Revisiting LEGOs: Optimizations, Analysis, and their Limit

Yan Huang and Ruiyu Zhu

Indiana University, Bloomington {yh33,zhu52}@indiana.edu

Abstract. The Cut-and-choose paradigm gives by far the most popular and efficient secure two-party computation protocols in the standard malicious model, able to offer s bits of security with only s copies of garbled circuits in the one-time execution scenario [33]. Nielsen, Orlandi et al. [48,13] have even proposed the seminal idea of LEGO-style cut-and-choose to further reduce the number of circuit copies to less than s while still keep constant round complexity. However, a substantial gap still exists between the theoretical idea of LEGO cut-and-choose and a practical implementation, e.g., [48] is not compatible with free-XOR and uses expensive asymmetric key operations for soldering, while [13] leaves the important building-block of soldering unspecified.

In this work, we introduce *XOR-Homomorphic Interactive Hash* and propose an efficient implementation of this primitive by combining Reed-Solomon encoding and k-out-of-n oblivious transfers. We show how to apply this primitive to solve the performance-critical wire-soldering problem and propose a new LEGO-style cut-and-choose protocol. Comparing to previous LEGO-style protocols, ours only requires a single (as opposed to "a majority of") correctly garbled gate in each bucket to guarantee security against malicious adversaries. Plus, through integrating Half-Gates garbling, we double the chance a "bad" gate being detected in the check stage (compared to MiniLEGO [13]). Our construction is more bandwidth-efficient than Lindell (CRYPTO, 2013) [33] either when the circuit size N is sufficiently large, or when N is larger than a threshold solely determined by the ratio between the input and circuit sizes. E.g., we use less bandwidth for computing most linear and sub-linear functions.

Deploying a LEGO-style cut-and-choose protocol involves convoluted protocol parameter selection. To this end, we give a thorough analysis of the relations among all protocol parameters and propose efficient algorithms that automate the search for the optimal parameter configuration based on a requirement specification (i.e., the security parameters s, k and application parameter N) with provable accuracy.

Last, we formally prove a tight bound on the benefit of LEGO-style secure computation protocols, in the sense that the circuit duplication factor κ has to be larger than 2 and any $\kappa > 2$ is indeed achievable. This corrects a common mistake of claiming LEGO cut-and-choose can reduce κ to $O(sk/\log N)$ since $2 \notin O(sk/\log N)$.

1 Introduction

Secure two-party computation was shown to be feasible in the 1980s [56,17]. Since then, significant efforts have been devoted toward making such protocols practical. Work in this direction was pioneered by Fairplay [40], which implemented Yao's two-party computation, and followed by work that dramatically improves the software implementation [19,21,20,31,39,54], and work that significantly improves the performance of garbling [4,3,57,18]. All these work concerned the honest-but-curious model.

More desirable, of course, is to achieve security against malicious adversaries. While this is known to be feasible, in principle, using generic zero knowledge [17], a generic approach of this kind does not currently seem likely to result in efficient protocols even if specialized zero-knowledge proofs are used [26]. Indeed, the first technique explored for making efficient two-party computation protocols secure against active adversaries was the *cut-and-choose* paradigm. Alternative approaches to achieve active security in the two-party setting include the IPS compiler [25], which appears to have good asymptotic complexity [34] but seem challenging to implement in reality. Other approaches [47,12,11] have round complexity proportional to the depth of the circuit, exposing serious network latency issues in realistic deployment. Thus, in this work, we focus on protocols following the cut-and-choose paradigm.

With cut-and-choose, roughly speaking, one party generates κ garbled circuits (where κ depends on the statistical security parameter s); some fraction of those are "checked" by the other party—who aborts if any misbehavior is detected—and the remaining fraction are evaluated with the results being used to derive the final output. Cut-and-choose was first relatively naively used in Fairplay [40], which was later shown to be flawed [41,28].) A rigorous analysis of the cut-and-choose paradigm was first given by Lindell and Pinkas [35], followed by numerous others exploring variations of this technique and their application to (ever more) efficient secure two-party computation [55,48,50,36,52,32,22,33,2,23,37,38].

The critical question regarding the cut-and-choose approach is: *how many garbled-circuit copies (namely, κ) are needed to ensure some desired security level*? The value of κ has great impact on the efficiency of cut-and-choose protocols, especially as larger circuits are evaluated. Most cut-and-choose protocols' computation/communication complexity is $O(\kappa \cdot k \cdot N) + \mathsf{poly}(n_I, s, \kappa)$ where N is the circuit size, k is the computational security parameter and n_I is the input length. Typically $N \gg k$ and $N \gg s$, so if $N \gg n_I$, minimizing κ is obviously of key importance.

1.1 Prior Work

Notably, Lindell showed that, when circuit size is significantly greater than the input and output length of the target function, s bits of statistical security can be obtained with merely s copies of the circuits (i.e., $\kappa = s$). The key idea is to penalize cheating circuit generators, whose secret input x will be revealed to the honest evalutor, through running conventional actively-secure cut-and-choose protocols [35,36] per output bit. Alternatively, Huang et al. [22] proposed symmetric cut-and-choose that offers the same parallel cost ($2s$ copies are generated in total, concurrently), but avoid using expensive conventional cut-and-choose to bootstrap.

Nielsen and Orlandi [48] proposed the seminal LEGO approach, which allows the evaluator to cut-and-choose individual gates and exploits the evaluator's randomness in grouping garbled gates to thwart active attacks from the generator. This could reduce κ to values even less than s for most circuits! While their first proposal was based on garbling NAND gates and required expensive asymmetric-key operations for soldering, an improved version of this technique, MiniLEGO [13], is compatible with free-XOR technique and would use only symmetric-key operations. However, both work focused on demonstrating the asymptotic advantage, leaving the concrete analysis open to practitioners. In fact, it is not possible to derive a concrete measurement of improvement for MiniLEGO as the construction of some critical components, e.g., the ssenc algorithm heavily used in wire soldering, remain unspecified.

More recently, researchers studied the amortized cost of securely computing a function f many times through cut-and-choose circuits across many executions [37,23,38]. The conclusion was that the amortized κ for cut-and-choose can be reduced to $O(s/\log N)$ where N is the number of executions. However, this result does not apply to a single execution of f, which fits to many practical scenarios where users want to run different computations with different peers in an ad hoc fashion. Thus, in the remainder of this paper, we restrict our attention to cut-and-choose protocols that target single execution scenarios.

Related to our work on XOR-Homomorphic Interactive Hash, [48,13,38,53] considered the problem of realizing XOR-homomorphic commitments. However, none of them satisfies the tight budget of practically useful LEGO-protocols, while we note that XOR-homomorphic commitments, which is a stronger primitive than XOR-homomorphic interactive hash, is not necessary for building secure LEGO-protocols.

