Improved OT Extension for Transferring Short Secrets and Application to Secure Multi-Party Computation

Abstract

We propose an optimization and generalization of OT extension of Ishai et al. of Crypto 2003. For computational security parameter k, our OT extension for short secrets offers $O(\log k)$ factor performance improvement in communication and computation, compared to prior work. In concrete terms, for today's security parameters, this means approx. factor 2-3 improvement.

This results in corresponding improvements in applications relying on such OT. In particular, for two-party semi-honest SFE, this results in $O(\log k)$ factor improvement in communication over state of the art Yao Garbled Circuit, and has the same asymptotic complexity as the recent multi-round construction of Kolesnikov and Kumaresan of SCN 2012. For multi-party semi-honest SFE, where their construction is inapplicable, our construction implies $O(\log k)$ factor communication and computation improvement over best previous constructions. As with our OT extension, for today's security parameters, this means approximately factor 2 improvement in semi-honest multi-party SFE.

Our building block of independent interest is a novel IKNP-based framework for 1-out-of-n OT extension, which offers $O(\log n)$ factor performance improvement over previous work (for $n \leq k$), and concrete factor improvement of up to 9 for today's security parameters (n=k=128).

Our protocol is the first practical OT with communication/computation cost sublinear in the security parameter (prior sublinear constructions Ishai et al. [IKOS08, IKOS09] are not efficient in concrete terms).

Keywords: OT extension, 1-out-of-2 OT, 1-out-of-n OT.

1 Introduction

Our main contribution is an asymptotic and concrete efficiency improvement of Oblivious Transfer (OT) extension of Ishai et al. [IKNP03]. Our improvement applies to OT transfers of short secrets. In this Introduction we first motivate the problem, and then give intuition behind our approach.

Oblivious Transfer (OT) [Rab81, EGL85, BCR87, Kil88, NP05] is a fundamental cryptographic primitive that is used as a building block in a variety of cryptographic protocols. It is a critical piece in general secure computation [Yao86, GMW87, GV88, Kil88], as well as in a number of tailored solutions to specific problems of interest, such as contract signing [EGL85]. OT performance improvement directly translates into that of secure function evaluation (SFE). In turn, SFE performance is the subject of major research effort in cryptography [IKNP03, BDNP08, MNPS04, KS08, CHK$^+$12, HKE12, PSSW09, KSS09, IPS08, HEKM04, AJLA$^+$12, SS11, BCD$^+$09, LOP11, NNOB12, LP07]. Our work can be plugged into several existing candidate solutions, resulting in factor $2-3$ performance improvement, which is a major step forward in the state of the art of secure computation.

1.1 Secure Computation

SFE allows two (or more) parties to evaluate any function on their respective inputs x and y, while maintaining privacy of both x and y. SFE is justifiably a subject of an immense amount of research. Efficient SFE algorithms enable a variety of electronic transactions, previously impossible due to mutual mistrust of participants. Examples include auctions, contract signing, set intersection, etc. As computation and communication resources have increased, SFE of many useful functions has become practical for common use. Still, SFE of many of today's functions of interest carries costs sufficient to deter would-be adopters, who instead choose stronger trust models, entice users to give up their privacy with incentives, or use similar crypto-workarounds. We believe that truly practical efficiency is required for SFE to see use in real-life applications.

The current state of the art of SFE research is quite sophisticated. Particularly in the semi-honest model, there have been very few asymptotic/qualitative improvements since the original protocols of Yao [Yao86] and Goldreich et al. [GMW87]. Possibly the most important development in the area of SFE since the 1980's was the very efficient OT extension technique of Ishai et al. [IKNP03], which allowed to evaluate an arbitrarily large number of OTs by executing a small (security parameter) number of (possibly inefficient) "bootstrapping" OT instances, and a number of symmetric key primitives. This possibility of cheap OTs made a dramatic difference for securely computing functions with large inputs (relative to the size of the function).

As secure computation moves from theory to practice, even "small" improvements can have a significant effect. Today, even small factor performance improvements to state-of-the-art algorithms are quite hard to achieve, and are most welcome. This is especially true about the semi-honest model protocols, where the space for improvement appears to be much smaller than in the malicious model.

In this work, we propose an improvement to OT extension of Ishai et al. [IKNP03], for the case of OT of short secrets. As we will describe below, this will result in a new multi-party SFE protocol, which is approximately factor 2 (and, asymptotically factor $O(\log k)$) more efficient than state of the art. Our constructions also improve on standard two-party garbled circuit protocols in asymptotic ($O(\log k)$) and concrete terms, and offer performance in line with the recent work of [KK12].

1.2 Secure Computation via OT

We note some of the results that show how to efficiently construct protocols for general secure computations when given black box implementation of OT.

Garbled Circuit. Probably the most well-known approach is the Garbled Circuit (GC) approach of Yao [Yao86]. The GC construction can be viewed as encryption of the boolean circuit implementing the computed function. The circuit encryption includes encrypting all of the gates' truth tables and the signals on each of the circuit's wires, including input and output wires. The circuit encryption has the property that each of the encrypted circuit gates can be evaluated "under encryption" given encryptions its inputs. Clearly, this allows to compute the encryption of the output, which can then be decrypted among the two players, achieving secure computation. Relevant to our work, in this approach OT is used to deliver the encryptions of the circuit evaluator's input to him. The cost of the protocol is linear in the size of the circuit.

GMW. The approach of Goldreich, Micali, and Wigderson (GMW) [GMW87] is also based on secure evaluation of a boolean circuit representing the computed function. In their construction, the two players secret-share the values of the circuit wires, starting from the input wires. Then they evaluate the circuit, gate by gate (or, rather, layer by layer), as follows. For the boolean addition gate (XOR), the parties simply locally add the shares of the gate's input that they hold. It is not hard to see that this will result in the two players secret-sharing the output of the XOR gate. However, in case of the boolean multiplication gate (AND), the players need a round of interaction, and one execution of OT to secret-share the the output of the AND gate. The cost of the protocol is linear in the size of the circuit.

Kilian [Kil88] was the first to show a one-round IT reduction (of complexity $\Theta(4^d)$) of SFE to OT, where d is the depth of the boolean circuit. Kilian relies on Barrington's [Bar86] representation of $\mathbf{NC}^1$ circuits as permutation BPs. It is possible to replace Barrington's representation in Kilian's construction with a more efficient construction of Cleve [Cle90] (see, e.g. Cramer et al. [CFIK03]). The resulting complexity is $\Theta(2^d 2^{\Theta(\sqrt{d})})$, which is the best previously known for $\mathbf{NC}^1$ circuits and (re)balanced formulas. As in Yao [Yao86], OT is used to transfer the encoding of the evaluator's input to him.

Ishai and Kushilevitz [IK00, IK02] suggested a way of representing a circuit as a predicate on a vector of degree 3 *randomizing polynomials* (degree of the input variables x_i is 1). Their construction assigns an (exponential in d in size) polynomial representation to each wire of the corresponding fan-out 1 circuit, and implies a one-round SFE-to-OT reduction, of complexity $\Theta(4^d)$. As in Yao [Yao86] and Kilian [Kil88], OT is used to transfer the encoding of the evaluator's input to him.

Kolesnikov [Kol05] proposed another, more efficient, one-round information-theoretic SFE-to-OT reduction. His Gate Evaluation Secret Sharing (GESS) scheme is designed to match with the circuit gate function g, so that the two secrets corresponding to the inputs of the gate allow to reconstruct the secret of the output wire of the gate. Because the wires' secrets are allowed to have a lot of common information, the secret sharing scheme can be made very efficient. As in above works, the OT is used to transfer the encoding of the circuit evaluator's input to him.

1.3 Secure Computation and OT Efficiency Considerations

As we briefly mentioned above, the efficiency of OT plays a critical role in the overall efficiency of secure computation. It is so to the point that OT performance determines which is the most efficient approach. Until recently, in the semi-honest model, Yao's Garbled Circuit was a clear winner. With the work of [KK12] and our improved OT extension technique, the GMW approach will outperform Yao with a factor of ≈ 2 for today's security parameters. Asymptotically, the performance improvement is logarithmic in security parameter compared to GC.

On the cost of SFE rounds. One common consideration in SFE protocol design is the number of rounds. Indeed, in some scenarios the latency associated with the communication rounds can more than double the total execution time. This holds, e.g., when the evaluated circuit is small; with the GMW evaluation, where we need a round of communication per layer of the circuit, the latency may be costly for deep and narrow circuits. This may cause somewhat increased latency of an individual computation – a possible inconvenience to the user of interactive applications.

At the same time, many SFE protocols allow for a significant precomputation and also for streaming, where message transmission may begin (and even a response may be received) before the sender completes the computation and transmission of the message. Thus, round-related latency will usually not be a wasted time and will not cause extra delays. Most importantly, with the speed of the CPU advancing faster than that of communication, the true bottleneck for SFE already is the channel transmission capacity, even for high-speed gigabit LAN.

In sum, we argue that in many scenarios, the number of communication rounds in SFE often plays an insignificant role in practice, and round-related latency either has no impact on performance, or it can be tolerated in exchange of achieving higher throughput.

1.4 Our Contributions

Our main contribution is an asymptotic and concrete efficiency improvement of Oblivious Transfer (OT) extension of Ishai et al. [IKNP03]. Our improvement applies to OT transfers of short secrets.

1-out of-2 OT extension. For a security parameter k, our $O(\log k)$ asymptotic improvement results in concrete efficiency improvement of about factor 2 for today's security parameters ($k = 240$). This immediately translates in same asymptotic and concrete improvements in multi-party computation in the semi-honest setting, when applied to state of the art solutions, which are based on GMW protocols.

1-out of-n OT extension offers $O(\log n)$ factor performance improvement over previous work (for $n < k$), and concrete factor improvement of up to 9 for today's security parameters.

Further, our protocol is the first OT sublinear in the security parameter other than the non-black-box construction of Ishai et al. [IKOS08], and is the only practical OT with this property. Our resulting secure computation protocols can also be viewed as a significant generalization of the technique of [KK12], which offered logarithmic in k improvement over state-of-the-art Yao's GC, but, in particular, did not extend to multiparty setting. We work in the RO model, but, like in [IKNP03], we can also use a variant of correlation-robust hash functions (cf. Appendix F).

1.4.1 Applications and Practical Performance Impact

As noted above, our 1-out of-2 OT construction immediately offers approximately factor 2 improvement in nearly all multi-party protocols – GMW and its variants[1]. In two-party computation, a similar, but more limited in scope, improvement was recently achieved [KK12]. In particular, [KK12] didn't work well on very shallow circuits, such as inner product computation. For such circuits, we have $O(\log k)$ improvement over 2PC state of the art, including [KK12].

