{S}$ computes q_i as an honest sender would, and defines the matrix $\mathbf{Q}$.
3. *Consistency Check.*
   (a) $\mathcal{S}$ samples $\sigma \leftarrow \{0,1\}^k$ and sends this to P_R.
   (b) $\mathcal{S}$ receives $\tilde{\boldsymbol{t}}, \tilde{x}$ and checks that $\mathbf{Q}\boldsymbol{\alpha} = \tilde{\boldsymbol{t}} + \tilde{x}\boldsymbol{s}$. If the check fails, abort.
   (c) Extract the inputs x'_j using Proposition 1.
   (d) Call $\mathcal{F}_{p_i\text{-ROT}}$ and send the choices $\{x'_j \bmod p_i\}_{i \in [\ell]}$, for $j \in [m]$. Receive back the OT outputs y^i_j, for $i \in [\ell], j \in [m]$.
4. *Output phase/Random Oracle Queries.* $\mathcal{S}$ sends a random seed $\rho \leftarrow \{0,1\}^k$, and responds to the random oracle queries as follows:
   (a) If a query is $(i, \rho, (\boldsymbol{q}_j - x'_j \boldsymbol{s}) \bmod p_i)$ for some $i \in [\ell], j \in [m]$ then respond with y^i_j. Otherwise, respond at random (consistent with previous queries).

Fig. 5. Simulator for a corrupted receiver

main challenge is to extract the inputs of the corrupt P_R, and also show that $\mathcal{Z}$ cannot query the random oracle H on a value corresponding to one of the sender's random outputs that was not chosen by the receiver.

We use the following technical lemma to analyse the soundness of the consistency check when taking random linear combinations over the ring $\mathbb{Z}_{P_\ell}$.

Lemma 2. *Let $\mathbf{E} \in \mathbb{Z}_{P_\ell}^{k\times(m+1)}$. Suppose that every column $\boldsymbol{e}_i$ of $\mathbf{E}$ satisfies $\|\boldsymbol{e}_i\|_\infty \le B$, and further that there is at least one column not in* $\mathrm{span}(\mathbf{1})$*. Then,*

$$\Pr_{\boldsymbol{\alpha} \leftarrow \mathbb{Z}^m_{P_\ell} \times \{1\}}[\mathbf{E}\boldsymbol{\alpha} \in \mathrm{span}(\mathbf{1})] \le 2B/P_\ell$$

Proof. From the assumption that at least one column of $\mathbf{E}$ not in $\mathrm{span}(\mathbf{1})$, there exist two rows $\boldsymbol{a}, \boldsymbol{b}$ of $\mathbf{E}$ with $\boldsymbol{a} \neq \boldsymbol{b}$. If $\mathbf{E}\boldsymbol{\alpha} \in \mathrm{span}(\mathbf{1})$ then $\langle \boldsymbol{a}, \boldsymbol{\alpha}\rangle = \langle \boldsymbol{b}, \boldsymbol{\alpha}\rangle$ and so $\langle \boldsymbol{a} - \boldsymbol{b}, \boldsymbol{\alpha}\rangle = 0$. Let $\boldsymbol{d} = \boldsymbol{a} - \boldsymbol{b}$, and j be an index where $d_j \neq 0$. Then $\langle \boldsymbol{d}, \boldsymbol{\alpha}\rangle = 0$ if and only if $d_j\alpha_j = -\sum_{i\neq j} d_i\alpha_i$.

First consider the case that $j \neq m+1$, so α_j is uniform in $\mathbb{Z}_{P_\ell}$. For any fixed choice of d_j, the number of distinct possibilities for $d_j\alpha_j \bmod P_\ell$ is given by the order of d_j in the additive group $\mathbb{Z}_{P_\ell}$, which equals $P_\ell/\gcd(d_j, P_\ell)$. Since $|a_j|, |b_j| \le B$, we have $d_j \in [0, 2B] \cup [P_\ell - 2B, P_\ell - 1]$, from which it follows that $\gcd(d_j, P_\ell) \le 2B$. Therefore, since α_j is random and independent of $\sum_{i\neq j} \alpha_i d_i$, we have that $\Pr[\langle \boldsymbol{d}, \boldsymbol{\alpha}\rangle = 0] \le 2B/P_\ell$.

On the other hand, if $j = m+1$ then $\alpha_j = 1$. But this means $d_j = -\sum_{i\neq j} d_i\alpha_i$, and because $d_j \neq 0$ there must be another index j' with $d_{j'} \neq 0$. We then apply the previous argument on j' to obtain the same probability. □

If $\boldsymbol{\alpha}$ is sampled using a PRG with a public seed, instead of uniformly at random, the previous statement still holds except with negligible probability.

Lemma 3. *Let* $\mathbf{E}$ *be as in Lemma 2 and let* $G : \{0,1\}^k \to \mathbb{Z}_{P_\ell}^m \times \{1\}$ *be a PRG. Then,*

$$\Pr_{\sigma \leftarrow \{0,1\}^k}[\mathbf{E}\boldsymbol{\alpha} \in \mathrm{span}(\mathbf{1}) : \boldsymbol{\alpha} = G(\sigma)] \leq 2B/P_\ell + \mathsf{negl}(k)$$

Proof. Define a distinguisher, D, for the PRG G, which on input a challenge $\boldsymbol{\alpha}$, outputs 1 if $\mathbf{E}\boldsymbol{\alpha} \in \mathrm{span}(\mathbf{1})$ and 0 otherwise. From Lemma 2 we know that $\Pr[D = 1]$ given that $\boldsymbol{\alpha}$ is uniformly random is $\leq 2B/P_\ell$. On the other hand, the advantage of D is $\mathsf{negl}(k)$, so if $\boldsymbol{\alpha}$ is an output of G then it must be that D outputs 1 with probability at most $2B/P_\ell + \mathsf{negl}(k)$. □

We now show indistinguishability of the simulation.