In an independent and current work, Frederiksen et al. [14] made some similar exploration. Nevertheless, our work is distinct in that: (1) we formally prove the tight bound on the efficiency of any LEGO-protocols and identify a common mistake on some efficiency claims in prior work of LEGO-style protocols; (2) we introduce the notion of XOR-Homomorphic Interactive Hash to address the wire soldering problem, while they use XOR-Homomorphic Commitment and regular cryptographic hashes; (3) We present analytical and numerical security analysis and a highly efficient automated tool to optimize the protocol parameters with proved high accuracy.

1.2 Our Contribution

Prior work [13,48] on LEGO-family of protocols claims that they offer a factor of $O(\log N)$ savings comparing to non-LEGO cut-and-choose (where N is the size of the circuit). We show that *this claim does not hold.*

In fact, the savings have to be upper-bounded by $s/2$, where s is the statistical security parameter. We give a rigorous proof (Section 6) that the benefit of LEGO-protocols is *tightly bounded* at a duplication factor $\kappa = 2$, in the sense that any $\kappa > 2$ (note κ can be decimals) is achievable if N is sufficiently large, while any $\kappa \leq 2$ is impossible to attain however large N is. Note that this result also applies to correct a similarly fallacious claim on the amortized performance of cut-and-choose [37].

This lower bound on the best achievable κ postulates a tight budget on handling each garbled gate in every LEGO-protocol that aims to outperform the most efficient non-LEGO secure computation protocol in practice. Using Lindell's cheat-then-reveal protocol [33] (the most efficient construction when $N \gg n_I, N \gg k$) as a baseline, a LEGO-protocol outperforming this baseline could afford at most $s \cdot c(k)/2$ on each garbled gate because $\kappa > 2$ (where $c(k)$ is the cost per boolean gate using [33]). For example, considering the cost of bandwidth, when $s = 40$, $k = 128$, $c(k) = 256$ bits (assuming Half-Gates [57] is used), any LEGO-like protocol that uses more than $40 \times 256/2 = 5120$ bits of bandwidth per garbled gate will not outperform [33] when $N \gg n_I$. No existing LEGO-protocols achieved such a tight budget. In fact, the state-of-the-art LEGO-protocol [13] left its solution to "soldering", the most performance-critical component, unspecified.

In this work, we propose an efficient actively-secure two-party computation protocol in the Random Oracle Model (ROM) [6] that fully instantiates the idea of LEGO. It is the first construction that meets the aforementioned budget limit, and can actually outperform Lindell's cheat-and-reveal protocol [33]. We show that a single (as opposed to "majority" required by MiniLEGO) correctly garbled gate in every bucket suffices to guarantee active-security (Section 4.4, Lemma 3). In addition, we show how to leverage the idea of Half-Gates [57] to improve the evaluator's detection rate (from 1/4 in MiniLEGO) to 1/2 (Lemma 4). The two enhancements alone can lead to substantial savings compared to MiniLEGO, e.g., 15% when $N = 10K$ and 25% when $N = 100K$.

We introduce a new cryptographic primitive, called *XOR-Homomorphic Interactive Hash*, which played a paramount role in our LEGO-protocol construction but may be of independent interest as well. It is the primitive that enables efficient wire "soldering". It also helps to relax the security requirement to "one good gate per bucket". This is because the evaluator in our protocol will learn the generator's global secret $\mathit{\Delta}$ whenever two "valid" but different wire labels were found at any bucket, where the "validity" is determined by a label's interactive hash.

An XOR-homomorphic interactive hash scheme involves two parties (a message holder and a hash receiver) where the hash receiver does not learn any information beyond what can be (efficiently) inferred from the hash while the message holder does not learn the hash. It offers some sense of "hiding" (which is precisely defined in terms of the ideal functionality $\mathcal{F}_{\text{XorIHash}}$, see Section 3.1), in addition to the "binding" property that comes with traditional hashes. However, this security notion differs from the standard notion of cryptographic commitment in that it allows leaking partial information on the pre-image of the hash.

We provide an instantiation of $\mathcal{F}_{\text{XorIHash}}$, which is inspired by Reed-Solomon encoding [51] and makes black-box calls to a constant number of k-out-of-n oblivious transfers. The basic idea is to obliviously transfer k symbols (chosen by the hash receiver) out of the n-symbol encoding of a message. If Reed-Solomon code is used, we can carefully choose n, k and σ (number of bits per symbol) to *statistically hide* the pre-image to the hash receiver meanwhile *statistically bind* the pre-image to the hash sent to the receiver (because any change in the original message will be evident from one of the k symbols in the encoding watched by the receiver with overwhelming probability). We show how to efficiently identify the optimal parameters of the interactive hash scheme to mininize the overall bandwidth for the secure two-party computation protocol (Section 5.1).

We formally prove the security of our secure two-party computation protocol in the active adversary model (Section 4.4). We present a thorough numerical analysis of the complex relation between the bandwidth cost and the rich set of parameters, including s, k, T, N, B, τ (see Figure 1 for definitions of these variables) in the main protocol, and n, σ, w, ℓ (see Figure 3 for their definitions) of the interactive hash protocol. In particular, we present efficient algorithms that automate the search of the optimal parameter setting based on the security parameters s, k and the circuit parameter N. Our work has pushed LEGO-style two-party computation protocols a step closer to practical deployment.

2 Notation and Building Blocks

We let H be a hash function that will be treated in the analysis as a random oracle. We use the standard definitions of secure two-party computation for active adversaries [16].

We use P_1 to denote the circuit generator and P_2 the circuit evaluator. In addition, we assume for simplicity that only P_2 will receive $f(x, y)$, the output of the computation (over P_1's secret input x and P_2's secret input y). We assume the original function f can be computed by a circuit C consisting of N AND gates (whereas the rest are all XORs). To execute it securely, our protocol will generate a total number of T garbled AND gates. We summarize in Figure 1 the variables used to describe our main secure computation protocol.

s	The statistical security parameter.
k	The computational security parameter.
C	The boolean circuit that computes the target function f.
N	Number of AND gates in C (i.e., the number of buckets).
T	Total number of garbled AND gates generated by P_1.
B	Number of garbled AND gates in a bucket.
τ	P_2's detection rate when checking a bad gate.
λ_w	Wire label length, in bits.
λ_p	Length of permutation randomness (denoted by ρ) in bits.