As noted, our 1-out of-n OT gives logarithmic performance improvement in transferring one in n random secret keys. This is a common use of OT (cf. Garbled Circuits). However, in some cases, where the OT of specific secrets is required, the improvement factor may be smaller due to the fact that all n secrets encrypted with the n keys need to be transferred. In this case, logarithmic improvement applies only to the offline phase, where the secrets are not available.

Another application which immediately benefits from this work is string-selection OT (SOT), a variant of 1-out of-n OT and a building block of [KK12]. In SOT, the receiver selects one of the sender's two secrets based on his n-bit selection string.

1.5 Related Work

OT is a critical and heavily used component in much of cryptography, and in particular in secure computation protocols. Naturally, a lot of effort went into optimizing its performance. Unfortunately, there are fundamental limits to OT efficiency. Impagliazzo and Rudich [IR89] showed that a black-box reduction from oblivious transfer to a one-way function or a one-way permutation would imply $\mathbf{P} \neq \mathbf{NP}$. It is further not known whether such non-black-box reductions exist.

[1]We note that additional small factor improvement is possible by using PRG to compress the $m \times k$ matrix in OT extension (see Appendix D). This, combined with our improved OT, brings the total improvement to our claimed factor $2 - 3$. Because our PRG compression also applies to standard IKNP, we did not include it in our performance tables for fair comparison of the two approaches.

Beaver [Bea96] was the first to propose OT extension, a non-black-box scheme where a large number of OTs can be obtained from a small number of OTs (possibly executed by using public-key primitives) and one-way functions. Lindell and Zarosim [LZ13] recently showed that one-way functions are in fact needed for OT extension.

Ishai, Kilian, Nissim, and Petrank [IKNP03], in their breakthrough work showed a black-box OT extension, which is truly practical. Its cost, in addition to the security parameter number of base OTs, is only two random oracle (RO) evaluations and output transfers. By dramatically changing the cost structure of two-party SFE, especially in the semi-honest model, this work enabled greatly improved SFE for functions with large inputs, previously considered too costly due to the need of a large number of public key operations. It also started a rise in the study of GMW-based SFE protocols, where an OT is needed per multiplicative node. Indeed, recent (yet unoptimized) GMW-based and multiple-round protocols began to outperform traditional GC protocols. In particular, [NNOB12] outperforms state-of-the-art GC protocols in the malicious model, and [KK12] outperforms state-of-the-art GC protocols in the semi-honest model. In addition to considering the semi-honest model, [IKNP03] presents a construction secure against malicious participants. In a few follow-up works [Nie07, HIKN08], the performance of the malicious setting of the IKNP OT extension was substantially improved. We present the high-level idea of the basic IKNP construction in Section 3.2.

By employing a more efficient pseudorandom generator in Beaver's non-black-box OT extension protocol, Ishai, Kushilevitz, Ostrovsky, and Sahai [IKOS08] obtained an asymptotically more efficient (but expensive in concrete terms) construction for oblivious transfer extension, and consequently for secure computation. In fact, their protocol enjoys a *constant computational/communication overhead* over an insecure evaluation of the function to be evaluated. In order to obtain these strong efficiency results, Ishai et al. [IKOS08] make strong complexity-theoretic assumptions on pseudorandom generators. Specifically, they assume that there exists an (arbitrary stretch) pseudorandom generator in $\mathbf{NC}^0$ [AIK04, App12].

In this work, we show logarithmic in the security parameter improvement for black-box OT extension transfer of short secrets. In other words, we improve efficiency of the black-box OT extension protocol of Ishai et al. [IKNP03] asymptotically by a $\log(k/\ell)$ factor when the length of the transferred secrets is ℓ. This has important practical applications for secure computation solutions, such as GMW, that require precisely 1-out-of-2 OT of 1-bit secrets. We calculate both asymptotic and concrete performance of the resulting protocols. Our constructions are presented in the semi-honest model.

We stress that in contrast to the non-black-box techniques of Ishai et al. [IKOS08], our extension protocol makes only black-box use of a (non-programmable) random oracle (or a $\mathcal{C}$-correlation robust hash function, see Section F). Also, unlike [IKOS08, IKOS09] who mainly focus on asymptotic complexity, we calculate also the concrete efficiency of our construction, and demonstrate a factor of approximately 2 improvement over state-of-the-art protocols [IKNP03, CHK$^+$12].

Finally, we mention PIR work (e.g., [Lip08]) that construct communication efficient 1-out of-n OT protocols but perform $O(n)$ computationally intensive (e.g., public-key operations) per instance. We perform a fixed number of public-key operations independent of the number of OT instances.

2 Overview of Our Approach

We give a high-level overview of our solution prior to presenting its technical details in Section 4. We aim that the reader somewhat familiar with the IKNP construction should understand the main idea of our construction from this overview.

Consider the random $m \times k$ matrix designed by [IKNP03], which is transferred column-wise via k 1-out-of-2 base OTs from the receiver $\mathcal{R}$ to the sender $\mathcal{S}$. In [IKNP03], each row of this matrix is used to implement a 1-out-of-2 OT, as it has the randomness from which a random OT can be constructed.

Our main observation is that, *for the same communication cost*, each row of this matrix can be instead used to perform a 1-out-of-n OT, but of shorter secrets. Further, a 1-out-of-n OT of $\log n$-bit long secrets can be trivially used to construct $\log n$ 1-out-of-2 1-bit OTs, which is precisely the kind of OT needed in the GMW protocol and its variants. Thus, effectively, we trade the length of the OT-transferred secrets for the number of OTs, which results in significant gain for MPC applications.

The intuition for our 1-out-of-n OT is as follows. First, recall that in IKNP, for each column of the $m \times k$ matrix, $\mathcal{S}$ randomly selects (via OT), whether he receives the random column, or the random column XORed with the m-bit long input of $\mathcal{R}$. Viewed row-wise, this effectively means that for each row j, $\mathcal{S}$ either receives (via OT) the j-th row of the randomly chosen $m \times k$ matrix (if $\mathcal{R}$'s j-th selection bit is 0), or that row XORed with his k-bit selection vector to the OT (if $\mathcal{R}$'s j-th selection bit is 1). Then $\mathcal{S}$ masks each of his two j-th input secrets with (RO hashes of) vector received as output from OT and the same vector XORed with its k-bit selection vector respectively and sends both to $\mathcal{R}$, who is able to take the mask off one of the two messages. The second masked message remains hidden since $\mathcal{R}$ does not learn the selection vector provided by $\mathcal{S}$.

In the following, let $\mathcal{C}$ denote a binary code, and let r_j denote the input of $\mathcal{R}$ to the j-th instance of 1-out-of-n OT. In our 1-out-of-n OT, we modify the scheme presented above such that for each row j, $\mathcal{S}$ receives (via OT) the actual j-th row of the $m \times k$ matrix XORed with a vector that is the result of the r_j-th codeword in $\mathcal{C}$ bitwise-ANDed with the k-bit selection vector. This allows $\mathcal{S}$ to generate n random pads from each row of the matrix—the i-th such pad being the j-th row it received (via OT) XORed with a vector that is the result of the i-th codeword in $\mathcal{C}$ bitwise-ANDed with the k-bit selection vector. These n random pads may then be used by $\mathcal{S}$ to carry out a 1-out-of-n OT with $\mathcal{R}$. The security of this construction naturally depends on the underlying code. The exact property that we need is that $\mathcal{C}$ must contain at least n codewords, each of length at most k, such that the codewords in $\mathcal{C}$ are spaced as far apart as possible from each other. This, combined with the fact that $\mathcal{R}$ does not learn the selection vector provided by $\mathcal{S}$, will ensure that $\mathcal{R}$ can efficiently recover only one of the n pads used by $\mathcal{S}$. The above is presented in detail in Section 4.

Using Walsh-Hadamard code for $\mathcal{C}$ gives a 1-out-of-n OT for n equal to the security parameter k. This OT is suitable for generation of $\log n$ 1-out-of-2 OTs (Section 5.1). Using a higher-rate code with similarly high distance results in 1-out-of-n OT for any n polynomial in k (Section 5.3).

3 Preliminaries and Notation

3.1 Notation

We use the notation OT_ℓ^m to denote m instances of 1-out-of-2 string-OT where the string is ℓ bits long. Let $\mathcal{S}$ denote the sender, and let $\mathcal{R}$ denote the receiver. In 1-out-of-2 OT, the sender's input is $\{(x_{j,0}, x_{j,1})\}_{j\in[m]}$, i.e., m pairs of strings, each of length ℓ, and the receiver holds input $\{r_j\}_{j\in[m]}$, where each r_j is an integer which is either 0 or 1. Note that if $\mathcal{S}$ provides input $\{(x_{j,0}, x_{j,1})\}_{j\in[m]}$ to OT_ℓ^m, and if $\mathcal{R}$ provides input $\{r_j\}_{j\in[m]}$ to OT_ℓ^m, then $\mathcal{R}$ receives back $\{x_{j,r_j}\}_{j\in[m]}$, while $\mathcal{S}$ receives nothing.

In Section 4, we construct protocols for 1-out-of-n OT, which is a straightforward generalization of 1-out-of-2 OT. We explain this further. We use the notation $\binom{n}{1}$-OT_ℓ^m to denote m instances of 1-out-of-n string-OT where the string is ℓ bits long. In 1-out-of-n OT, the sender's input is $\{(x_{j,0},\ldots,x_{j,n-1})\}_{j\in[m]}$, and the receiver holds input $\{r_j\}_{j\in[m]}$, where each r_j is an integer which between 0 and $n-1$. Note that if $\mathcal{S}$ provides input $\{(x_{j,0},\ldots,x_{j,n-1})\}_{j\in[m]}$ to $\binom{n}{1}$-OT_ℓ^m, and if $\mathcal{R}$ provides input $\{r_j\}_{j\in[m]}$ to $\binom{n}{1}$-OT_ℓ^m, then $\mathcal{R}$ receives back $\{x_{j,r_j}\}_{j\in[m]}$, while $\mathcal{S}$ receives nothing.

Following the convention in IKNP, we denote vectors in bold, and matrices in capitals. For a matrix A, we let $\mathbf{a}_j$ denote the j-th row of A, and $\mathbf{a}^i$ denote the i-th column of A. If $\mathbf{a} = a_1\|\cdots\|a_p$ and $\mathbf{b} = b_1\|\cdots\|b_p$ are two vectors, then we define $\oplus$ and $\odot$ operations as follows. We use the notation $\mathbf{a}\oplus\mathbf{b}$ to denote the vector $(a_1\oplus b_1)\|\cdots\|(a_p\oplus b_p)$. Similarly, the notation $\mathbf{a}\odot\mathbf{b}$ denotes the vector $(a_1\cdot b_1)\|\cdots\|(a_p\cdot b_p)$. Finally, suppose $c\in\{0,1\}$, then $c\cdot\mathbf{a}$ denotes the vector $(c\cdot a_1)\|\cdots\|(c\cdot a_p)$.