Lemma 4. *For every adversary* $\mathcal{A}$ *who corrupts* P_R*, and for every environment* $\mathcal{Z}$*, it holds that*

$$\mathsf{IDEAL}_{\mathcal{F}_{p_i\text{-}\mathsf{ROT}},\mathcal{S}_R,\mathcal{Z}} \overset{c}{\approx} \mathsf{HYBRID}^{\mathcal{F}_{\Delta\text{-}\mathsf{ROT}}}_{\Pi_{p_i\text{-}\mathsf{ROT}},\mathcal{A},\mathcal{Z}}$$

Proof. We first show how $\mathcal{S}_R$ (Fig. 5) extracts the corrupt receiver's inputs in step 3c. $\mathcal{S}_R$ received the values $u_i, u_i' = u_i + e_i'$ which $\mathcal{A}$ used as input to $\mathcal{F}_{\Delta\text{-}\mathsf{ROT}}$. $\mathcal{S}_R$ can also compute the actual errors e_i (which would equal e_i' if P_R was honest), since it knows $\boldsymbol{r}$ and $\boldsymbol{b}_i$. For each $j \in [m+1]$ and $i \in [k]$, $\mathcal{S}_R$ defines a value (omitting j subscripts) $q_i = q_i' - (u_i + s_i e_i')$, and then puts all these values into the vector $\boldsymbol{q}_j$. We also compute the values x_j and $\boldsymbol{t}_j$ as an honest P_R would do according to the protocol.

Now, if P_R was honest we would have $\boldsymbol{q}_j = \boldsymbol{t}_j + x_j \cdot \boldsymbol{s}$, but it actually holds that

$$\boldsymbol{q}_j = \boldsymbol{t}_j + x_j \cdot \boldsymbol{s} + \boldsymbol{e}_j * \boldsymbol{s}$$

where $\boldsymbol{e}_j$ contains the values $(e_1 - e_1', \ldots, e_k - e_k')$ from iteration j of this phase, and $*$ denotes component-wise product. Note that since $e_i \in \{0, \pm 1\}$ and $e_i' \in \{0, \ldots, B-1\}$ we have $\|\boldsymbol{e}_j\|_\infty \leq B$ for all j.

At this point, although we have computed values x_j which could be used to define the inputs of P_R, these may *not* be the correct inputs $\mathcal{S}_R$ should send to $\mathcal{F}_{p_i\text{-}\mathsf{ROT}}$. This is because $\mathcal{A}$ could choose, for instance, $\boldsymbol{e}_j = \mathbf{1}$, and then the actual inputs would correspond to $x_j + 1$ and not x_j. Proposition 1 shows how we obtain the correct inputs. Let $\mathtt{view}(\mathcal{A})$ denote the view of the corrupt receiver at this point in the execution.

Proposition 1. *Suppose the consistency check passes, and* $\mathcal{A}$ *makes no* ($\mathtt{guess}$) *queries to* $\mathcal{F}_{\Delta\text{-}\mathsf{ROT}}$*. Then with overwhelming probability there exists a set* $S \subset [k]$ *and values* $x_j', \boldsymbol{e}_j'$*, for* $j \in [m]$ *such that:*

1. $\boldsymbol{q}_j = \boldsymbol{t}_j + x_j' \cdot \boldsymbol{s} + \boldsymbol{e}_j' * \boldsymbol{s}$.
2. For all $i \in [k] \setminus S$, $\boldsymbol{e}_j'[i] = 0$

3. $H_\infty((s_i)_{i\in S}|\mathtt{view}(\mathcal{A})) = 0$
4. $H_\infty((s_i)_{i\in[k]\setminus S}|\mathtt{view}(\mathcal{A})) = k - |S|$

Proof. Recall that $\tilde{\boldsymbol{t}}, \tilde{x}$ are the values received by $\mathcal{S}_R$ from P_R during the consistency check, and we have $\boldsymbol{q}_j = \boldsymbol{t}_j + x_j \cdot \boldsymbol{s} + \boldsymbol{e}_j * \boldsymbol{s}$.

This means we can write

$$\mathbf{Q} = \mathbf{T} + \underbrace{(x_1 \cdot \mathbf{1} + \boldsymbol{e}_1 \| \cdots \| x_m \cdot \mathbf{1} + \boldsymbol{e}_m)}_{=\mathbf{Y}} * \boldsymbol{s}$$

where we extend the $*$ operator to apply to every column of $\mathbf{Y}$ in turn.

Define the vectors, in $\mathbb{Z}_{P_\ell}^k$,

$$\boldsymbol{\delta}_x = \mathbf{1}\tilde{x} - \mathbf{Y}\boldsymbol{\alpha}, \quad \boldsymbol{\delta}_t = \mathbf{T}\alpha - \tilde{\boldsymbol{t}}$$

We can think of these as representing the deviation between what P_R sent, and the values P_R *should have* sent, given $\mathbf{Y}, \mathbf{T}$. If the check succeeds, then we know that

$$\mathbf{Q}\boldsymbol{\alpha} = \tilde{\boldsymbol{t}} + (\mathbf{1}\tilde{x}) * \boldsymbol{s}$$

and so

$$\begin{aligned}(\mathbf{T} + \mathbf{Y} * \boldsymbol{s})\boldsymbol{\alpha} &= \tilde{\boldsymbol{t}} + (\mathbf{1}\tilde{x}) * \boldsymbol{s} \\ \Leftrightarrow \boldsymbol{\delta}_t &= (\mathbf{1}\tilde{x}) * \boldsymbol{s} - (\mathbf{Y} * \boldsymbol{s})\boldsymbol{\alpha} \\ &= \boldsymbol{\delta}_x * \boldsymbol{s}\end{aligned}$$

For each index $i \in [k]$, if the check passes then it must hold that either $\boldsymbol{\delta}_t[i] = \boldsymbol{\delta}_x[i] = 0$ — which essentially means there was no deviation at position i — or $\boldsymbol{\delta}_x[i] \neq 0$. In the latter case, because $s_i \in \{0,1\}$, the cheating receiver must guess s_i in order to pass the check.

Define $S \subset [k]$ to be the set of all indices i for which $\boldsymbol{\delta}_x[i] \neq 0$. From the above, we have that the probability of passing the check (over the random choice of $\boldsymbol{s}$) is at most $2^{-|S|}$. If the check passes, this also implies the last two claims of the proposition, that $H_\infty((s_i)_{i\in S}|\mathsf{view}(\mathcal{A})) = 0$, and $H_\infty((s_i)_{i\in[k]\setminus S}|\mathsf{view}(\mathcal{A})) = k - |S|$.