Fig. 1: Variables in XOR-Homomorphic Verifiable Commitment Scheme

2.1 Oblivious Transfer

We make black-box use of 1-out-of-2 oblivious transfers (OT) to send wire labels corresponding to the evaluator's input, and k-out-of-n OT in constructing the XOR-Homomorphic interactive hash. A k-out-of-n oblivious transfer protocol takes n messages $m_1, \ldots, m_n$ from the sender and a set of k indices $i_1, \ldots, i_k \in [1, n]$ from the receiver, and outputs the k messages $m_{i_1}, \ldots, m_{i_k}$ to the receiver (but nothing else). Actively secure 1-out-of-2 OT can be efficiently constructed [49,44] and extended [27,1] so that only a small number of expensive base OTs are needed. An actively secure k-out-of-n oblivious transfer protocol could be efficiently constructed from black-box use of 1-out-of-2 committed oblivious transfer (COT) protocols [29,15], or slightly more efficiently by combining committing OT [28] and committed OT. UC secure committed and committing 1-out-of-2 OTs could be efficiently instantiated from dual-mode cryptosystems [49]. Camenisch, Neven, and Shelat [8] proposed efficient and simulatable k-out-of-n OT in the Random Oracle Model.

2.2 Garbled Circuits

Following the formalism proposed by Bellare, Hoang, and Rogaway [4], a *garbling scheme* $\mathcal{G}$ is is 5-tuple $(\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De}, f)$ of algorithms, where Gb is an efficient randomized *garbler* that, on input $(1^k, f)$, outputs (F, e, d); En is an *encoder* that, on input (e, x), outputs X; Ev is an *evaluator* that, on input (F, X), outputs Y; De is a *decoder* that, on input (d, Y), outputs y. The correctness of $\mathcal{G}$ requires for every $(F, e, d) \leftarrow \mathsf{Gb}(1^k, f)$ and every x, $\mathsf{De}(d, \mathsf{Ev}(F, \mathsf{En}(e, x))) = f(x)$. Let Φ be a prefixed function modeling the acceptable information leak and "$\approx$" symbolizes *computational indistinguishability. Privacy* of $\mathcal{G}$ implies that there exists an efficient simulator $\mathcal{S}$ such that for any $x \in \{0, 1\}^{n_I}$,

$$\big\{(F, e, d) \leftarrow \mathsf{Gb}(1^k, f), X \leftarrow \mathsf{En}(e, x) : (F, X, d)\big\} \approx \{\mathcal{S}(1^k, f(x), \Phi(f))\}.$$

Obliviousness of $\mathcal{G}$ implies that there exists an efficient simulator $\mathcal{S}$ such that

$$\{(F, e, d) \leftarrow \mathsf{Gb}(1^k, f), X \leftarrow \mathsf{En}(e, x) : (F, X)\} \approx \{\mathcal{S}(1^k, \Phi(f))\}.$$

Authenticity of $\mathcal{G}$ requires that for every efficient adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,

$$\Pr\left(Y \neq \mathsf{Ev}(F, X) \textbf{ and } \mathsf{De}(d, Y) \neq \bot) : \begin{array}{l} (f, x) \leftarrow \mathcal{A}_1(1^k), \\ (F, e, d) \leftarrow \mathsf{Gb}(1^k, f), X \leftarrow \mathsf{En}(e, x), \\ Y \leftarrow \mathcal{A}_2(1^k, F, X). \end{array} \right) = \mathsf{negl}(k),$$

where negl is a negligible function.

Our protocol, uses the Half-Gates garbling scheme recently proposed by Zahur, Rosulek, and Evans [57], which offers the simulation-based definition of privacy, obliviousness, and authenticity under a *circular correlation robustness* assumption of the hash function H. We summarize their AND gate Gb and Ev algorithms, GenAND (Algorithm 1) and EvlAND (Algorithm 2), respectively, in the Appendix.

2.3 LEGO-Style Cut-and-Choose

Nielsen and Orlandi [48] proposed the seminal idea of LEGO, which enables the circuit evaluator to check individual garbled gates and randomly group evaluation gates to accomplish actively secure two-party computation. The first construction [48] was based on NANDs and require public key operations for soldering. Fredericksen et al. [13] improved the LEGO construction to work with free-XOR technique. The backbone of LEGO-style protocols are made of three basic components:

1. **Generate.** P_1 generates a total of T garbled gates.
2. **Evaluate.** P_2 randomly picks BN gates and groups them into N buckets, each of which will realize a gate in C. P_2 evaluates every bucket by first translating the bucket's input-wire labels to input-wire labels of each garbled gate in the bucket, then evaluating the garbled gates and finally translating the output-wire labels obtained from evaluating gates of the same bucket to the output-wire labels of the bucket for majority selection. (The wire label translation is also known as *wire soldering*.
3. **Check.** P_2 checks the rest $T - BN$ gates for their correctness, using per-gate secrets revealed by P_1. Note that unlike the circuit checking of traditional cut-and-choose, every bad gates could be detected with probability less than 1.

Note that in [48], all garbled gates in the description above are NANDs, as NANDs alone is *functional complete* to realize arbitrary circuits; while with [13], all garbled gates are ANDs, which, combined with XORs, are also functional complete. Because XORs can be securely computed locally, no extra cut-and-choose is needed to ensure P_1's correct behavior on XORs.

3 XOR-Homomorphic Interactive Hash

3.1 Definition.

We introduce the notion of XOR-Homomorphic Interactive Hash, which is a pair of algorithms ($\mathsf{Hash}, \mathsf{Verify}$), where Hash involves two participants, P_1 and P_2. Roughly speaking, Hash is an efficient two-party probabilistic algorithm that takes a message m from P_1 and outputs nothing to P_1, but *only* the hash of m (denoted as $\langle m \rangle$) to P_2 without revealing any extra information. Verify is an efficient deterministic algorithm (executable by P_2 alone) that takes a hash $\langle m \rangle$ and a message m' as inputs, and outputs a bit b indicating whether $m{=}m'$ (but nothing beyond). Like conventional notion of hashing, we require $|\langle m \rangle| < |m|$ and that for any two distinct messages m_1 and m_2 from P_1, their hashes received by P_2 are distinct except for a negligible probability. In addition, the hashes need to be XOR-homomorphic, i.e., $\langle m_1 \rangle \oplus \langle m_2 \rangle = \langle m_1 \oplus m_2 \rangle$.

Formally, we define the ideal functionality $\mathcal{F}_{\text{XORIHASH}}$ of XOR-Homomorphic Interactive Hash in Figure 2.