Our constructions assume the existence of a random oracle H. We denote the security parameter by k, and assume (without loss of generality) that it is a power of 2.

3.2 IKNP OT Extension

In this section, we present the OT extension protocol of Ishai, Kilian, Nissim, and Petrank [IKNP03]. The protocol will reduce OT_ℓ^m to OT_m^k. As described in Appendix B.3, this implies a reduction to OT_k^k with some additional cost. The security of the protocol holds as long as the receiver is semi-honest. (Note: the sender may be malicious.)

We now describe the protocol that realizes OT_ℓ^m given ideal access to OT_m^k.

Input of $\mathcal{S}$: m pairs $(x_{j,0},x_{j,1})$ of ℓ-bit strings, $1\le j\le m$.
Input of $\mathcal{R}$: m selection bits $\mathbf{r}=(r_1,\ldots,r_m)$.
Common Input: a security parameter k.
Oracle: a random oracle $H:[m]\times\{0,1\}^k\to\{0,1\}^\ell$.
Cryptographic Primitive: an ideal OT_m^k primitive.

1. $\mathcal{S}$ chooses $\mathbf{s}\leftarrow\{0,1\}^k$ at random. Let s_i denote the i-th bit of $\mathbf{s}$.

2. $\mathcal{R}$ forms $m\times k$ matrices T_0, T_1 in the following way:

   - Choose $\mathbf{t}_{j,0},\mathbf{t}_{j,1}\leftarrow\{0,1\}^k$ at random such that $\mathbf{t}_{j,0}\oplus\mathbf{t}_{j,1}=(r_j\|\cdots\|r_j)$.

   Let $\mathbf{t}_0^i,\mathbf{t}_1^i$ denote the i-th column of matrices T_0, T_1 respectively.

3. $\mathcal{S}$ and $\mathcal{R}$ interact with OT_m^k in the following way:

   - $\mathcal{S}$ acts as *receiver* with input $\{s_i\}_{i\in[k]}$.
   - $\mathcal{R}$ acts as *sender* with input $\{\mathbf{t}_0^i,\mathbf{t}_1^i\}_{i\in[k]}$.
   - $\mathcal{S}$ receives output $\{\mathbf{q}^i\}_{i\in[k]}$.

   $\mathcal{S}$ forms $m\times k$ matrix Q such that the i-th column of Q is the vector $\mathbf{q}^i$. (Note $\mathbf{q}^i = t_{s_i}^i$.) Let $\mathbf{q}_j$ denote the j-th row of Q. (Note $\mathbf{q}_j=((\mathbf{t}_{j,0}\oplus\mathbf{t}_{j,1})\odot\mathbf{s})\oplus\mathbf{t}_{j,0}$. Simplifying, $\mathbf{q}_j\oplus\mathbf{t}_{j,0}=r_j\cdot\mathbf{s}$.)

4. For $j\in[m]$, $\mathcal{S}$ sends $y_{j,0}=x_{j,0}\oplus H(j,\mathbf{q}_j)$ and $y_{j,1}=x_{j,1}\oplus H(j,\mathbf{q}_j\oplus\mathbf{s})$.

5. For $j\in[m]$, $\mathcal{R}$ recovers $z_j=y_{j,r_j}\oplus H(j,\mathbf{t}_{j,0})$.

Efficiency. The protocol makes a single call to OT_m^k. As described in Appendix B.3, the cost of OT_m^k is the cost of OT_k^k (which is independent of m) plus a generation of $2k$ pseudorandom strings each of length m. Other than this call to OT_m^k, each party evaluates at most $2m$ times (an implementation of) a random oracle. It is easy to see that the total communication cost of OT_ℓ^m is the communication cost of implementing OT_m^k plus $2m\ell$ bits transferred between $\mathcal{S}$ and $\mathcal{R}$ in Step 4. Thus we conclude that the communication cost of OT_ℓ^m is $O(m(k+\ell))$ bits. (The exact communication cost is $2mk + 2m\ell$ bits.) Note that the total computational cost of the protocol is proportional to its communication cost.

3.3 Walsh-Hadamard (WH) Codes

For $\alpha \in \{0,1\}^q$, let $\mathrm{WH}(\alpha) = (\langle \alpha, x\rangle)_{x\in\{0,1\}^q}$, where the inner product between the two vectors is taken modulo 2. That is, $\mathrm{WH}(\alpha)$, also known as the Walsh-Hadamard encoding of α, is the 2^q-bit string consisting of inner products of each q-bit string with α. For each k, Walsh-Hadamard codes, denoted by $\mathcal{C}_{\mathrm{WH}}^k$, are simply defined as the set $\{\mathrm{WH}(\alpha)\}_{\alpha\in\{0,1\}^{\log k}}$. Note that $\mathcal{C}_{\mathrm{WH}}^k$contains k strings (or, codewords) each of length k bits. In our constructions, we will use the well-known fact that the relative distance of $\mathcal{C}_{\mathrm{WH}}^k$ is 1/2 when k is a power of 2.

4 Extending 1-out-of-n OT

Recall, k is a security parameter. We present a natural generalization of 1-out-of-2 OT extension protocol given in [IKNP03]. We consider 1-out-of-n OT for any $n \le k$.[2] First, recall that it is easy to construct a 1-out-of-n OT protocol from $O(\log n)$ instances of a 1-out-of-2 OT protocol in the semi-honest setting. (See Appendix B.1 for an explicit construction.) The communication cost of m instances of 1-out-of-n OT on ℓ-bit strings would be the cost of $\mathrm{OT}_k^{m\log n}$ plus the cost required to transmit at most mn masked secrets each of length ℓ. Thus, the communication cost of obtaining m instances of 1-out-of-n OT on ℓ-bit strings is at most $O(m(k\log n + n\ell))$ bits. Further, its computational cost is proportional to the communication cost.

Our main contribution, formally presented in this section, is showing how to generalize IKNP's technique to directly obtain (i.e., without going via a construction for 1-out-of-2 OT) an extension protocol for 1-out-of-n OT when $n \le k$. For the same security parameter and the same size of setup matrix at IKNP, the concrete security of our construction corresponds to that provided by security parameter $k_{\mathrm{IKNP}} \approx k/2$ (cf. Appendix A). If exactly same concrete security as IKNP is desired, this can be achieved by setting our security parameter $k \approx 2k_{\mathrm{IKNP}}$, which results in a multiplicative factor 2 overhead compared to IKNP. However, because we do 1-out-of-n OT at this cost, our construction will still result in asymptotic and concrete performance improvement of 1-out-of-n OT.

Let $\binom{n}{1}$-OT_ℓ^m denote m instances of 1-out-of-n OT on ℓ-bit strings. As in [IKNP03], we will reduce $\binom{n}{1}$-OT_ℓ^m to OT_m^k(which can be trivially efficiently reduced to OT_k^k, cf. Appendix B.3). As the [IKNP03] basic protocol, our protocol is secure against a malicious sender and semi-honest receiver. Our protocol will use Walsh-Hadamard codes, denoted by $\mathcal{C}_{\mathrm{WH}}^k = (\mathbf{c}_0, \ldots, \mathbf{c}_{k-1})$.

We now describe our protocol that realizes $\binom{n}{1}$-OT_ℓ^m given ideal access to OT_m^k.

Construction 1 (1-out-of-n OT Extension)**.**
Input of $\mathcal{S}$: *m tuples $(x_{j,0}, \ldots, x_{j,n-1})$ of ℓ-bit strings,* $1 \le j \le m$.

[2]We discuss how to extend 1-out-of-n OT for $n = \mathrm{poly}(k)$ in Section 5.3.

Input of $\mathcal{R}$: *m selection integers $\mathbf{r} = (r_1, \ldots, r_m)$ such that $0 \le r_j < n$ for $1 \le j \le m$.*
Common Input: *a security parameter k such that $k \ge n$, and Walsh-Hadamard codes $\mathcal{C}^k_{\text{WH}} = (\mathbf{c}_0, \ldots, \mathbf{c}_{k-1})$.*
Oracle: *a random oracle $H : [m] \times \{0,1\}^k \to \{0,1\}^\ell$.*
Cryptographic Primitive: *an ideal OT^k_m primitive.*

1. *$\mathcal{S}$ chooses $\mathbf{s} \leftarrow \{0,1\}^k$ at random. Let s_i denote the i-th bit of $\mathbf{s}$.*
2. *$\mathcal{R}$ forms $m \times k$ matrices T_0, T_1 in the following way:*
   - *Choose $\mathbf{t}_{j,0}, \mathbf{t}_{j,1} \leftarrow \{0,1\}^k$ at random such that $\mathbf{t}_{j,0} \oplus \mathbf{t}_{j,1} = \mathbf{c}_{r_j}$.*

   Let $\mathbf{t}^i_0, \mathbf{t}^i_1$ denote the i-th column of matrices T_0, T_1 respectively.
3. *$\mathcal{S}$ and $\mathcal{R}$ interact with OT^k_m in the following way:*
   - *$\mathcal{S}$ acts as* receiver *with input $\{s_i\}_{i \in [k]}$.*
   - *$\mathcal{R}$ acts as* sender *with input $\{\mathbf{t}^i_0, \mathbf{t}^i_1\}_{i \in [k]}$.*
   - *$\mathcal{S}$ receives output $\{\mathbf{q}^i\}_{i \in [k]}$.*

   $\mathcal{S}$ forms $m \times k$ matrix Q such that the i-th column of Q is the vector $\mathbf{q}^i$. (Note $\mathbf{q}^i = \mathbf{t}^i_{s_i}$.) Let $\mathbf{q}_j$ denote the j-th row of Q. (Note $\mathbf{q}_j = ((\mathbf{t}_{j,0} \oplus \mathbf{t}_{j,1}) \odot \mathbf{s}) \oplus \mathbf{t}_{j,0}$. Simplifying, $\mathbf{q}_j \oplus \mathbf{t}_{j,0} = \mathbf{c}_{r_j} \odot \mathbf{s}$.)
4. *For $j \in [m]$ and for every $0 \le r < n$, $\mathcal{S}$ sends $y_{j,r} = x_{j,r} \oplus H(j, \mathbf{q}_j \oplus (\mathbf{c}_r \odot \mathbf{s}))$.*
5. *For $j \in [m]$, $\mathcal{R}$ recovers $z_j = y_{j,r_j} \oplus H(j, \mathbf{t}_{j,0})$.*

This concludes the description of the protocol. It is easy to verify that the protocol's outputs are correct (i.e., $z_j = x_{j,r_j}$) when both parties follow the protocol.