Let $\mathbf{Y}_{-S}$ denote the matrix $\mathbf{Y}$ with its rows corresponding to indices in the set S removed. Note that for any $i \notin S$, we have $\boldsymbol{\delta}_x[i] = 0$ and so $(\mathbf{Y}\boldsymbol{\alpha})_{-S} = (\mathbf{Y}_{-S})\boldsymbol{\alpha} = \mathbf{1}\tilde{x}$, which lies in $\mathrm{span}(\mathbf{1})$. Since column j of $\mathbf{Y}$ is equal to $\mathbf{1}x_j + \boldsymbol{e}_j$, it must also hold that $(\mathbf{E}_{-S})\boldsymbol{\alpha} \in \mathrm{span}(\mathbf{1})$, where $\mathbf{E} = (\boldsymbol{e}_1 \| \cdots \| \boldsymbol{e}_{m+1})$.

Applying Lemma 3 with $\mathbf{E}_{-S}$, it then holds that every column of $\mathbf{E}_{-S}$ is in $\mathrm{span}(\mathbf{1})$ with overwhelming probability, provided $2B/P_\ell$ is negligible. Therefore, for every $j \in [m]$, we can compute x'_j and $\boldsymbol{e}'_j$ such that the j-th column of $\mathbf{Y}$ satisfies $\boldsymbol{y}_j = x'_j \cdot \mathbf{1} + \boldsymbol{e}'_j$, where $(\boldsymbol{e}'_j)_{-S} = 0$. These are values needed to satisfy points 1 and 2. □

The set S in the proposition represents the indices where P_R cheated, corresponding to the set of bits of $\boldsymbol{s}$ which P_R must guess to pass the consistency check. Passing the check also guarantees that the error vectors $\boldsymbol{e}'_j$ can only be non-zero in the positions corresponding to S, which is crucial for the next stage.

After extracting the corrupt receiver's inputs, we need to show that the random oracle calls made by $\mathcal{Z}$ cannot allow it to distinguish. In particular, if $\mathcal{Z}$ queries

$$(i, \rho, (\boldsymbol{q}_j - y_j \boldsymbol{s}) \bmod p_i))$$

for some $y_j \neq x'_j \bmod p_i$ then $\mathcal{Z}$ will be able to distinguish, since the simulator's response will be random, whereas the response in the real world will be one of the sender's OT outputs.

From Proposition 1, we know that if no (`guess`) queries were made to $\mathcal{F}_{\Delta\text{-ROT}}$ then there are exactly $k - |S|$ bits of the secret $\boldsymbol{s}$ that are unknown in the view of $\mathcal{A}$, and these correspond to the index set $[k] \setminus S$.

Now, from the first part of the proposition, we can rewrite a 'bad' query of the form given above as

$$(i, \rho, (\boldsymbol{t}_j + \boldsymbol{e}'_j * \boldsymbol{s} + (x'_j - y_j)\boldsymbol{s}) \bmod p_i)$$

Since $\boldsymbol{t}_j$ and $\boldsymbol{e}'_j * \boldsymbol{s}$ are fixed in the view of $\mathcal{Z}$, it must be the case that coming up with such a query requires knowing all of $\boldsymbol{s}$. This happens with probability at most $(p_i - 1) \cdot 2^{|S|-k}$ per query with index i. Taking into account the probability of passing the consistency check, we get an overall success probability bounded by $Q \cdot (p_\ell - 1) \cdot 2^{-k}$, where Q is the number of random oracle queries, which is negligible. Making key queries to $\mathcal{F}_{\Delta\text{-ROT}}$ cannot help guess $\boldsymbol{s}$ because any incorrect guess causes an abort, so this does not affect the distinguishing probability. □

3.4 Choosing the Parameters

We first show how to securely choose parameters to optimize communication, and then discuss instantiating the key-homomorphic PRF. After the setup phase, and ignoring the short seeds sent in the consistency check, the only communication is to the (`chosen`) command of $\mathcal{F}_{\Delta\text{-ROT}}$, which can be implemented with λ bits of communication when $B = 2^\lambda$ (see Section 4). This gives an overall complexity of $\lambda k m$ bits to generate $m\ell$ random OTs. If $\ell = \omega(\lambda k)$ then we obtain an amortized cost per random OT of $o(1)$, which gets smaller as ℓ increases.

To realise 1-out-of-2 bit-OT on chosen strings, each random 1-out-of-p_i OT must be converted to a 1-out-of-2 OT, at a cost of sending $\log p_i$ bits from the receiver and 2 bits from the sender (see Appendix A). This adds a cost of $T_{m,\ell} = m \sum_{i=1}^{\ell} (\log_2 p_i + 2)$ bits, and from the prime number theorem we have $p_i = O(i \log i)$, so $T_{m,\ell} = m \sum_{i=1}^{\ell} O(\log i) = O(m\ell \log \ell)$, giving an overall, amortized cost of $O(\log \ell) = O(\log k)$ bits per OT when $\ell = \Omega(\lambda k)$.

Instantiating the key-homomorphic PRF. We can instantiate F using the random oracle-based construction from Section 2 based on learning with rounding, or standard model constructions from LWE [BLMR13,BP14]. With LWR, the parameters affecting security are the dimension n and moduli p, q. In our case we fix $p = P_\ell$ and can choose n, q to ensure security. With an exponential modulus, we know that LWR is at least as hard as LWE with the same dimension n and modulus q, where the LWE error distribution is bounded by $\beta = q/(2^\lambda P_\ell)$, and λ is a statistical security parameter [BPR12,AKPW13]. This gives a modulus-to-noise ratio of $q/\beta = O(2^\lambda P_\ell)$. LWE with an exponential modulus-to-noise ratio has previously been used to construct attribute-based encryption [GVW13] and fully key-homomorphic encryption [BGG+14], and is believed to be hard if $q/\beta \le 2^{n^\epsilon}$, for some constant $0 < \epsilon < 1/2$ chosen to resist known attacks based on lattice reduction and BKW. To achieve optimal communication in our protocol, we need $\ell = \omega(\lambda k)$, which from the prime number theorem implies that $\log P_\ell = \omega(\lambda k \log k)$. This gives $\log(q/\beta) = \omega(\lambda^2 k \log k)$, so we can have a dimension of $n = \omega((\lambda^2 k \log k)^{1/\epsilon})$ and ensure security.

4 Actively Secure Base OTs

We now show how to implement the functionality $\mathcal{F}^{n,k,\mathbb{G}}_{\Delta\text{-ROT}}$ (Fig. 1), which creates k correlated base OTs over $\mathbb{G}^n$, for an additive abelian group $\mathbb{G}$. We achieve active security using a modification of the consistency check from the OT extension protocol by Asharov et al. [ALSZ15,ALSZ17a]. Additionally, in Section 4.1 we identify a crucial security flaw in their protocol, whereby a passively corrupted sender can obtain an 'oracle' that allows brute-force guessing of the receiver's choice bits by computing hash values. This bug has since been fixed in a revised version [ALSZ17b], and we apply the same fix to our protocol.