– **Hash.** Upon receiving `(Hash,` m`)` from the committer P_1: if there is a recorded value (cid, m), generate a delayed output `(Receipt,`cid`,` $\langle m\rangle$`)` to P_2, where $\langle m\rangle$ denotes the hash of m and $|\langle m\rangle| < |m|$; otherwise, pick a fresh number cid, record (cid, m) and generate a delayed output `(Receipt,`cid`,` $\langle m\rangle$`)` to P_2.
– **Verify.** Upon receiving `(Verify,` $cid_1, \ldots, cid_t$`,` d`)` for any $t \geq 1$ from P_2: if there are recorded values $(cid_1, m_1), \ldots, (cid_t, m_t)$ (otherwise do nothing), set $z = 1$ if $m_1 \oplus \cdots \oplus m_t = d$, and $z = 0$, otherwise; generate a delayed output `(VerifyResult,` z`)` to R.

Fig. 2: Ideal functionality of XOR-Homomorphic Interactive Hash. ("send a delayed output x to party P" signals a standard treatment of fairness: it means "send (x, P) to the adversay; whenreceiving `ok` from the adversary, output x to P.")

Like cryptographic commitments, $\mathcal{F}_{\text{XORIHASH}}$ also offers certain "*hiding*" and "*binding*" properties: (1) Even after receiving $\langle m\rangle$, P_2 learns nothing about m except for what can be efficiently inferred from $\langle m\rangle$; (2) Once $\langle m\rangle$ is sent to P_2, P_1 (who does not know $\langle m\rangle$) can't claim $\langle m\rangle$ to be the hash of a different message m'. Due to these "limited" notions of hiding and binding, we borrow two expressions from the domain of commitments:

1. **"P_1 *sends* $\langle m\rangle$ to P_2"** refers to the action where P_1 and P_2 collaboratively run Hash over P_1's private input m so that P_2 learns $\langle m\rangle$ (but nothing else) while P_1 learns nothing.
2. **"P_1 *opens* $\langle m\rangle$ to P_2"** refers to the action where P_1 reveals the pre-image m to P_2, who then checks whether m is consistent with its claimed hash $\langle m\rangle$ via calling $\mathsf{Verify}(\langle m\rangle, m)$.

Unlike traditional commitments, which fully hides the committed message to the receiver, an interactive hash allows leaking some information of m to the receiver, yet preserve the remaining entropy (if m is picked uniformly random, there has to be some entropy left since $|\langle m\rangle| < |m|$). In addition, with an interactive hash scheme, the hashes are kept secret from the party (i.e. P_1) who holds the pre-images, while the party (i.e. P_2) who has the hash can efficiently query (by itself) whether a message is a pre-image of the hash.

3.2 Construction.

Inspired by Reed-Solomon code, we give a concrete implementation of XOR-homomorphic interactive hash scheme that leverages a black-box use of w-out-of-n oblivious transfer protocols. If P_1's input messages to Hash are sampled uniformly random, our construction perfectly hides $|m| - |\langle m\rangle|$ bits of information in m. Meanwhile, it guarantees statistical binding in that P_1 can only claim a different message m' to be the pre-image of $\langle m\rangle$ with negligible probability. We summarize the variables relevant to describing the interactive hash protocol in Figure 3.

$\mathbb{F}_{2^\sigma}$	The symbol space, i.e., the Finite Field of size 2^σ.
n	Number of symbols in a codeword.
w	Number of symbols to be watched by the receiver.
ℓ	Number of symbols in a message.
σ	Number of bits to represent a symbol.
μ	Bits of entropy (per message) retained.

Fig. 3: Variables in the construction of XOR-homomorphic interactive hash

Our construction combines the idea of Reed-Solomon code and oblivious transfer: to hash a message $\boldsymbol{m}$ (we use bold face $\boldsymbol{m}$ to emphasize that messages are arrays of σ-bit *symbols*), P_1 encodes its ℓ-symbol private message m using a Reed-Solomon code and then sends the n-symbols encoding to P_2 using a w-out-of-n obliviously transfer. Intuitively, *perfect hiding* is guaranteed for uniform-randomly sampled messages of P_1, because even after observing w symbols of P_2's choice, there remains $2^{(\ell-w)\sigma}$ possible pre-images equally likely mapping to $\langle m\rangle$; *statistical binding* is ensured since if P_1 claims the hash with a different message $\boldsymbol{m'}$, it will be detected by P_2 with probability at least $\binom{\ell-1}{w}/\binom{n}{w}$ (since the encodings of $\boldsymbol{m}$ and $\boldsymbol{m'}$ will be identical at $\ell-1$ symbols at most while w out of the n encoding symbols have been watched by P_2).

Our optimized instantiation of $\mathcal{F}_{\text{XorIHash}}$ is detailed in Figure 4. We leverage a symmetric encryption scheme, $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, to limit the use of OT to Setup stage only. As Setup needs to run only once to prepare P_1 and P_2 for all subsequent calls to Hash and Verify, the overall cost of OT is well amortized.

– Setup
  1. P_1 runs KeyGen n times to obtain n keys, denoted by $\{k_1, \ldots, k_n\}$.
  2. The P_2 randomly picks a set of w integers from $\{1, \ldots, n\}$, denoting them with $\{i_1, \ldots, i_w\}$.
  3. P_1 and P_2 execute a w-out-of-n OT protocol where P_1 is the sender with inputs $(k_1, \ldots, k_n)$ while P_2 is the receiver with input choices $(i_1, \ldots, i_w)$. At the end of this step, P_2 learns $\{k_{i_1}, \ldots, k_{i_w}\}$.
  4. P_1 and P_2 agree on a set of n $(\ell < n \le 2^\sigma)$ ℓ-dimensional vectors $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\} \subset \left\{ [1, x_i^1, \ldots, x_i^\ell] \,\middle|\, x_i \in \mathbb{F}_{2^\sigma} \right\}$, where any subset of ℓ vectors are linearly independent.

– $\mathsf{Hash}(\boldsymbol{m})$, where $\boldsymbol{m} \in \mathbb{F}_{2^\sigma}^\ell$ is the message to hash.
  1. P_1 computes $\mathsf{Enc}_{k_1}(\boldsymbol{v}_1 \cdot \boldsymbol{m}), \ldots, \mathsf{Enc}_{k_n}(\boldsymbol{v}_n \cdot \boldsymbol{m})$ and sends them to P_2.
  2. P_2 decrypts all watched symbols using $k_{i_1}, \ldots, k_{i_w}$ to obtain $(\boldsymbol{v}_{i_1}\boldsymbol{m}, \ldots, \boldsymbol{v}_{i_w}\boldsymbol{m})$, which is output as the interactive hash of $\boldsymbol{m}$, written as $\langle \boldsymbol{m} \rangle$.