Efficiency. The protocol makes a single call to OT^k_m. As described in Appendix B.3, the cost of OT^k_m is the cost of OT^k_k (which is independent of m) plus a generation of $2k$ pseudorandom strings each of length m. Other than this call to OT^k_m, each party evaluates at most mn times (an implementation of) a random oracle. It is easy to see that the total communication cost of OT^m_ℓ is the communication cost of implementing OT^k_m plus $mn\ell$ bits transferred between $\mathcal{S}$ and $\mathcal{R}$ in Step 4. Thus we conclude that the communication cost of OT^m_ℓ is $O(m(k + n\ell))$ bits. Note that the total computational cost of the protocol is proportional to its communication cost. Recall that $n \le k$, and thus when $\ell = 1$, the asymptotic cost of our $\binom{n}{1}$-OT^m_ℓ protocol is $O(mk)$ which is the same as the asymptotic cost of Ishai et al.'s OT^m_ℓ protocol described in Section 3.2. In terms of concrete performance, as mentioned above, we need to use a security parameter $k \approx 2k_{\text{IKNP}}$, resulting in a factor 2 overhead compared to IKNP's OT^m_ℓ execution. Because we are performing the more powerful $\binom{n}{1}$-OT^m_ℓ, this corresponds to asymptotic (and concrete!) performance improvement.

Theorem 1. *Construction 1 is a secure protocol for evaluating $\binom{n}{1}$-OT^m_ℓ in the semi-honest model.*

Proof of security of Theorem 1 is presented in Appendix A for the lack of space. □

Remarks. In Construction 1, one can replace $\mathcal{C}^k_{\text{WH}}$ with an encoding map $\mathsf{enc} : \{0,1\}^{\log n} \to \{0,1\}^k$ that has the property that for $r, r' \in \{0,1\}^{\log n}$ with $r \ne r'$, the Hamming distance between $\mathsf{enc}(r)$ and $\mathsf{enc}(r')$ is at least $\Omega(k)$. It is instructive to see that when $n = 2$ and when enc is the k-bit *repetition* encoding of the input bit, i.e., $\mathsf{enc}(r) = (r, \ldots, r) \in \{0,1\}^k$, then we get exactly the IKNP construction. Note that for $r \neq r'$, the Hamming distance between $\mathsf{enc}(r)$ and $\mathsf{enc}(r')$ is exactly k. As we saw in Construction 1, using the encoding map $\mathsf{enc}(r) = \mathbf{c}_r$, where $\mathbf{c}_r$ is the r-th Walsh-Hadamard codeword, gives us an $\log k$ efficiency improvement. Since the Walsh-Hadamard code is a low-rate code, the maximum value of n is restricted to be less than or equal to k. A natural question that arises is whether a code with a better rate enables us to remove this restriction. Indeed, in Section 5.3, by using more sophisticated codes (cf. Claim 2) we show an improvement in the (offline) communication complexity of 1-out-of-n OT extension for arbitrary $n = \mathrm{poly}(k)$.

5 Resulting Efficiency Improvements

We evaluate performance improvements of Construction 1, and corresponding two- and multi-party SFE improvements. Recall that in the semi-honest model, a single instance of 1-out-of-n OT may be used to generate $\log n$ instances of 1-out-of-2 OT over slightly shorter strings but with no additional cost (see Appendix B.2 for an explicit construction). More precisely, the cost of OT_ℓ^m is exactly equal to the cost of $\binom{n}{1}$-$\mathrm{OT}_{\ell \log n}^{m/\log n}$. This observation will allow us to leverage our efficient construction of $\binom{n}{1}$-OT_ℓ^m to obtain improved efficiency for 1-out-of-2 OT, and consequently for secure computation.

5.1 Efficiency Improvements for 1-out-of-2 OT

In this section, we demonstrate a $\log k$ asymptotic improvement in the efficiency of 1-out-of-2 OT when sender's secrets are just bits (i.e., length of sender's secrets, $\ell = 1$). As observed previously, we do this by constructing 1-out-of-2 OTs via 1-out-of-n OTs (cf. Appendix B.2).

Recall that the cost of our $\binom{n}{1}$-OT_ℓ^m protocol described in Section 4 is $O(m(k + n\ell))$. Using the fact that the cost of OT_ℓ^m is exactly equal to the cost of $\binom{n}{1}$-$\mathrm{OT}_{\ell \log n}^{m/\log n}$, we conclude that OT_ℓ^m may be reduced to OT_k^k while incurring an additional cost at most $O((m/\log n) \cdot (k + n\ell \log n))$. By choosing n such that $n \log n = k/\ell$, we see that this additional cost is asymptotically $O(mk/\log(k/\ell))$. In summary, we have shown a reduction from OT_ℓ^m to OT_k^k with cost $O(mk/\log(k/\ell))$.

Contrast our result above with the result of [IKNP03], where the cost of the reduction from OT_ℓ^m to OT_k^k was $O(m(k + \ell))$. Observe that for the important case when $\ell = 1$, our construction offers a logarithmic factor improvement in the efficiency of the reduction.

As noted in Section 4, to achieve concrete security equal to that of IKNP, we need a security parameter approximately twice theirs, which results in a factor 2 overhead of our protocol. Even with this efficiency loss we have both asymptotic and concrete performance advantage over IKNP.

Concrete Efficiency. We perform an exact calculation of the communication cost of our OT_ℓ^m construction. We begin with a concrete cost analysis of $\binom{n}{1}$-OT_ℓ^m. Recall that the exact cost of reduction from OT_m^k to OT_k^k is $2mk$. Then, in Step 4, $\mathcal{S}$ transmits $mn\ell$ bits to $\mathcal{R}$. Thus, the concrete cost of $\binom{n}{1}$-OT_ℓ^m is $m(2k + n\ell)$. Using the fact that the cost of our OT_ℓ^m is exactly equal to the cost of $\binom{n}{1}$-$\mathrm{OT}_{\ell \log n}^{m/\log n}$, we conclude that OT_ℓ^m may be reduced to OT_k^k with cost $(m/\log n) \cdot (2k + n\ell \log n)$ bits. The minimum cost can then be obtained by choosing a suitable value of n.

In contrast, the concrete communication cost of IKNP's construction of OT_ℓ^m is $2m(k + \ell)$ bits. As described earlier, there's a small gap between the security guarantees between our consruction and IKNP's. We take that into account in our cost calculation, and present the results in Table 1. We refer the reader to Appendix C and D (in particular to Tables 3, 4, 5) for more details on the concrete efficiency of (some variants of) our constructions.

level of security	our cost	IKNP cost
50	74	102
112	130	226
238	227	478

Table 1: Comparison of (amortized) communication cost (measured in bits) of 1-out-of-2 bit OT for a given security level. The costs are computed assuming parties are semi-honest. The performance improvement ratio betwen our work and IKNP represents the resulting improvement factor for MPC protocols.

level of security	our cost per gate	[PSSW09] cost per gate	[KK12] cost per gate
50	148	100	66
112	260	224	112
238	454	476	196

Table 2: Comparison of (amortized) communication cost (measured in bits) per gate of the circuit for various semi-honest secure two-party protocols. We note that protocols of [KK12] do not extend to multi-party setting, while ours do.

5.2 Efficiency Improvements for Secure Computation

In this section, we will discuss applications of our OT_ℓ^m protocol to secure two-party and multi-party computation. As pointed out in the Introduction, efficient OT forms a criticial component of secure computation protocols, and improvements in the efficiency of OT translates to an improvement in the efficiency of secure computation protocols built on top of OT.

In the previous section, we saw how our construction asymptotically outperforms the extension protocol of [IKNP03] by a factor of $O(\log(k/\ell))$. Clearly, this improvement factor is maximized when $\ell = 1$, i.e., for 1-bit OT. Thus, our construction has maximum benefit for secure computation protocols that extensively rely on 1-bit OTs. One such example is the well known GMW protocol [GMW87] where each AND gate of the circuit is evaluated using (two invocations of) 1-bit OTs (and negligible additional cost). Until now, efficient implementations of the GMW protocol in the semi-honest setting (e.g., [CHK$^+$12]) relied on the OT extension protocol of [IKNP03]. Because OT costs dominate the protocol costs, simply by using our extension protocol (instead of [IKNP03]), the semi-honest GMW protocol will enjoy an asymptotic $\log k$ efficiency improvement (and improvement in concrete terms as well).

Secure Two-Party Computation. The concrete improvements for the specific case of two-party computation are shown in Table 2. From the table, it is evident that our protocol begins to outperform state-of-the-art constant round protocols (e.g., [PSSW09]) for reasonable levels of security. However, for practical values of the security parameter, it performs worse, in concrete terms, when compared to [KK12], a non-constant round protocol that generalizes both Yao garbled circuits and GMW. We point out that the approach of [KK12] can be viewed as somewhat related to ours (but more narrow; in particular, it is not applicable to multiparty comptuation), and the communication cost of our protocol is asymptotically the same as theirs. Furthermore, while our improvements are independent of the topology of the circuit being evaluated, this is not the case in [KK12] where in the calculations it is assumed that the width of the circuit is constant.

Secure Multi-Party Computation. Today, practical protocols for secure *multi-party* computation are based on the GMW approach (e.g., [NNOB12, CHK$^+$12]).[3] GMW-based secure computation protocols for t parties, in the semi-honest setting, operate in almost the same way as in the two-party case except that now parties compute pairwise OTs (more precisely, a total of $2t^2$ OTs) to securely evaluate each AND gate. That is, for each AND gate of the circuit parties evaluate a total of $2t^2$ 1-bit OTs (with negligible additional cost). Therefore, simply by using our extension protocol (instead of [IKNP03]), we will improve the asymptotic complexity by a $\log k$ factor. Concrete improvements in this setting are the same as those found in Table 1. Specifically, for "50-bit security" we obtain an improvement of $102/74 = 1.378$ in the communication cost. Similarly, we obtain an improvement factor of > 2 for "238-bit security".

5.3 Efficiency Improvements for 1-out-of-n OT

Recall that the cost of extending 1-out-of-n OT from [IKNP03] is $O(m(k \log n + n\ell))$ bits. Our main construction presented in Section 4 reduces the cost of 1-out-of-n OT extension to $O(m(k + n\ell))$. However, this improvement holds only when $n \leq k$. In this section, we show how to modify Construction 1 to support $n = \text{poly}(k)$. In the resulting protocol, the (offline) communication cost of the generating 1-out-of-n OT correlations will be $O(mk)$ bits, i.e., completely independent of n. (We demonstrate how to adapt our protocol to the preprocessing model in Appendix E.) This improves over the best known offline communication complexity (which was $O(mk \log n)$ bits).