We let $\mathcal{F}^{k,k}_{\mathsf{OT}}$ denote the standard functionality for k sets of 1-out-of-2 OTs on k-bit strings. In the correlated OT phase of our protocol, shown in Fig. 6, the parties first extend the base OTs from $\mathcal{F}_{\mathsf{OT}}$ using a PRF, and the sender P_S (who would be receiver when running the main OT extension protocol) then sends the u_i values which create the correlation over the group $\mathbb{G}$. The consistency check is based on the check in the OT extension protocol by Asharov et al. [ALSZ15], which is used to verify the sender's inputs are bit strings of the form $(b_i, b_i \oplus \Delta)$. We adapt this to ensure they have the form $(b_i, b_i + \Delta)$, where Δ and each b_i are vectors of length n over any finite abelian group $\mathbb{G}$. In our protocol the parties then output the correlated base OTs, instead of transposing and hashing them to perform OT extension as in [ALSZ15]. This means we need to account for some leakage on the s_i choice bits of P_R, caused by the consistency check, which is modeled by the key query feature of $\mathcal{F}_{\Delta\text{-ROT}}$.

We also have an additional (non-correlated) **Chosen OTs** phase, which extends the base OTs further with arbitrary inputs from the sender, P_S, and the same choice bits from P_R, in a standard manner using the PRF. Both of these phases can be called repeatedly after the setup phase has run.

4.1 Security

We prove the following theorem by considering separately the two cases of a corrupted P_R and corrupted P_S.

Protocol $\Pi^{n,k,\mathbb{G}}_{\Delta\text{-ROT}}$

Let $F : \{0,1\}^k \times \{0,1\}^k \to \mathbb{G}^{n+k'}$ be a PRF and $H : \mathbb{G}^{n+k'} \times \mathbb{G}^{n+k'} \to \{0,1\}^k$ be a random oracle, where k' is chosen so that $|\mathbb{G}|^{k'} \geq 2^k$.

The protocol consists of two main commands, which can be repeatedly called after the **Initialize** stage. Both parties maintain a counter $c := 0$.

Initialize: On input $(\texttt{init}, s_1, \ldots, s_k)$ from P_R and $(\texttt{init})$ from P_S, where $(s_1, \ldots, s_k) \in \{0,1\}^k$:

1. P_S samples seeds $k_i^0, k_i^1 \leftarrow \{0,1\}^k$ for $i \in [k]$
2. The parties run $\mathcal{F}^{k,k}_{\mathsf{OT}}$, where P_S is sender with input $(k_i^0, k_i^1)_i$ and P_R is receiver with input s_i.
3. P_R receives $k_i^{s_i}$, for $i = 1, \ldots, k$.

Δ-OTs: On input $(\texttt{correlated})$ from P_R, and $(\texttt{correlated}, \Delta)$ from P_S, where $\Delta \in \mathbb{G}^n$:

1. *Create correlation:*
   (a) P_S samples $\rho \leftarrow \mathbb{G}^{k'}$ and sets $\Delta' := (\Delta \| \rho)$.
   (b) P_S computes $t_i^0 = F(k_i^0, c)$ and $t_i^1 = F(k_i^1, c)$, then computes
   $$u_i = t_i^0 + t_i^1 + \Delta', \quad i = 1, \ldots, k$$
   and sends these to P_R.
   (c) P_R computes $a_i = (-1)^{s_i} \cdot F(k_i^{s_i}, c) + s_i \cdot u_i \quad (= t_i^0 + s_i \cdot \Delta')$
   (d) Set $c := c + 1$.
2. *Consistency Check:* For every pair $(\alpha, \beta) \in [k]^2$:
   (a) P_S computes
   $$h^{0,0}_{\alpha,\beta} = H(t^0_\alpha - t^0_\beta), \quad h^{0,1}_{\alpha,\beta} = H(t^0_\alpha - t^1_\beta)$$
   $$h^{1,0}_{\alpha,\beta} = H(t^1_\alpha - t^0_\beta), \quad h^{1,1}_{\alpha,\beta} = H(t^1_\alpha - t^1_\beta)$$
   and sends these to P_R.
   (b) P_R checks that:
      i. $h^{s_\alpha, s_\beta}_{\alpha,\beta} = H(t^{s_\alpha}_\alpha - t^{s_\beta}_\beta)$
      ii. $h^{\bar{s}_\alpha, \bar{s}_\beta}_{\alpha,\beta} = H(u_\alpha - u_\beta - t^{s_\alpha}_\alpha + t^{s_\beta}_\beta)$
      iii. $u_\alpha \neq u_\beta$
3. *Output:* P_R outputs the first n components of a_i, and P_S outputs n components of $b_i := t_i^0$.

Chosen OTs: On input $(\texttt{chosen}, B, (u_i^0, u_i^1)_{i \in [k]})$ from P_S and $(\texttt{chosen}, B)$ from P_R, where each $u_i^b \in [0, B-1]$ and $B \leq 2^k$:

1. P_S sends $d_i^0 = F(k_i^0, c) \oplus u_i^0$ and $d_i^1 = F(k_i^1, c) \oplus u_i^1$ to P_R, for $i \in [k]$
2. P_R outputs $v_i = d_i^{s_i} \oplus F(k_i^{s_i}, c)$[a]
3. Set $c := c + 1$

[a] Only the first $\log_2 B$ output bits of the PRF are used in this stage.

Fig. 6. Base OT protocol for correlated OTs over an additive abelian group $\mathbb{G}$

Theorem 2. *If F is a secure PRF, H is a random oracle and $\lambda \leq k/2$ then protocol $\Pi^{n,k,\mathbb{G}}_{\Delta\text{-ROT}}$ securely realises the functionality $\mathcal{F}^{n,k,\mathbb{G}}_{\Delta\text{-ROT}}$ in the $\mathcal{F}^{k,k}_{\mathsf{OT}}$-hybrid model with static security.*

Corrupt P_R. To be secure against a corrupted P_R, it is vital that P_S appends the additional randomness ρ to the input Δ in step 1a, before creating the correlated OTs. Without this, P_R can obtain an 'oracle' that allows guessing whether a candidate value $\tilde{\Delta}$ equals the input of P_S or not by just computing one hash value. For example, let α, β be indices where $s_\alpha = s_\beta = 0$. Given $t^0_\alpha, t^0_\beta, u_\alpha$ and the hash values sent by P_S, P_R can compute $\tilde{t}^1_\alpha := u_\alpha - t^0_\alpha - \tilde{\Delta}$, and then test whether $h^{1,0}_{\alpha,\beta} = H(\tilde{t}^1_\alpha - t^0_\beta)$. This only holds if $\Delta = \tilde{\Delta}$, so allows testing any candidate input $\tilde{\Delta}$. Including the extra randomness ρ prevents this attack by ensuring that $\Delta' = (\Delta \| \rho)$ always has at least k bits of min-entropy (as long as $|\mathbb{G}|^{k'} \geq k$), so P_R can only guess Δ' with negligible probability.[5]

Note that this step was missing in the published versions of [ALSZ15,ALSZ17a], which leads to an attack on their actively secure OT extension protocol. This has now been fixed in a revised version [ALSZ17b].