– $\mathsf{Verify}(\langle \boldsymbol{m} \rangle, \boldsymbol{m'})$.
  1. P_2 return 1 if for all $j \in \{i_1, \ldots, i_w\}$, $\boldsymbol{v}_j \cdot \boldsymbol{m'} = \langle \boldsymbol{m} \rangle_j$, where $\langle \boldsymbol{m} \rangle_j$ denotes the j-th entry in the hash; and 0, otherwise.

Fig. 4: Realize XOR-Homomorphic Interactive Hash

3.3 Proof of Security

It can be proved that the construction in Figure 4 realizes $\mathcal{F}_{\text{XorIHash}}$ in the honest-but-curious model, though this is not much more useful than Lemma 1 in the security proof of our main protocol. (Thus, we leave the proof of the theorem to the full version of the paper.) Intuitively, the lemma suggests that (1) if $\boldsymbol{m}$ is picked random, $(\ell-w)\sigma$ bits of entropy remains even if after $\langle \boldsymbol{m} \rangle$ is sent; (2) any w symbols is a valid hash; (3) a hash $\langle \boldsymbol{m} \rangle$ statistically binds the sender to the message $\boldsymbol{m}$.

Lemma 1. *Assume σ-bit symbols. Let $\boldsymbol{m}$ be a uniform-randomly sampled ℓ-symbol message and $\langle \boldsymbol{m} \rangle$ be the w-symbol hash of $\boldsymbol{m}$ produced by the* Hash *algorithm of Figure 4. Let $\mathbf{H}_{min}$ be the min-entropy function. Then*

1. $\mathbf{H}_{min}(\boldsymbol{m} | \langle \boldsymbol{m} \rangle) = (\ell - w)\sigma$. *That is, in P_2's perspective, there is still $(\ell-w)\sigma$ bits of entropy about $\boldsymbol{m}$ after P_2 receives $\langle \boldsymbol{m} \rangle$.*
2. *Any tuple of w symbols can be regarded as an interactive hash of some message $\boldsymbol{m}$.*
3. *Once P_2 receives $\langle \boldsymbol{m} \rangle$, P_1, even with unbounded computation power, cannot open $\langle \boldsymbol{m} \rangle$ to a different message $\boldsymbol{m'}$ with probability more than $\binom{\ell-1}{w}/\binom{n}{w}$.*

Theorem 1. *In the semi-honest model, the protocol in Figure 4 realizes $\mathcal{F}_{\text{XorIHash}}$ defined in Figure 2.*

4 The Protocol

4.1 High-Level Description

At the high level, the protocol proceeds in the following steps:

1. **Generate garbled ANDs.** P_1 generates a total of T garbled AND gates using the Half-Gates garbling algorithm GenAND (but using elongated wire labels).
2. **Commit and send garbled AND gates.** P_1 commits and sends all garbled ANDs it generated to P_2.
3. **Evaluate.** P_2 randomly picks a random string $\mathcal{J}$ and send it to P_1. Using $\mathcal{J}$ as the source of randomness, P_1 and P_2 randomly select BN ANDs and group them into N buckets, each of which will be used to compute an AND gate in C. For every bucket,
   (a) P_1 and P_2 securely solder the B garbled ANDs using the XOR-homomorphic interactive hash. P_2 delays its response to soldering failures to step 5b.
   (b) P_2 evaluates all B garbled ANDs in the bucket.
   P_2 locally evaluates all XORs in circuit C.
4. **Check.** P_2 checks the rest $(T - BN)$ garbled ANDs to verify their correctness (but delays its response to check-failures to step 5b).
5. **Output determination.**
   (a) P_1 "opens" the permutation randomness associated to the final output-wires so that P_2 knows how to interpret the final output-wire labels.
   (b) If any failure occurred at the time of checking (step 4), soldering (step 3a), or opening (step 5a) above, P_2 aborts. Otherwise, P_2 translates the output-wire labels into plaintext bits.

Remark 1. Our protocol is distinguished from previous LEGO-style protocols in that it requires only one correctly generated garbled gate in each bucket. This is enabled by combining the free-XOR technique with our XOR-homomorphic hash: (1) P_2 can always learn, from $\langle \Delta \rangle$ and $\langle w^p \rangle$ (the hash of the 0-label permuted by a bit p associated with every wire), whether a wire label w it obtains in the evaluation stage is consistent with either $\langle w^0 \rangle$ or $\langle w^1 \rangle$ regardless of the value of p; (2) whenever P_2 learns both w^0 and w^1 (via evaluating both good and bad gates in the same bucket), it can learn the global secret Δ and further P_1's secret input x, except with negligible probability. We will formally prove this point as Lemma 3.

Remark 2. Unlike MiniLEGO [13] where the detection rate $\tau = 1/4$, we are able to increase τ to $1/2$, through leveraging the Half-Gates garbling [57]. The intuition behind this idea is that a correctly garbled gate can be modeled as the solution to a linear system of four equations over three variables where all the coefficients are fixed by the six interactive hashes associated with the gate. Since any three of the four coefficients vectors are linearly independent (due to the hash function used in the Half-Gates garbling), if P_1 cheats so that one of the equations does not hold, there has to be at least another equation that is also not satisfied. That is, if the gate is bad, at least two out of four rows of the garbled table is corrupted. We will formally prove this observation as Lemma 4.

Remark 3. We stress that P_2 should not complain in event of: (1) any bad gate found in the checking stage (step 4); (2) XOR values not matching with their hashes in soldering (step 3a); and (3) any pre-images not matching the hashes of the wire permutation randomness in step 5a, until in the end (step 5b). This prevents malicious P_1 from probing the watched symbols of P_2 from observing P_2's responses.

4.2 Formal Specification

Fix a function f that P_1 and P_2 wish to compute over their respective inputs x, y (let $n_I^{P_1}$ and $n_I^{P_2}$ are the bit length of x and y, respectively. Assume f is realized by a boolean circuit C. The protocol proceeds as follows.

0. **Setup.**

(a) P_1 and P_2 execute the Setup step of the XOR-homomorphic hash scheme, where P_1 is the hash sender and P_2 is the hash receiver.

(b) P_1 randomly samples the global secret $\Delta \leftarrow \{0,1\}^{\lambda_w}$ (as is used in Half-Gates [57] to enable the free-XOR technique [30]) and commit it to P_2, who learns $\langle \Delta \rangle$.