The total complexity (i.e., both online and offline) of our construction will asymptotically outperform existing constructions only for $n \leq ck$ where c is an arbitrary constant. For $n = \omega(k)$, the online cost of our protocol $O(mn\ell)$ dominates the total cost, but is still as efficient as existing constructions.

The main idea of our construction is to replace $\mathcal{C}^k_{\text{WH}}$ with a code from a family of linear error correcting codes with the following special properties. (Our claim below is taken verbatim from [IKOS09].)

Claim 2 ([IKOS09, CC06, GS96])**.** *There exists a finite field $\mathbb{F}$ of characteristic 2 and an efficiently constructible family of linear error-correcting codes $C_K : \mathbb{F}^K \to \mathbb{F}^{N_K}$ with the following properties: (1) $N_K = O(K)$; (2) The dual distance of C_K is $\delta_K = \Omega(K)$; (3) The linear code C'_K spanned by all pointwise-products of pairs of codewords in C_K has minimal distance $\Delta_K = \Omega(K)$ and supports efficient decoding of up to $\mu_K = \Omega(K)$ errors. (The pointwise product of $(c_1, \ldots, c_N)$ and $(c'_1, \ldots, c'_N)$ is $(c_1c'_1, \ldots, c_Nc'_N)$.)*

The last property implies that C_K also has minimal distance $d_K = \Omega(K)$.

Setting $N_K = k$ and $K \geq \log n$ is enough to provide the desired improvements stated above. The security level provided by this construction will be $\log(2^{d_K}/n^2) = \Omega(k)$ for n polynomial in k.

References

[AIK04] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *45th Annual Symposium on Foundations of Computer Science*, pages 166–175. IEEE Computer Society Press, October 2004.

[3] Yao GC-based approach does not seem to map naturally into the multiparty setting. This is true even for the three party semi-honest setting. A more complicated solution is possible [BMR90], but much less practical than GMW-based approaches [BDNP08, CHK$^+$12].

[AJLA$^+$12] Gilad Asharov, Abhishek Jain, Adriana Lopez-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold fhe. In *Advances in Cryptology – EUROCRYPT 2012*, Lecture Notes in Computer Science, pages 483–501. Springer, 2012.

[App12] Benny Applebaum. Pseudorandom generators with long stretch and low locality from random local one-way functions. In *44th Annual ACM Symposium on Theory of Computing*, pages 805–816. ACM Press, 2012.

[Bar86] David Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1. In *18th Annual ACM Symposium on Theory of Computing*, pages 1–5. ACM Press, May 1986.

[BCD$^+$09] Peter Bogetoft, Dan Lund Christensen, Ivan Damgαrd, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael I. Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In Roger Dingledine and Philippe Golle, editors, *FC 2009: 13th International Conference on Financial Cryptography and Data Security*, volume 5628 of *Lecture Notes in Computer Science*, pages 325–343. Springer, February 2009.

[BCR87] Gilles Brassard, Claude Crépeau, and Jean-Marc Robert. All-or-nothing disclosure of secrets. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 234–238. Springer, August 1987.

[BDNP08] Assaf Ben-David, Noam Nisan, and Benny Pinkas. FairplayMP: a system for secure multi-party computation. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 08: 15th Conference on Computer and Communications Security*, pages 257–266. ACM Press, October 2008.

[Bea92] Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432. Springer, August 1992.

[Bea95] Donald Beaver. Precomputing oblivious transfer. In Don Coppersmith, editor, *Advances in Cryptology – CRYPTO'95*, volume 963 of *Lecture Notes in Computer Science*, pages 97–109. Springer, August 1995.

[Bea96] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th Annual ACM Symposium on Theory of Computing*, pages 479–488. ACM Press, May 1996.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols. In *22nd Annual ACM Symposium on Theory of Computing*, pages 503–513. ACM Press, May 1990.

[CC06] Hao Chen and Ronald Cramer. Algebraic geometric secret sharing schemes and secure multi-party computations over small fields. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 521–536. Springer, August 2006.

[CDI05] Ronald Cramer, Ivan Damgαrd, and Yuval Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 342–362. Springer, February 2005.

[CFIK03] Ronald Cramer, Serge Fehr, Yuval Ishai, and Eyal Kushilevitz. Efficient multi-party computation over rings. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 596–613. Springer, May 2003.

[CHK$^+$12] Seung Geol Choi, Kyung-Wook Hwang, Jonathan Katz, Tal Malkin, and Dan Rubenstein. Secure multi-party computation of boolean circuits with applications to privacy in on-line marketplaces. In *Topics in Cryptology – CT-RSA 2012*, Lecture Notes in Computer Science, pages 416–432. Springer, 2012.

[Cle90] Richard Cleve. Towards optimal simulations of formulas by bounded-width programs. In *22nd Annual ACM Symposium on Theory of Computing*, pages 271–277. ACM Press, May 1990.

[EGL85] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. C. ACM, 28:637-647, 1985.

[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 Annual ACM Symposium on Theory of Computing*, pages 218–229. ACM Press, May 1987.

[GS96] A. Garcia and H. Stichtenoth. On the asymptotic behavior of some towers of function fields over finite fields. Journal of Number Theory, 61(2):248-273, 1996.

[GV88] Oded Goldreich and Ronen Vainish. How to solve any protocol problem - an efficiency improvement. In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, volume 293 of *Lecture Notes in Computer Science*, pages 73–86. Springer, August 1988.

[HEKM04] Yan Huang, David Evans, Jonathan Katz, and Lior Malka. Faster secure two-party computation using garbled circuits. USENIX Security Symposium, 2004.

[HIKN08] Danny Harnik, Yuval Ishai, Eyal Kushilevitz, and Jesper Buus Nielsen. OT-combiners via secure computation. In Ran Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 393–411. Springer, March 2008.

[HKE12] Yan Huang, Jonathan Katz, and David Evans. Quid-pro-quo-tocols: Strengthening semi-honest protocols with dual execution. IEEE Symposium on Security and Privacy, 2012.

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

[IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan Eidenbenz, and Ricardo Conejo, editors, *ICALP 2002: 29th International Colloquium on Automata, Languages and Programming*, volume 2380 of *Lecture Notes in Computer Science*, pages 244–256. Springer, July 2002.

[IKM$^+$13] Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, Claudio Orlandi, and Anat Paskin-Cherniavsky. On the power of correlated randomness in secure computation. page To Appear at TCC, 2013.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161. Springer, August 2003.

[IKOS08] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 433–442. ACM Press, May 2008.

[IKOS09] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Extracting correlations. In *50th Annual Symposium on Foundations of Computer Science*, pages 261–270. IEEE Computer Society Press, 2009.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 572–591. Springer, August 2008.

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

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31. ACM, 1988.

[KK12] Vladimir Kolesnikov and Ranjit Kumaresan. Improved secure two-party computation via information-theoretic garbled circuits. In *8th Conference on Security and Cryptography for Networks*, pages ??–??, 2012.

[Kol05] Vladimir Kolesnikov. Gate evaluation secret sharing and secure one-round two-party computation. In Bimal K. Roy, editor, *Advances in Cryptology – ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 136–155. Springer, December 2005.

[KS08] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In Luca Aceto, Ivan Damgαrd, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008: 35th International Colloquium on Automata, Languages and Programming, Part II*, volume 5126 of *Lecture Notes in Computer Science*, pages 486–498. Springer, July 2008.

[KSS09] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS 09: 8th International Conference on Cryptology and Network Security*, volume 5888 of *Lecture Notes in Computer Science*, pages 1–20. Springer, December 2009.

[Lip08] Helger Lipmaa. New communication-efficient oblivious transfer protocols based on pairings. In Tzong-Chen Wu, Chin-Laung Lei, Vincent Rijmen, and Der-Tsai Lee, editors, *ISC 2008: 11th International Conference on Information Security*, volume 5222 of *Lecture Notes in Computer Science*, pages 441–454. Springer, September 2008.

[LOP11] Yehuda Lindell, Eli Oxman, and Benny Pinkas. The IPS compiler: Optimizations, variants and concrete efficiency. In *Advances in Cryptology – CRYPTO 2011*, Lecture Notes in Computer Science, pages 259–276. Springer, August 2011.

[LP07] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 52–78. Springer, May 2007.

[LZ13] Yehuda Lindell and Hila Zarosim. On the feasibility of extending oblivious transfer. page To Appear at TCC, 2013.

[MNPS04] Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay - secure two-party computation system. USENIX Security Symposium, 2004.

[Nie07] Jesper Buus Nielsen. Extending oblivious transfers efficiently - how to get robustness almost for free. *IACR Cryptology ePrint Archive*, 2007:215, 2007.

[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*, volume 7417 of *Lecture Notes in Computer Science*, pages 681–700. Springer, 2012.

[NP05] Moni Naor and Benny Pinkas. Computationally secure oblivious transfer. *Journal of Cryptology*, 18(1):1–35, January 2005.

[PSSW09] Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure two-party computation is practical. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 250–267. Springer, December 2009.

[Rab81] Michael O. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.

[SS11] Abhi Shelat and Chih-Hao Shen. Two-output secure computation with malicious adversaries. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 386–405. Springer, May 2011.

[SZ12] Thomas Schneider and Michael Zohner. Private communication. 2012.

[Yao86] Andrew Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167. IEEE Computer Society Press, October 1986.

A Proof of Theorem 1

The proof of security of our construction closely follows that of the basic IKNP protocol described in Section 3.2. As in [IKNP03], we prove that the protocol described above is secure against a malicious sender and against a semi-honest receiver. We will demonstrate a perfect simulator for any malicious sender $\mathcal{S}^*$ and a statistical simulator for (semi-honest) $\mathcal{R}$. In the latter case, the output of the ideal process involving the simulator is indistinguishable from that of the real process even if the distinguisher is allowed to adaptively make $2^{k/2}/k^{\omega(1)}$ additional calls to H. (Contrast this with [IKNP03], where the distinguisher is allowed to adaptively make $2^k/k^{\omega(1)}$ calls to H.)

We let TP denote the trusted party for the OT_ℓ^m functionality in the ideal function evaluation process.