To formally prove security against a corrupted P_R, we construct a simulator $\mathcal{S}_R$, who interacts with $\mathcal{F}_{\Delta\text{-ROT}}$ whilst simulating the communication from the honest P_S and the $\mathcal{F}_{\mathsf{OT}}$ functionality to the adversary, $\mathcal{A}$. $\mathcal{S}_R$ is described below.

1. In the **Initialize** phase, $\mathcal{S}_R$ receives the inputs $\{s_i\}_{i\in[k]}$ from $\mathcal{A}$ to $\mathcal{F}^{k,k}_{\mathsf{OT}}$, then samples random strings $k^{s_i}_i \leftarrow \{0,1\}^k$ and sends these to $\mathcal{A}$.
2. Whenever the **Δ-OTs** phase is called, $\mathcal{S}_R$ starts by sampling $u_i \leftarrow \mathbb{G}^{n+k'}$, for $i \in [k]$, and sends these to $\mathcal{A}$.
3. In the consistency check, $\mathcal{S}_R$ computes and sends the hash values $h^{s_\alpha,s_\beta}_{\alpha,\beta} = H(t^{s_\alpha}_\alpha - t^{s_\beta}_\beta)$ and $h^{\overline{s_\alpha},\overline{s_\beta}}_{\alpha,\beta} = H(u_\alpha - u_\beta - t^{s_\alpha}_\alpha + t^{s_\beta}_\beta)$. The other two values $h^{s_\alpha,\overline{s_\beta}}_{\alpha,\beta}, h^{\overline{s_\alpha},s_\beta}_{\alpha,\beta}$ are sampled uniformly at random.
4. $\mathcal{S}_R$ then sends $\{s_i\}_i$ to $\mathcal{F}_{\Delta\text{-ROT}}$, and computes the values $\{a_i\}_i$ as an honest P_R would in the protocol. $\mathcal{S}_R$ then sends $\{a_i\}_i$ to $\mathcal{F}_{\Delta\text{-ROT}}$ and increments the counter c.
5. Whenever the **Chosen OTs** phase is called, $\mathcal{S}_R$ calls $\mathcal{F}_{\Delta\text{-ROT}}$ with input $(\texttt{chosen}, B)$, and receives $\{v_i\}^k_{i=1}$. $\mathcal{S}_R$ computes $d^{s_i}_i = F(k^{s_i}_i, c) \oplus v_i$, samples a random string $d^{\overline{s_i}}_i$, then sends d^0_i, d^1_i to $\mathcal{A}$ and increments c.

Lemma 5. *If H is a (non-programmable, non-observable) random oracle and F is a secure PRF, then for every adversary $\mathcal{A}$ who corrupts P_R, and for every environment $\mathcal{Z}$, it holds that*

$$\mathsf{IDEAL}_{\mathcal{F}_{\Delta\text{-ROT}},\mathcal{S}_R,\mathcal{Z}} \stackrel{c}{\approx} \mathsf{HYBRID}^{\mathcal{F}_{\mathsf{OT}}}_{\Pi_{\Delta\text{-ROT}},\mathcal{A},\mathcal{Z}}$$

[5] This modification is not strictly needed for our application to the protocol in Section 3, because P_S's input to $\mathcal{F}_{\Delta\text{-ROT}}$ is always uniformly random and cannot be guessed. However, making this change allows for a simpler, more modular exposition and the functionality may be useful in other applications.

Proof. We consider a sequence of hybrid distributions, going from the ideal execution to the real execution, defined as follows. The first hybrid H_0 is identical to the ideal execution with $\mathcal{S}_R$ and $\mathcal{F}_{\Delta\text{-}\mathsf{ROT}}$.

Hybrid H_1: This is identical to H_0, except that both sets of keys k_i^0, k_i^1 are sampled by $\mathcal{S}_R$, instead of just $k_i^{s_i}$. We also modify the **Chosen OTs** phase so that both values d_i^0, d_i^1 are computed according to the protocol, using the PRF keys and the real inputs of the honest P_S.

Hybrid H_2: Here we modify H_1 further, so that the u_i values in the **Δ-OTs** stage are also computed according to the real protocol, using P_S's real input Δ and a random value ρ. These u_i values are then used by $\mathcal{S}_R$ to compute the a_i's which are sent to $\mathcal{F}_{\Delta\text{-}\mathsf{ROT}}$.

Hybrid H_3: This is the same as H_2, except the two hash values $h_{\alpha,\beta}^{s_\alpha,\overline{s_\beta}}, h_{\alpha,\beta}^{\overline{s_\alpha},s_\beta}$ are computed as in the protocol, instead of with random strings.

It is easy to see the view of $\mathcal{Z}$ in H_3 is identical to the real execution, since all messages are computed as an honest P_S would using the real inputs, and the outputs computed by $\mathcal{F}_{\Delta\text{-}\mathsf{ROT}}$ are the same as in the protocol.

Hybrids H_0 and H_1 are computationally indistinguishable because the keys $k_i^{\overline{s_i}}$ are unknown to $\mathcal{Z}$, which means the values $d_i^{\overline{s_i}}$ are indistinguishable from the previously uniform values, by a standard reduction to the PRF security. Similarly, H_1 and H_2 are computationally indistinguishable because $t_i^{\overline{s_i}}$ is pseudorandom based on the PRF, so masks P_S's input in u_i.