1. **Generate garbled ANDs.** For $i = \{1, \ldots, T\}$, P_1 runs the garbling algorithm GenAND (Figure 1) to create T garbled AND gates:
$$(w^0_{i,l}, w^0_{i,r}, w^0_{i,o}, T_{i,G}, T_{i,E}) \leftarrow \mathsf{GenAND}(i, \Delta)$$
where $w^0_{i,l}, w^0_{i,r}, w^0_{i,o}$ are the wire labels representing 0-values on the left input-wire, the right input wire, and the output-wire, respectively, of the i-th garbled AND; $T_{i,G}$ is the single garbled row in the generator half-gate and $T_{i,E}$ the single row in the evaluator half-gate.
2. **Commit and send garbled AND gates.** For $i = \{1, \ldots, T\}$,
   (a) P_1 samples $\rho^{i,l} \leftarrow \{0,1\}^{\lambda_p}, \rho^{i,r} \leftarrow \{0,1\}^{\lambda_p}, \rho^{i,o} \leftarrow \{0,1\}^{\lambda_p}$, and computes $w^1_{i,l} := w^0_{i,l} \oplus \Delta$; $w^1_{i,r} := w^0_{i,r} \oplus \Delta$; $w^1_{i,o} := w^0_{i,o} \oplus \Delta$.
   (b) Let $p_{i,l} = \rho^{i,l}_1 \oplus \cdots \oplus \rho^{i,l}_{\lambda_p}$ (where $\rho^{i,l}_j$ denotes the j-th bit of $\rho^{i,l}$), while $p_{i,r}$ and $p_{i,o}$ are similarly derived from $\rho^{i,r}$ and $\rho^{i,o}$, respectively. P_1 and P_2 hash $\rho^{i,l}, w^{p_{i,l}}_{i,l}$, $\rho^{i,r}, w^{p_{i,r}}_{i,r}$, $\rho^{i,o}, w^{p_{i,o}}_{i,o}$ so that P_2 learns $\langle \rho^{i,l} \rangle$, $\langle w^{p_{i,l}}_{i,l} \rangle$, $\langle \rho^{i,r} \rangle$, $\langle w^{p_{i,r}}_{i,r} \rangle$, $\langle \rho^{i,o} \rangle$, $\langle w^{p_{i,o}}_{i,o} \rangle$, respectively.
   (c) P_1 sends $T_{i,G}, T_{i,E}$ to P_2.
3. **Evaluate.** P_2 randomly picks a random string $\mathcal{J}$ and send it to P_1. Using $\mathcal{J}$ as a random source, BN garbled ANDs are selected and grouped into N buckets.
P_1 and P_2 follow an identical topological order to process the initial input-wires and every binary gate of the boolean circuit C. (Note C only contains ANDs and XORs.)
   (a) For every input-wire W_i associated to P_1's private input bit x_i: P_1 samples $\rho^i \leftarrow \{0,1\}^{\lambda_p}$ and a wire label $w^0_i \leftarrow \{0,1\}^{\lambda_w}$ (thus $w^1_i := w^0_i \oplus \Delta$) and sends $w^{x_i}_i$ and the hashes $\langle \rho^i \rangle$, $\langle w^{p_i}_i \rangle$ to P_2 (where $p_i = \rho^i_1 \oplus \cdots \oplus \rho^i_{\lambda_p}$).
   (b) For every input-wire W_i associated to P_2's private input y_i:
      i. W_i is $\oplus$-split into s wires $W_{i,1}, \ldots, W_{i,s}$.
      ii. P_1 samples $w^0_{i,1} \leftarrow \{0,1\}^{\lambda_w}, \ldots, w^0_{i,s} \leftarrow \{0,1\}^{\lambda_w}$ and runs the interactive hashing with P_2 so that P_2 learns $\langle w^0_{i,1} \rangle, \ldots, \langle w^0_{i,s} \rangle$.
      iii. P_2 samples $y_{i,1} \leftarrow \{0,1\}, \ldots, y_{i,s} \leftarrow \{0,1\}$ subject to $y_{i,1} \oplus \cdots \oplus y_{i,s} = y_i$.
      iv. For every $j \in \{1, \ldots, s\}$, P_2 retrieves $w^{y_{i,j}}_{i,j}$ from P_1 through oblivious transfer, and verifies $w^{y_{i,j}}_{i,j}$ against $\langle w^{y_{i,j}}_{i,j} \rangle$ (note P_2 can compute $\langle w^{y_{i,j}}_{i,j} \rangle := \langle w^0_{i,j} \rangle \oplus y_{i,j} \langle \Delta \rangle$). A verification failure at any j results in P_2's delayed abort at step 5.
      v. P_2 sets $w^{y_i}_i := w^{y_{i,1}}_{i,1} \oplus \cdots \oplus w^{y_{i,s}}_{i,s}$ and $\langle w^{y_i}_i \rangle := \langle w^{y_{i,1}}_{i,1} \rangle \oplus \cdots \oplus \langle w^{y_{i,s}}_{i,s} \rangle$, $\langle r^i \rangle := \langle 0 \rangle$.
   (c) For every AND gate in C (recall that P_2 should have already obtained two wire labels w^a_l, w^b_r, which represent the signals a, b on the left and right input wires of the AND gate, and hashes $\langle \rho^l \rangle, \langle w^{p_l}_l \rangle$ and $\langle \rho^r \rangle, \langle w^{p_r}_r \rangle$):
      i. P_2 selects B garbled ANDs (solely determined by randomness r_{cc}), $g_1, \ldots, g_B$, from $\mathcal{T} - \mathcal{K}$ and notifies P_1 its choices.
      ii. P_1 samples $\rho^o \leftarrow \{0,1\}^{\lambda_p}$ and $w^0_o \leftarrow \{0,1\}^{\lambda_w}$ and sends hashes $\langle \rho^o \rangle$ and $w^{p_o}_o$ to P_2 (where $p_o = \rho^o_1 \oplus \cdots \oplus \rho^o_{\lambda_p}$). P_2 sets $\mathcal{O}$ to an empty set, and executes the following for $i \in \{1, \ldots, B\}$ (note that whenever P_2 receives an incorrect preimage reuqired to verify a hash, it continues its execution until its delayed abort at step 5),
         A. Let $p_{i,l} = g_i.p_l$, $p_{i,r} = g_i.p_r$. P_1 opens $\langle \rho^l \rangle \oplus \langle \rho^{i,l} \rangle$, $\langle \rho^r \rangle \oplus \langle \rho^{i,r} \rangle$ to P_2 so that P_2 learns $p_l \oplus p_{i,l} := \bigoplus_{1 \le j \le \lambda_p} (\rho^l_j \oplus \rho^{i,l}_j)$ and $p_r \oplus p_{i,r} := \bigoplus_{1 \le j \le \lambda_p} (\rho^r_j \oplus \rho^{i,r}_j)$.
         B. Let $w^x_{i,l} = g_i.w^x_l$ and $w^x_{i,r} = g_i.w^x_r$ for any bit x, P_1 opens $\langle w^{p_l}_l \rangle \oplus \langle w^{p_{i,l}}_{i,l} \rangle \oplus ((p_l \oplus p_{i,l}) \langle \Delta \rangle)$ and $\langle w^{p_r}_r \rangle \oplus \langle w^{p_{i,r}}_{i,r} \rangle \oplus ((p_r \oplus p_{i,r}) \langle \Delta \rangle)$to P_2 so that P_2 learns d_l and d_r (where $d_l = w^0_l \oplus w^0_{i,l}, d_r = w^0_r \oplus w^0_{i,r}$), hence can compute $w^a_{i,l} := w^a_l \oplus d_l$ and $w^b_{i,r} := w^b_{i,r}$.
         C. Let $T_{i,G} = g_i.T_G$, $T_{i,E} = g_i.T_E$. If P_2 runs $w_{i,o} := \mathsf{EvlAND}(w^a_l, w^b_r, T_{i,G}, T_{i,E})$.