Simulating $\mathcal{S}^*$. It is easy to argue that the output of an arbitrary $\mathcal{S}^*$ can be perfectly simulated. Indeed, all $\mathcal{S}^*$ views throughout the protocol is a k-tuple of uniformly random and independent vectors, received from the OT_m^k primitive in Step 3. This guarantees that the receiver's selections remain perfectly private. A simulator for a malicious sender $\mathcal{S}^*$ may proceed as follows:

- Run $\mathcal{S}^*$ with a uniformly chosen random input ρ. Let $\mathbf{s}^*$ be the input $\mathcal{S}^*$ sends to OT_m^k primitive in Step 3. Generate a random $m \times k$ matrix Q, and feed $\mathcal{S}^*$ with the columns of Q as the output from the OT_m^k primitive.
- Let $(y_{j,0}^*, \ldots, y_{j,n-1}^*)$ be the messages sent by $\mathcal{S}^*$ in Step 4. Call TP with inputs $(x_{j,0}^*, \ldots, x_{j,n-1}^*)$ for $1 \le j \le m$ where $x_{j,r}^* = y_{j,r}^* \oplus H(j, \mathbf{q}_j \oplus (\mathbf{c}_r \odot \mathbf{s}^*))$.
- Output whatever $\mathcal{S}^*$ outputs.

Correctness: It is easy to verify that the joint distribution of $\rho, \mathbf{s}^*, Q$, the values $(y_{j,0}^*, \ldots, y_{j,n-1}^*)$, and all values of H queried by $\mathcal{S}^*$ in the ideal process is identical to the corresponding distribution in the real process. It remains to show that, conditioned on all the above values, the receiver's outputs x_{j,r_j}^* in the ideal process are distributed identically to these outputs in the real process. This follows from the way the output of $\mathcal{R}$ is defined in the Step 5 of the protocol and from the fact that (in the real process) $\mathbf{t}_j^* = \mathbf{q}_j \oplus (\mathbf{c}_{r_j} \odot \mathbf{s}^*))$. Note that the above simulation remains perfect even if the distinguisher makes an arbitrary number of calls to H. Thus we have:

Claim 3. *The protocol is perfectly secure with respect to an arbitrary sender.*

Simulating $\mathcal{R}$. The semi-honest receiver $\mathcal{R}$ can be simulated as follows:

- Call TP with input $\mathbf{r}$. Let z_j denote the j-th output received from TP.
- Run the protocol between $\mathcal{R}$ and $\mathcal{S}$, substituting the values z_j for the known input x_{j,r_j} of $\mathcal{S}$ and the default value 0^ℓ for the unknown inputs $\{x_{j,r}\}_{r \neq r_j}$. Output the entire view of $\mathcal{R}$.

In our protocol, the pads used for masking the sender's inputs in the j-th iteration are $\{H(j, \mathbf{q}_j \oplus (\mathbf{c}_r \odot \mathbf{s}))\}_{0 \le r < n}$. Since $\mathbf{q}_j = \mathbf{t}_{j,0} \oplus (\mathbf{c}_{r_j} \odot \mathbf{s})$, we see that the pads are exactly the set $\{H(j, \mathbf{t}_{j,0} \oplus ((\mathbf{c}_r \oplus \mathbf{c}_{r_j}) \odot \mathbf{s}))\}_{0 \le r < n}$. (Contrast this with IKNP, where the pads used for masking the sender's inputs hidden input in the j-th iteration are $\{H(j, \mathbf{t}_{j,0}), H(j, \mathbf{t}_{j,0} \oplus \mathbf{s})\}$.)

Suppose $\mathbf{0}$ represents the all-zero vector. It is clear from the discussion above, and the description of the simulator that, conditioned on the event $\mathbf{0} \notin \{(\mathbf{c}_r \oplus \mathbf{c}_{r'}) \odot \mathbf{s}\}_{0 \le r < r' < n}$, the simulated view is distributed *identically* to the receiver's view in the real process. Indeed, if

$\mathbf{0} \notin \{(\mathbf{c}_r \oplus \mathbf{c}_{r'}) \odot \mathbf{s}\}_{0\le r<r'<n}$, the values of H used for masking the unknown inputs $\{x_{j,r}\}_{r\neq r_j}$ are uniformly random and independent of the receiver's view and of each other. Since $\mathcal{C}^k_{\mathrm{WH}}$ is a code with relative distance $1/2$, we have that for any $r \neq r'$, the Hamming weight of $\mathbf{c}_r \oplus \mathbf{c}_{r'}$ is at least $k/2$. Thus by means of a simple union bound we conclude that the simulator's output is (at least) $n^2 2^{-k/2}$-close to the real view.[4] However, to make a meaningful security statement in the random oracle model, we must also allow the distinguisher to (adaptively) make additional calls to H, where the answers to these calls are provided by the same H that was used in the process (real or ideal) generating the distinguisher's input.

Now, if the distinguisher can "guess" the oracle query used in the real process to mask secret $x_{j,r}$ for some $r \neq r_j$ which $\mathcal{R}$ is not supposed to learn, then it clearly wins (i.e., it can easily distinguish between any two values for this unknown secret). On the other hand, as long as it does not guess such a critical query, the masks remain random and independent given its view, and so indistinguishability is maintained. The crucial observation is that from the distinguisher's point of view, each of these $(n-1)\cdot m$ offending queries is (individually) distributed uniformly at random over a domain of size (at least) $2^{k/2}$. This follows from the fact that (1) for any $0 \le r < r' < n$, the Hamming weight of $\mathbf{c}_r \oplus \mathbf{c}_{r'}$ is at least $k/2$, and since $\mathbf{s}$ is picked at random from $\{0,1\}^k$, the value $(\mathbf{c}_r \oplus \mathbf{c}_{r'}) \odot \mathbf{s}$ is uniformly distributed over a domain of size at least $2^{k/2}$, and (2) the distinguisher has no information about $\mathbf{s}$ as long as it makes no offending query. Hence, the distinguisher can only win the above game with negligible probability. This is formalized by the following lemma.

Lemma 4. *Any distinguisher D which makes at most t calls to H can have at most a $(t+n^2)\cdot 2^{-k/2}$-advantage in distinguishing between the output of the real process and that of the ideal process.*

Proof. Define the *extended* real (resp., ideal) process to be the real (resp., ideal) process followed by the invocation of D on the output of the process. The output of the extended process includes the output of the original process along with the transcript of the oracle calls made by D. For each of the extended processes, define an *offending query* to be a call to H on some input in the set $\{(j, \mathbf{t}_j \oplus ((\mathbf{c}_r \oplus \mathbf{c}_{r'}) \odot \mathbf{s}))\}_{0\le r<r'<n}$ for some $1 \le j \le m$, and define B to be the (bad) event that an offending query is ever made by either $\mathcal{R}$ or D. (Actually, we need not iterate over r and r'... just over r' is enough, since r is fixed for a semihonest receiver.) It is easy to verify that, as long as no offending query is made, the outputs of the two extended processes are perfectly indistinguishable from one another. Thus, the event B has the same probability in both extended processes, and the outputs of the two extended processes are identically distributed conditioned on B not occurring. It remains to show that $\Pr[B] \le (t+n^2)\cdot 2^{-k/2}$. This, in turn, follows by noting that: (1) $\mathcal{R}$ makes an offending query only if $\mathbf{0} \in \{(\mathbf{c}_r \oplus \mathbf{c}_{r'}) \odot \mathbf{s}\}_{0\le r<r'<n}$, and (2) as long as no offending query is made, D's view is completely independent of the value of $\mathbf{s}$. □

Thus, we have:

Claim 5. *As long as $n \le k$ and $m = 2^{o(k)}$, the protocol is statistically secure with respect to a semi-honest receiver and a polynomial-time distinguisher having access to the random oracle.*

B OT Reductions

For completeness, we present several simple reductions among the OT primitives.

[4]We remark that it may be possible to obtain a more precise bound on the "closeness" between the ideal and real processes by a careful analysis of the codewords in $\mathcal{C}^k_{\mathrm{WH}}$.

B.1 Reducing 1-out-of-n OT to 1-out-of-2 OT

Here we show a simple idea [NP05] how to implement a 1-out-of-n OT, given a $\log n$ number of 1-out-of-2 OTs on *strings* of length k in the semi-honest model.

$\mathcal{S}$ and $\mathcal{R}$ perform $\log n$ 1-out-of-2 OTs, where $\mathcal{S}$'s input consists of $2\log n$ random k-bit strings, and $\mathcal{R}$'s input consists of $\log n$ bits representing its selection. $\mathcal{S}$ then generates and sends to $\mathcal{R}$ the n inputs, each encrypted with the RO hash (or, applying PRF to the k-bit strings [NP05]) of the corresponding sequence of OT-transferred secrets. Implementing these $m\log n$ instances of k-bit string OTs using IKNP would cost $(2m\log n)\cdot(k+k) = 4mk\log n$ bits. In the final step, the encryptions for each of n strings of length ℓ would cost a total of $mn\ell$ bits. Thus, the total cost of $\binom{n}{1}\text{-OT}_\ell^m$ based on IKNP is $(4mk\log n + mn\ell)$ bits.

Contrast the above with our constructions in Section 4 where the total cost of $\binom{n}{1}\text{-OT}_\ell^m$ is $m(2k+n\ell)$ bits. It is easy to see that for $k=n$ and for $\ell=1$, we improve efficiency by a factor $(4\log k+1)/3$. For $k=128$, this corresponds to a factor 9.67 improvement in the communication complexity.

B.2 Reducing OT_ℓ^m to $\binom{n}{1}\text{-OT}_{\ell\log n}^{m/\log n}$

$\log n$ 1-out-of-2 1-bit OTs can be obtained from a 1-out-of-n $\log n$-bit OT as follows. The sender $\mathcal{S}$ from his input of $\log n$ pairs of bits generates n $\log n$-bit secrets, each corresponding to a $\log n$-bit long selection string. Then $\mathcal{S}$ and $\mathcal{R}$ execute 1-out-of-n $\log n$-bit OT. $\mathcal{R}$ interprets his received output as the bits received in $\log n$ 1-out-of-2 OTs.

B.3 Reducing OT_m^p to OT_k^p

Oblivious transfer of long strings can be efficiently reduced to oblivious transfer of shorter strings (of length at least equal to security parameter) using any pseudorandom generator. The simple idea is use OT_k^p to send one of two short random strings, and use them as keys for encryption of the two longer strings sent later. We describe the reduction below.

Input of $\mathcal{S}$: p pairs $(x_{i,0}, x_{i,1})$ of m-bit strings, $1 \le i \le p$.
Input of $\mathcal{R}$: m selection bits $\mathbf{r} = (r_1, \dots, r_p)$.
Common Input: a security parameter k.
Oracle: a PRG $G : \{0,1\}^k \to \{0,1\}^m$.
Cryptographic Primitive: an ideal OT_k^p primitive.