Regarding H_2, and H_3, notice that the two relevant hash values in H_3 are equal to

$$\begin{aligned}
h_{\alpha,\beta}^{s_\alpha,\overline{s_\beta}} &= H(t_\alpha^{s_\alpha} - t_\beta^{\overline{s_\beta}}) = H(t_\alpha^{s_\alpha} - u_\beta + t_\beta^{s_\beta} + \Delta') \\
h_{\alpha,\beta}^{\overline{s_\alpha},s_\beta} &= H(t_\alpha^{\overline{s_\alpha}} - t_\beta^{s_\beta}) = H(u_\alpha - t_\alpha^{s_\alpha} - \Delta' - t_\beta^{s_\beta})
\end{aligned}$$

Looking at the values on the right-hand side, P_R knows everything in both sets of inputs to H except for $\Delta' = (\Delta \| \rho)$.

The only way $\mathcal{Z}$ can distinguish between H_2 and H_3 is by querying H on one of the two inputs above, which occurs with probability at most $q \cdot |G|^{-k'} \leq q \cdot 2^{-k}$, where q is the number of random oracle queries, since ρ is uniformly random in $\mathbb{G}^{k'}$ and unknown to $\mathcal{Z}$. This completes the proof. $\square$

Corrupt P_S. When P_S is corrupt, the main challenge is to analyse soundness of the consistency check, similarly to [ALSZ15] (with a corrupt receiver in that protocol). Most of the analysis turns out to be identical, so we focus on the differences and state the main properties that we need from that work to show that our protocol securely realises $\mathcal{F}_{\Delta\text{-}\mathsf{ROT}}$. For the proof to go through here we need to assume that the statistical security parameter λ is no more than $k/2$, but can always increase k to ensure this holds.

Note that the main way a corrupt P_S may cheat in the protocol is by using different Δ' values when sending the u_i values. To account for this, for each $i \in [k]$ we define $\Delta_i = u_i - t_i^0 - t_i^1$; if P_S behaves honestly then we have $\Delta_1 = \cdots = \Delta_k = \Delta'$, otherwise they may be different.

The following lemma is analogous to [ALSZ15, Lemma 3.1], except we work over $\mathbb{G}$ instead of bit strings, and implies that the rest of the analysis of the consistency check from that work also applies in our case. Using the terminology of Asharov et al, if the consistency check passes for some set of messages $\mathcal{T} = \{\{k_i^0, k_i^1, u_i\}_i, \{H_{\alpha,\beta}\}_{\alpha,\beta}\}$ and some secret $\boldsymbol{s}$, we say that $\mathcal{T}$ is *consistent* with $\boldsymbol{s}$. If the check fails then it is *inconsistent*.

Lemma 6. *Let $\mathcal{T}_{\alpha,\beta} = \{k_\alpha^0, k_\alpha^1, k_\beta^0, k_\beta^1, u_\alpha, u_\beta, H_{\alpha,\beta}\}$ and suppose that H is a collision-resistant hash function. Then, except with negligible probability:*

1. *If $\Delta_\alpha \neq \Delta_\beta$ and $\mathcal{T}_{\alpha,\beta}$ is consistent with (s_α, s_β) then $\mathcal{T}_{\alpha,\beta}$ is inconsistent with $(\overline{s_\alpha}, \overline{s_\beta})$.*
2. *If $\Delta_\alpha = \Delta_\beta$ and $\mathcal{T}_{\alpha,\beta}$ is consistent with (s_α, s_β) then $\mathcal{T}_{\alpha,\beta}$ is also consistent with $(\overline{s_\alpha}, \overline{s_\beta})$*

Proof. For the first claim, suppose that $\Delta_\alpha \neq \Delta_\beta$, and $\mathcal{T}_{\alpha,\beta}$ is consistent with *both* (s_α, s_β) and $(\overline{s_\alpha}, \overline{s_\beta})$. Then from the consistency with (s_α, s_β) we have

$$h_{\alpha,\beta}^{s_\alpha,s_\beta} = H(t_\alpha^{s_\alpha} - t_\beta^{s_\beta}), \quad h_{\alpha,\beta}^{\overline{s_\alpha},\overline{s_\beta}} = H(u_\alpha - u_\beta - t_\alpha^{s_\alpha} + t_\beta^{s_\beta})$$

On the other hand, consistency with $(\overline{s_\alpha}, \overline{s_\beta})$ implies

$$h_{\alpha,\beta}^{\overline{s_\alpha},\overline{s_\beta}} = H(t_\alpha^{\overline{s_\alpha}} - t_\beta^{\overline{s_\beta}}), \quad h_{\alpha,\beta}^{s_\alpha,s_\beta} = H(u_\alpha - u_\beta - t_\alpha^{\overline{s_\alpha}} + t_\beta^{\overline{s_\beta}})$$

By the collision resistance of H, except with negligible probability it must then hold that

$$\begin{aligned} t_\alpha^{s_\alpha} - t_\beta^{s_\beta} &= u_\alpha - u_\beta - t_\alpha^{\overline{s_\alpha}} + t_\beta^{\overline{s_\beta}} \\ &= t_\alpha^{s_\alpha} - t_\beta^{s_\beta} + (\Delta_\alpha - \Delta_\beta) \end{aligned}$$

This means $\Delta_\alpha = \Delta_\beta$, which is a contradiction.

For the second claim, if $\Delta_\alpha = \Delta_\beta$ then $u_\alpha - u_\beta = t_\alpha^0 + t_\alpha^1 - t_\beta^0 - t_\beta^1$, and it can be seen from the above equations that the checks for (s_α, s_β) and $(\overline{s_\alpha}, \overline{s_\beta})$ are equivalent. □

For the case of a corrupted P_S, we construct a simulator $\mathcal{S}$, who interacts with $\mathcal{A}$ and plays the role of the honest P_R and the $\mathcal{F}_{\mathsf{OT}}$ functionality. $\mathcal{S}_S$ is described below.

1. $\mathcal{S}_S$ receives all the keys k_i^0, k_i^1 as inputs to $\mathcal{F}_{\mathsf{OT}}$, and then the messages u_i.
2. Using these it computes t_i^0, t_i^1 as in the protocol, and $\Delta_i = u_i - t_i^0 - t_i^1$. If P_S is honest then $\Delta_1 = \cdots = \Delta_k$.
3. $\mathcal{S}_S$ defines Δ' to be the most common of the Δ_i's, and sends the first n components of this as P_S's input to $\mathcal{F}_{\Delta\text{-}\mathsf{ROT}}$.
4. $\mathcal{S}_S$ then receives the sets of 4 hash values $H_{\alpha,\beta} = \{h_{\alpha,\beta}^{0,0}, h_{\alpha,\beta}^{0,1}, h_{\alpha,\beta}^{1,0}, h_{\alpha,\beta}^{1,1}\}$, for each $\alpha, \beta \in [k]$, as part of the consistency check.