D. P_2 verifies $w_{i,o}$ against $\langle w_{i,o}^{p_{i,o}} \rangle$ and $\langle w_{i,o}^{p_{i,o}} \rangle \oplus \langle \Delta \rangle$. If neither succeeds, P_2 stops processing g_i and advances to processing gate g_{i+1}.

E. Let $p_{i,o} = g_i.p_o$. P_1 opens $p_o \oplus p_{i,o}$ to P_2.

F. Let $w_{i,o} = g_i.w_o$. P_1 opens $\langle w_o^{p_o} \rangle \oplus \langle w_{i,o}^{p_{i,o}} \rangle \oplus ((p_o \oplus p_{i,o})\langle \Delta \rangle)$ to P_2 so that P_2 learns $d_{i,o}$ (note $d_{i,o} \coloneqq w_o^0 \oplus w_{i,o}^0$), hence can compute $w_o \coloneqq w_{i,o} \oplus d_{i,o}$.

G. P_2 verifies w_o against $\langle w_o^{p_o} \rangle$ and $\langle w_o^{p_o} \rangle \oplus \langle \Delta \rangle$. If either verification succeeds, P_2 adds w_o to $\mathcal{O}$; otherwise, continue evaluating the next garbled gate.

iii. If there exist two different labels w_o and w_o' in $\mathcal{O}$. P_2 computes $\Delta^* \coloneqq w_o \oplus w_o'$, and uses Δ^* to find out P_1's private inputs x and computes $f(x, y)$.

iv. Otherwise, i.e., $\mathcal{O}$ contains a single element w, P_2 sets $w_o = w$.

(d) For every XOR gate in C: since XOR is free, P_1 does nothing while P_2 calculates $w_o \coloneqq w_l \oplus w_r$. At the same time, P_2 derives $\langle \rho^o \rangle \coloneqq \langle \rho^l \rangle \oplus \langle \rho^r \rangle$ and $\langle w_o^{p_o} \rangle \coloneqq \langle w_l^{p_l} \rangle \oplus \langle w_r^{p_r} \rangle$.

4. **Check.** P_2 checks the correctness of the rest $T - BN$ garbled ANDs. For every check-gate denoted by $\big(\langle \rho^l \rangle, \langle w_l^{p_l} \rangle, \langle \rho^r \rangle, \langle w_r^{p_r} \rangle, \langle \rho^o \rangle, \langle w_o^{p_o} \rangle, T_G, T_E\big)$,
   (a) P_2 samples $a \leftarrow \{0,1\}, b \leftarrow \{0,1\}$ and sends a, b to P_1.
   (b) P_1 opens $\langle \rho^l \rangle, \langle \rho^r \rangle, \langle \rho^o \rangle$ to P_2. Let $p_l = \rho_1^l \oplus \cdots \oplus \rho_{\lambda_p}^l$, $p_r = \rho_1^r \oplus \cdots \oplus \rho_{\lambda_p}^r$, $p_o = \rho_1^o \oplus \cdots \oplus \rho_{\lambda_p}^o$,
      i. P_1 opens $\langle w_l^{p_l} \rangle \oplus ((a \oplus p_l)\langle \Delta \rangle)$ to P_2 so that P_2 learns w_l^a.
      ii. P_1 opens $\langle w_r^{p_r} \rangle \oplus ((b \oplus p_r)\langle \Delta \rangle)$ to P_2 so that P_2 learns w_r^b.
      iii. Let $z = a \wedge b$. P_1 opens $\langle w_o^{p_o} \rangle \oplus ((z \oplus p_o)\langle \Delta \rangle)$ to P_2 so that P_2 learns w_o^z.
   (c) P_2 computes $w_o \coloneqq \mathsf{EvlAND}(w_l^a, w_r^b, T_G, T_E)$. If $w_o \neq w_o^z$, P_2 goes to step 4d.
   (d) If any failure is detected in step 4(b)ii and 4(b)iii, P_2 keeps dummy-execution of the protocol as normal, until step 5, when it aborts..
5. **Output determination.** If any incorrect garbled AND gates or incorrect hash preimages were found in steps 4 and 3 above, P_2 aborts. If P_1's $f(x, y)$ was determined in step 3(c)iii (from leaked P_1's secret Δ), P_2 outputs $f(x, y)$. Otherwise, (cheating was not observed by P_2), for every final output-wire in C, P_1 opens $\langle \rho^o \rangle$ (the hash of the permutation bit on the final output-wire) to P_2 so that P_2 can find out the value represented by w_o (recall w_o is already either verified with $\langle w_o^{p_o} \rangle$ or $\langle w_o^{p_o} \rangle \oplus \langle \Delta \rangle$, thus once p_o is known from opening ρ^o, w_o can be translated to a plaintext bit).

4.3 Optimizations

Thwart selective-failure attacks, more *efficiently*. In order to prevent P_1 from probing P_2's input y using inconsistent wire labels in OT and garbling, in step 3b, every input-wire for P_2's input is split to s wires (which are XOR-ed together), requiring $n_I^{P_2} s$ 1-out-of-2 oblivious transfers (where $n_I^{P_2}$ is the number of bits in y and s is the statistical parameter). Lindell and Pinkas [35] suggested an optimization using a bundle of wires per circuit to reduce the cost down to $\max(4n_I^{P_2}, 8s)$. Shelat and Shen [53] proposed an optimization based on Reed-Solomon code to reduce the cost to as low as 25% of Lindell and Pinkas's already optimized construction.