- $\mathcal{S}$ initializes n pairs of random k-bit seeds $(u_{i,0}, u_{i,1})$.
- The parties invoke the OT_k^p primitive, where $\mathcal{S}$ acts as a sender with inputs $(u_{i,0}, u_{i,1})$ for $1 \le i \le n$, and $\mathcal{R}$ as a receiver with input $\mathbf{r}$.
- For $1 \le i \le n$, $\mathcal{S}$ sends $(y_{i,0}, y_{i,1})$, where $y_{i,b} = x_{i,b} \oplus G(u_{i,b})$.
- For $1 \le i \le n$, $\mathcal{R}$ outputs $z_i = y_{i,r_i} \oplus G(u_{i,r_i})$.

The security of this reduction is straightforward to prove. Its complexity overhead is $O(mp)$ bits obtained via $2p$ evaluations of a PRG that produces m bit outputs. We note that the use of the PRG G can be emulated using calls to the random oracle H.

k_{IKNP}	IKNP cost	our cost in terms of n	min n	our cost	improvement factor
20	42	$n + 8 + 80/\log n$	8	43	0.976744
30	62	$n + 8 + 120/\log n$	11	54	1.14815
40	82	$n + 8 + 160/\log n$	15	64	1.28125
50	102	$n + 8 + 200/\log n$	16	74	1.37838
60	122	$n + 8 + 240/\log n$	16	84	1.45238
80	162	$n + 8 + 320/\log n$	21	102	1.58824
100	202	$n + 8 + 400/\log n$	22	120	1.68333
120	242	$n + 8 + 480/\log n$	27	136	1.77941
160	322	$n + 8 + 640/\log n$	32	168	1.91667
200	402	$n + 8 + 800/\log n$	35	199	2.0201
240	482	$n + 8 + 960/\log n$	43	228	2.11404
300	602	$n + 8 + 1200/\log n$	48	271	2.2214
500	1002	$n + 8 + 2000/\log n$	66	405	2.47407
1000	2002	$n + 8 + 4000/\log n$	110	708	2.82768

Table 3: Comparison of (amortized) communication cost (measured in bits) of 1-out-of-2 bit OT for a given security level. The costs are computed assuming parties are semi-honest.

C Concrete Performance Analysis

We provide more details on how the calculations in Tables 1 and 2 were performed.
Cost of IKNP. In order to perform a reduction from OT_ℓ^m to OT_k^k, the communication cost of IKNP is given by $2mk + 2m\ell$. The amortized cost per instance is $2k + 2\ell$. For transmitting 1-bit secrets, we see that the amortized cost per instance is $2(k + 1)$. The security of their reduction is 2^{-k} (assuming that $m \gg k$, and that their reduction from OT_m^k to OT_k^k uses a random oracle instead of a PRG).
Cost of our protocol. The analysis is slightly more complicated in our protocol. First, note that the cost of our $\binom{n}{1}$-OT_ℓ^m protocol is $m(2k + n\ell)$. By using a communication preserving reduction from $\binom{n}{1}$-$\text{OT}_{\ell \log n}^{m/\log n}$ to OT_ℓ^m, we derive the cost of our protocol as $(m/\log n)(2k + n\ell \log n)$. The amortized cost per instance is therefore $n\ell + (2k/\log n)$. For transmitting 1-bit secrets, we see that the amortized cost per instance is $n + (2k/\log n)$. Under the same assumption as IKNP, the security of our reduction is $n^2 2^{-k/2}$.

In order to make a meaningful comparison of efficiency, we first choose values of k and n such that $n^2 2^{-k/2} = 2^{-k_{\text{IKNP}}}$, where k_{IKNP} is the security parameter used in IKNP. That is, we need to compare the cost functions of our protocol, i.e., $(n + (2k/\log n))$ with the IKNP protocol, i.e., $(2k_{\text{IKNP}} + 2)$ under the constraints $n^2 2^{-k/2} = 2^{-k_{\text{IKNP}}}$, and $n \le k$ (imposed by our use of Walsh-Hadamard codes).

Simplifying the constraint we have $2 \log n - (k/2) = -k_{\text{IKNP}}$, i.e., $k/2 = k_{\text{IKNP}} + 2 \log n$, i.e., $k = 2k_{\text{IKNP}} + 4 \log n$. Substituting for k in our cost function now yields $(n + (4k_{\text{IKNP}}/\log n) + 8)$. Thus, our task simply reduces to choosing the right value of n for which our cost function is minimized while satisfying $k = 2k_{\text{IKNP}} + 4 \log n \ge n$. The results are presented in Table 3.

D Optimizing the Reduction from $\binom{n}{1}$-OT_ℓ^m to OT_k^k

In our OT extension protocol, the OT_m^k primitive is reduced to OT_k^k. Further, the roles of $\mathcal{R}$ and $\mathcal{S}$ are reversed in our application of the reduction in our protocol. We provide an optimization that exploits this fact. This optimization was independently discovered by us and by Schneider and Zohner [SZ12].

The main idea of the optimization is that inside the OT extension protocol of IKNP (as well as our protocol) 1-out of-2 OT of very long (m-bit long) random-looking correlated strings is executed. We cut the communication almost in half by OT-sending a PRG seed used to generate the strings. In other words, we obtain efficiency improvements by employing pseudorandom additive sharing [CDI05] instead of a completely random additive sharing. Because the strings need to be correlated in a specific way, a "correction" string needs to be sent so that exactly the right secret is recovered. See below for details.

Construction 2 (Optimized 1-out-of-n OT Extension)**.**
Input of $\mathcal{S}$: *m tuples $(x_{j,0},\dots,x_{j,n-1})$ of ℓ-bit strings, $1 \le j \le m$.*
Input of $\mathcal{R}$: *m selection integers $\mathbf{r} = (r_1,\dots,r_m)$ such that $0 \le r_j < n$ for $1 \le j \le m$.*
Common Input: *a security parameter k such that $k \ge n$, and Walsh-Hadamard codes $\mathcal{C}_{\mathrm{WH}}^k = (\mathbf{c}_0,\dots,\mathbf{c}_{k-1})$.*
Oracle: *random oracles $H : [m] \times \{0,1\}^k \to \{0,1\}^\ell$, and $G : \{0,1\}^k \to \{0,1\}^m$.*
Cryptographic Primitive: *an ideal OT_m^k primitive.*

1. *$\mathcal{S}$ chooses $\mathbf{s} \leftarrow \{0,1\}^k$ at random. Let s_i denote the i-th bit of $\mathbf{s}$.*

2. *$\mathcal{R}$ forms a $(m \times k)$ matrix D by setting $\mathbf{d}_j = \mathbf{c}_{r_j}$. $\mathcal{R}$ then forms $m \times k$ matrices T_0, T_1 in the following way:*

   - *Set $\mathbf{t}_1^i = G(v_i)$ for a randomly chosen $v_i \leftarrow \{0,1\}^k$.*
   - *Set $\mathbf{t}_0^i = \mathbf{d}^i \oplus \mathbf{t}_1^i$.*

   In the above, $\mathbf{t}_0^i, \mathbf{t}_1^i$ denotes the i-th column of matrices T_0, T_1 respectively. (Note that T_0, T_1 form a pseudorandom sharing of the matrix D.)

3. *$\mathcal{S}$ and $\mathcal{R}$ interact with OT_k^k in the following way:*

   - *$\mathcal{S}$ acts as* receiver *with input $\{s_i\}_{i\in[k]}$.*
   - *$\mathcal{R}$ acts as* sender *with inputs $\{u_i, v_i\}_{i\in[k]}$, where each u_i is chosen uniformly at random from $\{0,1\}^k$. (Note v_i was already chosen by $\mathcal{R}$ in Step 2.)*
   - *$\mathcal{S}$ receives output $\{\mathbf{a}^i\}_{i\in[k]}$.*

   $\mathcal{S}$ forms $k \times k$ matrix A such that the i-th column of A is the vector $\mathbf{a}^i$.

4. *For each $i \in [k]$, $\mathcal{R}$ sends $w_i = G(u_i) \oplus \mathbf{t}_0^i$.*

5. *$\mathcal{S}$ forms $m \times k$ matrix Q such that*

   - *if $s_i = 0$, then $\mathbf{q}^i = w_i \oplus G(\mathbf{a}^i)$,*
   - *else if $s_i = 1$, then $\mathbf{q}^i = G(\mathbf{a}^i)$.*

Let $\mathbf{q}_j$ denote the j-th row of Q. (Note $\mathbf{q}^i = \mathbf{t}^i_{s_i}$. Note $\mathbf{q}_j = ((\mathbf{t}_{j,0} \oplus \mathbf{t}_{j,1}) \odot \mathbf{s}) \oplus \mathbf{t}_{j,0}$. Simplifying, $\mathbf{q}_j \oplus \mathbf{t}_{j,0} = \mathbf{c}_{r_j} \odot \mathbf{s}$.)

6. *For $j \in [m]$ and for every $0 \leq r < n$, $\mathcal{S}$ sends $y_{j,r} = x_{j,r} \oplus H(j, \mathbf{q}_j \oplus (\mathbf{c}_r \odot \mathbf{s}))$.*
7. *For $j \in [m]$, $\mathcal{R}$ recovers $z_j = y_{j,r_j} \oplus H(j, \mathbf{t}_{j,0})$.*

The amortized cost per instance of the $\binom{n}{1}$-OT^m_ℓ protocol above is $(k + n\ell)$. This yields a OT^m_ℓ protocol whose amortized concrete cost per instance is $n\ell + (k/\log n)$ bits.

Note that our pseudorandom secret sharing technique can be applied to the IKNP construction as well. Such an application would reduce the IKNP cost of OT^m_ℓ from $m(2k + 2\ell)$ to $m(k + 2\ell)$. Observe that the reduced costs also have an impact on the oblivious key transfer phase (by constant factor 4/3) of garbled circuit based constructions where $\ell = k$.