5. $\mathcal{S}_S$ then uses the transcript of $\mathcal{A}$ to define a set of constraints on the secret $\boldsymbol{s}$ that must be satisfied for the consistency check to pass, by running Algorithm 1 from [ALSZ15]. Note that each of the constraints produced by this algorithm either fixes individual bits of $\boldsymbol{s}$, or the XOR of two bits of $\boldsymbol{s}$, so from this we can efficiently define a set $S(\mathcal{T}) \subset \{0,1\}^k$ which describes the set of all $\boldsymbol{s}$ that are consistent with the messages from $\mathcal{A}$.
6. $\mathcal{S}_S$ queries $\mathcal{F}_{\Delta\text{-ROT}}$ with $(\mathtt{guess}, S(\mathcal{T}))$. If the query is successful, $\mathcal{S}_S$ continues as if the consistency check passed, otherwise, $\mathcal{S}$ aborts.
7. If the check passed, $\mathcal{S}_S$ defines values b'_i, for $i \in [k]$ as specified below. These are sent to $\mathcal{F}_{\Delta\text{-ROT}}$.
8. Whenever the **Chosen OTs** phase is run, $\mathcal{S}_S$ uses its knowledge of the keys to extract P_S's inputs (u_i^0, u_i^1), and sends these to $\mathcal{F}_{\Delta\text{-ROT}}$.

The key part of the simulation is step 5, which uses the hash values sent in the consistency check to define the exact bits (or XOR of bits) of the honest P_R's secret $\boldsymbol{s}$ which the corrupt P_S tried to guess from cheating. Note that $\mathcal{S}_S$ does not define its own secret $\boldsymbol{s}$, as this is already sampled internally in the functionality $\mathcal{F}_{\Delta\text{-ROT}}$. Therefore, $\mathcal{S}_S$ sends a description of all the possible consistent values of $\boldsymbol{s}$ to the $(\mathtt{guess})$ command of $\mathcal{F}_{\Delta\text{-ROT}}$ to see if the cheating attempt was successful or not.

Lemma 7. *If $\lambda \leq k/2$ and H is collision-resistant then for every adversary $\mathcal{A}$ who corrupts P_S, and for every environment $\mathcal{Z}$, it holds that*

$$\mathsf{IDEAL}_{\mathcal{F}_{\Delta\text{-ROT}},\mathcal{S}_S,\mathcal{Z}} \stackrel{c}{\approx} \mathsf{HYBRID}^{\mathcal{F}_{\mathsf{OT}}}_{\Pi_{\Delta\text{-ROT}},\mathcal{A},\mathcal{Z}}$$

Proof (sketch): Define the transcript of the simulation up until the consistency check by $\mathcal{T} = \{\{k_i^0, k_i^1, u_i\}_i, \{H_{\alpha,\beta}\}_{\alpha,\beta}\}$, and define $\mathsf{consistent}(\mathcal{T}, \boldsymbol{s})$ to be 1 if the consistency check would pass if $\boldsymbol{s}$ is the secret of P_R, and 0 otherwise. From Algorithm 1 in step 5, recall that the we defined set of all possible secrets $\boldsymbol{s} \in \{0,1\}^k$ of an honest P_R for which the check would pass to be $S(\mathcal{T}) = \{\boldsymbol{s} \in \{0,1\}^k : \mathsf{consistent}(\mathcal{T}, \boldsymbol{s}) = 1\}$, where $\mathcal{T}$ is the transcript produced by $\mathcal{A}$. Note that from the definition of $S(\mathcal{T})$, the probability that the consistency check passes is $|S(\mathcal{T})|/2^k$ in both the real and ideal executions. To complete the proof we just need to show how to extract the correct values b'_i defining P_S's output.

Below we give the key results from [ALSZ15] needed to analyse the consistency check.

Lemma 8. *For a given transcript $\mathcal{T}$, let U be the largest set of indices such that for all $i, j \in U$, $\Delta_i = \Delta_j$, and let $B = [k] \setminus U$ be the complementary set. We have:*

1. *If $B > \lambda$ then the probability of passing the consistency check is $\leq 2^{-\lambda}$.*
2. *If the consistency check passes, then for all $\boldsymbol{s}' \in S(\mathcal{T})$, it holds that either the bits $\{\boldsymbol{s}'_i\}_{i \in B}$ are fixed, or the bits $\{\boldsymbol{s}'_i\}_{i \in U}$ are fixed.*

Proof. The first claim follows from Claim B.3 of [ALSZ15] and Lemma 6. The second can be seen from inspection of the proof of Claim B.4 in the same work. □

From the first item, we conclude that $|B| \le \lambda$, except with negligible probability. We claim that this means we first be in the first case of item 2 in the lemma. If the bits $\{\boldsymbol{s}'\}_{i \in U}$ were fixed then the adversary must have guessed $|U|$ bits of the secret to pass the check, but since $|U| \ge k - \lambda \ge \lambda$, this can only happen with probability $\le 2^{-\lambda}$.

This implies that (except with negligible probability) the bits of $\boldsymbol{s}$ sampled by $\mathcal{F}_{\Delta\text{-ROT}}$ are fixed at the positions $i \in B$, so $\mathcal{S}$ can define a value $b_i' = t_i^0 + s_i \cdot (\Delta_i - \Delta')$ from the fact that s_i is fixed, for all $i \in B$. We then have $b_i' = a_i - s_i \cdot \Delta$, so this defines the correct output that $\mathcal{S}$ sends to $\mathcal{F}_{\Delta\text{-ROT}}$ in step 7. Note that for all $i \in U$ we have $\Delta_i = \Delta'$, so we can just let $b_i' = t_i^0$. These outputs are then identically distributed to the outputs of P_S in the real protocol, so (accounting for the negligible failure events) the simulation is statistically close. □

Acknowledgements

I would like to thank Claudio Orlandi for the observation that an XOR-homomorphic PRG would imply non-interactive OT extension, which inspired this work. I am also grateful to the PKC 2018 reviewers for helpful comments.

This work has been supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 731583 (SODA), and the Danish Independent Research Council under Grant-ID DFF–6108-00169 (FoCC).