D.1 Concrete Performance Analysis

We analyse the concrete performance improvements resulting from the observation above.
Cost of IKNP. In order to perform a reduction from OT^m_ℓ to OT^k_k, the communication cost of IKNP is given by $2mk + 2m\ell$. Using the optimization described above, this can be reduced to $mk + 2m\ell$. The amortized cost per instance is $k + 2\ell$. For transmitting 1-bit secrets, we see that the amortized cost per instance is $k + 2$. The security of their reduction is 2^{-k} (assuming that $m \gg k$, and that their reduction from OT^k_m to OT^k_k uses a random oracle instead of a PRG).
Cost of our protocol. Recall that the amortized cost per instance of our protocol with 1-bit secrets was derived as $(n + (2k/\log n))$ under the constraints $n^2 2^{-k/2} = 2^{-k_{\mathrm{IKNP}}}$, and $n \leq k$ (imposed by our use of Walsh-Hadamard codes). Using the optimization described above, our amortized cost for $\binom{n}{1}$-OT^m_ℓ can be reduced to $k + n\ell$ (from $2k + n\ell$). This implies that the cost of our OT^m_ℓ can be reduced to $(n + (k/\log n))$. Substituting $k = 2k_{\mathrm{IKNP}} + 4\log n$, we obtain the cost of our protocol as $(n + (2k_{\mathrm{IKNP}}/\log n) + 8)$. Thus, our task simply reduces to choosing the right value of n for which our cost function is minimized while satisfying $k = 2k_{\mathrm{IKNP}} + 4\log n \geq n$. The results are presented in Table 4.

We also present a comparison between the performance of variants of IKNP protocol and our protocols depending on inclusion of optimization in Table 5.

E 1-out-of-n OT Extension in the Preprocessing Model

We present our extension protocol in the preprocessing model. We do this by using a straightforward generalization of Beaver's circuit randomization technique [Bea95, Bea92, IKM+13]. The resulting construction has offline communication complexity $O(mk)$, and online communication complexity $O(m \log n + mn\ell)$. We present our construction assuming the existence of a suitable code $\mathcal{C}$.

The storage complexity of our offline phase is $O(mn\ell)$ bits on the sender side, and $O(m\ell)$ bits on the receiver side. Observe that no cryptographic operations are performed in the online phase.

Construction 3 (1-out-of-n OT Extension in the Preprocessing Model)**.**

OFFLINE PHASE.
COMMON INPUT: *a security parameter k such that $k \geq n$, and a suitable code $\mathcal{C} = (\mathbf{c}_0, \ldots, \mathbf{c}_{k-1})$.*
ORACLE: *a random oracle $H : [m] \times \{0,1\}^k \to \{0,1\}^\ell$.*
CRYPTOGRAPHIC PRIMITIVE: *an ideal OT^k_m primitive.*

k_{IKNP}	Optimized IKNP	our (optimized) cost	min n	our cost	improvement factor
20	22	$n+8+40/\log n$	6	30	0.733333
30	32	$n+8+60/\log n$	8	36	0.888889
40	42	$n+8+80/\log n$	8	43	0.976744
50	52	$n+8+100/\log n$	11	48	1.08333
60	62	$n+8+120/\log n$	11	54	1.14815
80	82	$n+8+160/\log n$	15	64	1.28125
100	102	$n+8+200/\log n$	16	74	1.37838
120	122	$n+8+240/\log n$	16	84	1.45238
160	162	$n+8+320/\log n$	21	102	1.58824
200	202	$n+8+400/\log n$	22	120	1.68333
240	242	$n+8+480/\log n$	27	136	1.77941
300	302	$n+8+600/\log n$	32	160	1.8875
500	502	$n+8+1000/\log n$	40	236	2.12712
1000	1002	$n+8+2000/\log n$	66	405	2.47407

Table 4: Comparison of (amortized) communication cost (measured in bits) of 1-out-of-2 bit OT for a given security level. The costs are computed assuming parties are semi-honest.

k_{IKNP}	Unopt. IKNP	Opt. IKNP	our unopt cost	imp over unopt IKNP	imp over opt IKNP	our opt cost	imp over unopt IKNP	imp over opt IKNP
20	42	22	43	0.976744	0.511628	30	1.4	0.733333
30	62	32	54	1.14815	0.592593	36	1.72222	0.888889
40	82	42	64	1.28125	0.65625	43	1.90698	0.976744
50	102	52	74	1.37838	0.702703	48	2.125	1.08333
60	122	62	84	1.45238	0.738095	54	2.25926	1.14815
80	162	82	102	1.58824	0.803922	64	2.53125	1.28125
100	202	102	120	1.68333	0.85	74	2.72973	1.37838
120	242	122	136	1.77941	0.897059	84	2.88095	1.45238
160	322	162	168	1.91667	0.964286	102	3.15686	1.58824
200	402	202	199	2.0201	1.01508	120	3.35	1.68333
240	482	242	228	2.11404	1.0614	136	3.54412	1.77941
300	602	302	271	2.2214	1.11439	160	3.7625	1.8875
500	1002	502	405	2.47407	1.23951	236	4.24576	2.12712
1000	2002	1002	708	2.82768	1.41525	405	4.94321	2.47407

Table 5: Comparison of (amortized) communication cost (measured in bits) of 1-out-of-2 bit OT for a given security level. The costs are computed assuming parties are semi-honest.

1. *$\mathcal{S}$ chooses $\mathbf{s} \leftarrow \{0,1\}^k$ at random. Let s_i denote the i-th bit of $\mathbf{s}$. $\mathcal{R}$ chooses m selection integers $\mathbf{r} = (r_1, \ldots, r_m)$ at random such that $0 \le r_j < n$ for $1 \le j \le m$.*

2. *$\mathcal{R}$ forms $m \times k$ matrices T_0, T_1 in the following way:*

   - *Choose $\mathbf{t}_{j,0}, \mathbf{t}_{j,1} \leftarrow \{0,1\}^k$ at random such that $\mathbf{t}_{j,0} \oplus \mathbf{t}_{j,1} = \mathbf{c}_{r_j}$.*

   Let $\mathbf{t}_0^i, \mathbf{t}_1^i$ denote the i-th column of matrices T_0, T_1 respectively.

3. *$\mathcal{S}$ and $\mathcal{R}$ interact with OT_m^k in the following way:*

   - *$\mathcal{S}$ acts as* receiver *with input $\{s_i\}_{i \in [k]}$.*
   - *$\mathcal{R}$ acts as* sender *with input $\{\mathbf{t}_0^i, \mathbf{t}_1^i\}_{i \in [k]}$.*
   - *$\mathcal{S}$ receives output $\{\mathbf{q}^i\}_{i \in [k]}$.*

   $\mathcal{S}$ forms $m \times k$ matrix Q such that the i-th column of Q is the vector $\mathbf{q}^i$. (Note $\mathbf{q}^i = \mathbf{t}_{s_i}^i$.) Let $\mathbf{q}_j$ denote the j-th row of Q. (Note $\mathbf{q}_j = ((\mathbf{t}_{j,0} \oplus \mathbf{t}_{j,1}) \odot \mathbf{s}) \oplus \mathbf{t}_{j,0}$. Simplifying, $\mathbf{q}_j \oplus \mathbf{t}_{j,0} = \mathbf{c}_{r_j} \odot \mathbf{s}$.)

4. *Parties perform the following operations locally:*

   - *$\mathcal{S}$ locally computes $h_{j,r} = H(j, \mathbf{q}_j \oplus (\mathbf{c}_r \odot \mathbf{s}))$ for $j \in [m]$ and $r \in [n]$.*
   - *$\mathcal{R}$ locally computes $g_j = H(j, \mathbf{t}_{j,0})$ for $j \in [m]$.*

ONLINE PHASE.
INPUT OF $\mathcal{S}$: *m tuples $(x_{j,0}, \ldots, x_{j,n-1})$ of ℓ-bit strings, $1 \le j \le m$.*
INPUT OF $\mathcal{R}$: *m selection integers $\mathbf{b} = (b_1, \ldots, b_m)$ such that $0 \le b_j < n$ for $1 \le j \le m$.*

1. *For $j \in [m]$, $\mathcal{R}$ sends $a_j = r_j - b_j$ (where the subtraction is done in the ring $\mathbb{Z}_m$) to $\mathcal{S}$.*

2. *For $j \in [m]$ and for every $0 \le r < n$, $\mathcal{S}$ sends $y_{j,r} = x_{j,r} \oplus h_{j,r+a_j}$.*

3. *For $j \in [m]$, $\mathcal{R}$ recovers $z_j = y_{j,b_j} \oplus g_j$.*

F On Instantiating the Random Oracle

In this section we define explicit primitives that may be used to replace the random oracle in our constructions. We define *$\mathcal{C}$-correlation-robust hash functions* as the following generalization of correlation-robust hash functions [IKNP03].

Definition 1 ($\mathcal{C}$-Correlation-Robust Hash Functions)**.** *Suppose k is a security parameter. Let $\mathcal{C} = \{\mathbf{c}_0, \ldots, \mathbf{c}_{n-1}\}$ be a set of k-bit strings such that $n = \mathrm{poly}(k)$ and for $i \ne j$, the Hamming distance between $\mathbf{c}_i$ and $\mathbf{c}_j$ is $\Omega(k)$. Then, $H : \{0,1\}^* \to \{0,1\}^{\ell_{out}(k)}$ is $\mathcal{C}$-*correlation-robust *if for any polynomial $p(\cdot)$ and any non-uniform polynomial-time distinguisher $\mathcal{A}$ provided with input $\mathcal{C}$, the following is negligible in the security parameter k:*

$$\left\{ R \leftarrow \{0,1\}^{\ell_{in}(k)} : \begin{array}{c} \{(i, j, H(1, w_1 \oplus ((\mathbf{c}_i \oplus \mathbf{c}_j) \odot R)))\}_{0 \le i,j < n, i \ne j}, \\ \vdots \\ \{(i, j, H(p, w_p \oplus ((\mathbf{c}_i \oplus \mathbf{c}_j) \odot R)))\}_{0 \le i,j < n, i \ne j}, \end{array} \right\}_{w_1, \ldots, w_p \in \{0,1\}^{\ell_{in}(k)}}$$

is computationally indistinguishable from the uniform distribution over $\{0,1\}^{p \cdot \ell_{out}(k)}$. (In both cases, $p = p(k)$.)

Observe that for $\mathcal{C} = \{0^k, 1^k\}$, the definition above exactly captures the notion of correlation robust hash functions from [IKNP03].

When $\mathcal{C}$ is the set containing the first n codewords of the Walsh-Hadamard code with codelength k, then we can replace the random oracle in Construction 1 with $\mathcal{C}$-correlation-robust hash functions.

For the implicit construction in Section 5.3, setting $\mathcal{C}$ to be the first n codewords of the code given by Claim 2 allows us to replace the random oracle with $\mathcal{C}$-correlation-robust hash functions.

The reduction of security of our construction to $\mathcal{C}$-correlation-robust hash functions (for the appropriate $\mathcal{C}$) is straightforward, and will appear in the full version.

Note that $\mathcal{C}$-correlation-robustness is a simple property enjoyed by a random function. Hence, any evidence that hash functions such as SHA1 or RC5 violate this property can be considered as a valid attack against these functions.