A Conversion to 1-out-of-2 OTs

The main protocol in Section 3 produces a batch of random 1-out-of-p_i OTs, for multiple small primes p_i. If an application requires 1-out-of-2 OTs (as is common) then we can convert each of the 1-out-of-p_i OTs to a 1-out-of-2 OT at a cost of $O(\log p_i)$ bits of communication using standard techniques, with active security.[6]

In the protocol, shown in Fig. 7, the receiver first converts the random choice c from the 1-out-of-N OT into its chosen input bit b by sending the difference $d = c - b \mod N$. The sender, who initially has random strings $r_0, \dots, r_{N-1}$, then uses r_d and $r_{d+1 \bmod N}$ to one-time-pad encrypt its two input messages. The receiver can recover exactly one of these, corresponding to $r_c = r_{d+b}$.

Security of the protocol is straightforward. The only concern is that if the receiver is corrupt, P_R might choose an inconsistent value $b \in \{2, ..., p-1\}$ instead of a bit, so learns a random string instead of one of the sender's two inputs. However, the fact that a corrupt P_R may not learn a valid output does not pose a problem, since in this case, in the security proof the simulator can just send an arbitrary choice bit to the $\mathcal{F}_{\mathsf{OT}}$ functionality and simulate the e_0, e_1 messages from the sender with random strings.

[6] It is possible to convert a 1-out-of-N OT into $\log_2 N$ 1-out-of-2 OTs [NP99,KK13], but this technique would not improve the asymptotic communication cost in our case, and is only passively secure.

Protocol Π_{Conv}

Sender input: strings $s_0, \ldots, s_{N-1}$
Receiver input: choice $b \in \{0, 1\}$

1. P_S obtains random strings $r_0, \ldots, r_{N-1}$ from $\mathcal{F}_{p_i\text{-}\mathsf{ROT}}$
2. P_R obtains a random choice $c \in \{0, \ldots, N-1\}$, and the string r_c.
3. P_R sends $d = c - b \mod N$ to P_S
4. P_S sends $e_0 = s_0 \oplus r_d$ and $e_1 = s_1 \oplus r_{d+1 \bmod N}$
5. P_R recovers s_b by computing $e_b \oplus r_c$

Fig. 7. Conversion from random 1-out-of-N OT to chosen 1-out-of-2 OT

References

AKPW13. Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited - new reduction, properties and applications. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 57–74. Springer, Heidelberg, August 2013.

ALSZ13. Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer and extensions for faster secure computation. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 13*, pages 535–548. ACM Press, November 2013.

ALSZ15. Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer extensions with security for malicious adversaries. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 673–701. Springer, Heidelberg, April 2015.

ALSZ17a. Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer extensions. *Journal of Cryptology*, 30(3):805–858, 2017.

ALSZ17b. Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer extensions with security for malicious adversaries. Cryptology ePrint Archive, Report 2015/061, 2017. Version 20171121:125742, https://eprint.iacr.org/2015/061.

Bea96. Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th ACM STOC*, pages 479–488. ACM Press, May 1996.

BFP+15. Abhishek Banerjee, Georg Fuchsbauer, Chris Peikert, Krzysztof Pietrzak, and Sophie Stevens. Key-homomorphic constrained pseudorandom functions. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 31–60. Springer, Heidelberg, March 2015.

BGG+14. Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Heidelberg, May 2014.

BGI16. Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 509–539. Springer, Heidelberg, August 2016.

BGI17. Elette Boyle, Niv Gilboa, and Yuval Ishai. Group-based secure computation: Optimizing rounds, communication, and computation. In *EUROCRYPT (2)*, volume 10211 of *Lecture Notes in Computer Science*, pages 163–193, 2017.

BLMR13. Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 410–428. Springer, Heidelberg, August 2013.

BMR16. Marshall Ball, Tal Malkin, and Mike Rosulek. Garbling gadgets for boolean and arithmetic circuits. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 16*, pages 565–577. ACM Press, October 2016.

BP14. Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 353–370. Springer, Heidelberg, August 2014.

BPR12. Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737. Springer, Heidelberg, April 2012.

BV15. Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic PRFs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 1–30. Springer, Heidelberg, March 2015.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

DHRW16. Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 93–122. Springer, Heidelberg, August 2016.

GMW86. Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th FOCS*, pages 174–187. IEEE Computer Society Press, October 1986.

GMW87. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

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

GVW13. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 545–554. ACM Press, June 2013.

HIJ+16. Shai Halevi, Yuval Ishai, Abhishek Jain, Eyal Kushilevitz, and Tal Rabin. Secure multiparty computation with general interaction patterns. In Madhu Sudan, editor, *ITCS 2016*, pages 157–168. ACM, January 2016.

HIKN08. Danny Harnik, Yuval Ishai, Eyal Kushilevitz, and Jesper Buus Nielsen. OT-combiners via secure computation. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 393–411. Springer, Heidelberg, March 2008.

HW15. Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In Tim Roughgarden, editor, *ITCS 2015*, pages 163–172. ACM, January 2015.

IKNP03. Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 145–161. Springer, Heidelberg, August 2003.

IKOS07. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30. ACM Press, June 2007.

IKOS08. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 433–442. ACM Press, May 2008.

IKOS09. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Extracting correlations. In *50th FOCS*, pages 261–270. IEEE Computer Society Press, October 2009.

IPS08. Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 572–591. Springer, Heidelberg, August 2008.

IR89. Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *21st ACM STOC*, pages 44–61. ACM Press, May 1989.

KK13. Vladimir Kolesnikov and Ranjit Kumaresan. Improved OT extension for transferring short secrets. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 54–70. Springer, Heidelberg, August 2013.

KKRT16. Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious PRF with applications to private set intersection. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 16*, pages 818–829. ACM Press, October 2016.

KOS15. Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure OT extension with optimal overhead. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 724–741. Springer, Heidelberg, August 2015.

NNOB12. Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 681–700. Springer, Heidelberg, August 2012.

NP99. Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *31st ACM STOC*, pages 245–254. ACM Press, May 1999.

OOS17. Michele Orrù, Emmanuela Orsini, and Peter Scholl. Actively secure 1-out-of-N OT extension with application to private set intersection. In *RSA Conference (Cryptographers' Track) 2017*, 2017.

PSS17. Arpita Patra, Pratik Sarkar, and Ajith Suresh. Fast actively secure OT extension for short secrets. In *NDSS 2017*. Internet Society, 2017.

PSZ18. Benny Pinkas, Thomas Schneider, and Michael Zohner. Scalable private set intersection based on ot extension. *ACM Trans. Priv. Secur.*, 21(2):7:1–7:35, January 2018.

Yao86. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.