Length preserving encryption. In the interactive hash scheme of Figure 4, a Hash requires n calls to the symmetric key encryption. To prevent ciphertext explotion and save bandwidth, the symmetric key secure encryption scheme can be replaced by a secure format-preserving encryption scheme [43,5,45].

4.4 Proof of Security

We first show that, for any security parameter s, k and circuit size N, it is possible to configure the protocol parameters T, B, n, ℓ, w such that if P_2 will be able to correctly output $f(x, y)$ (except with probability 2^{-s}) if it does not abort in our protocol. We break the proof into the following two lemmas. For concrete values of s, k, N, we detail how to setup T, B, n, ℓ, w to achieve the best performance while maintaining the security guarantee.

Lemma 2. *For any circuit size N, there are parameter settings of T, B such that if P_2 does not abort at step 5, then there is at least one correctly garbled gate in each bucket, except with probability 2^{-s}.*

Lemma 3. *If there is at least one correctly garbled gate in each bucket, P_2 is able to output $f(x,y)$ except with negligible probability when executing the two-party computation protocol of Section 4.2.*

Lemma 4. *Given $\langle\Delta\rangle$ and any committed garbled AND, $\big(\langle\rho^l\rangle, \langle w_l^{p_l}\rangle, \langle\rho^r\rangle, \langle w_r^{p_r}\rangle, \langle\rho^o\rangle, \langle w_o^{p_o}\rangle, T_G, T_E\big)$, where $p_l = \rho_1^l \oplus \ldots \oplus \rho_{\lambda_p}^l$, $p_r = \rho_1^r \oplus \ldots \oplus \rho_{\lambda_p}^r$, $p_o = \rho_1^o \oplus \ldots \oplus \rho_{\lambda_p}^o$. If any of the following is not satisfied (where* EvlAND *is the Half-Gate AND gate evaluation algorithm, see Algorithm 2),*

$$\mathsf{EvlAND}(w_l^0, w_r^0, T_G, T_E) = w_o^0; \quad \mathsf{EvlAND}(w_l^0, w_r^1, T_G, T_E) = w_o^0;$$
$$\mathsf{EvlAND}(w_l^1, w_r^0, T_G, T_E) = w_o^0; \quad \mathsf{EvlAND}(w_l^1, w_r^1, T_G, T_E) = w_o^1,$$

P_2 will be able to detect this fact with probability at least 1/2 in step 4.

Theorem 2. *Under the assumptions outlined in Section 2, and modeling H as a random oracle, the protocol in Section 4 securely computes f in the presence of malicious adversaries.*

Proof We analyze the protocol in a hybrid world where the parties have access to an ideal functionality for k-out-of-n oblivious transfer. The standard composition theorem [9] implies security when the sub-routine are instantiated with a secure k-out-of-n OT protocol.

For a corrupted P_1, we construct a polynomial-time simulator $\mathcal{S}$ that interacts with the corrupted P_1 using the protocol specified in Section 4.2 as P_2 with input $y = 0$, except with the following changes:

1. In step 0a of **Setup**, instead of using a real w-out-of-n OT protocol, $\mathcal{S}$ and P_1 use a trusted party (simulated by $\mathcal{S}$) to accomplish the w-out-of-n OT so that $\mathcal{S}$ learns all the symbol-encryption keys $k_1, \ldots, k_n$ that P_1 has generated.
2. In step 3a of **Evaluate**, in addition to receiving $\langle\rho^i\rangle$ and $\langle w_i^{p_i}\rangle$, $\mathcal{S}$ uses the symbol-encryption keys learned above to find out ρ^i, p_i, and $w_i^{p_i}$, which, combined with the correctly garbled gate where $w_i^{p_i}$ is used, allows $\mathcal{S}$ to learn x_i for all $1 \le i \le n_I^{P_1}$.
3. In step 5 of **Output determination**, if $\mathcal{S}$ does not abort, (instead of outputs $f(x,0)$), $\mathcal{S}$ sends x to the trusted party and receive in return an output $z = f(x,y)$.

Now we show that $\forall x,y$, $\mathbb{REAL}^{P_1,P_2}(x,y) \approx \mathbb{IDEAL}^{\mathcal{T},\mathcal{S},P_2}(x,y)$ by examining the following two cases:

1. If P_2 does abort in the real world execution, $\mathcal{S}$ will also abort in the ideal world execution because the difference between $\mathcal{S}$ and P_2 does not affect their behavior of aborts.
2. If P_2 does not abort in the real world execution, assuming the parameters T, B, n, ℓ, w are properly set, following Lemma 2 and Lemma 3, we know that P_1 and P_2 must be able to output $f(x,y)$ in the real world (except with probability less than 2^{-s}. In this case, $\mathcal{S}$ in the ideal world execution will not abort either, so $\mathcal{S}$ will send the extracted x (because $\mathcal{S}$ does not abort, thus except with probability less than 2^{-s}, at least one correctly garbled gate is in every bucket and P_1 had revealed all expected pre-images and expected xors correctly, hence indeed able to extract x) to the trusted party so $\mathcal{S}$ and P_2 can output $f(x,y)$ in the ideal world.

For a corrupted P_2**,** we construct a polynomial-time simulator $\mathcal{S}$ that interacts with the corrupted P_2 using the protocol specified in Section 4.2 as P_1 with input $x = 0$, except with the following changes:

1. In step 0a of **Setup**, instead of using a real w-out-of-n OT protocol, $\mathcal{S}$ and P_1 use a trusted party (simulated by $\mathcal{S}$) to accomplish the w-out-of-n OT so that $\mathcal{S}$ learns all positions of the watched symbols selected by P_2.
2. In step 3b of **Evaluate**, instead of using a real 1-out-of-2 OT protocol, $\mathcal{S}$ and P_1 use a trusted party (simulated by $\mathcal{S}$) to accomplish the 1-out-of-2 OT so that $\mathcal{S}$ learns $y_{i,1}, \ldots, y_{i,s}$, hence able to compute P_2's effective input $y_i \coloneqq y_{i,1} \oplus \cdots \oplus y_{i,s}$ for all $1 \le i \le n_I^{P_2}$.
3. In step 5 of **Output determination**, if $\mathcal{S}$ does not abort, $\mathcal{S}$ sends the y extracted above to the trusted party and receive in return $f(x,y)$. Leveraging the knowledge of all positions of P_2's watched symbols, for every output bit z_i where $f(x,y)$ differs from $f(0,y)$, let ρ be the original permutation randomness on the output-wire associated with z_i , $\mathcal{S}$ opens $\langle\rho\rangle$ to ρ', which differs from ρ in a single bit while matching $\langle\rho\rangle$ in of P_2's perspective.

Now we show that $\forall x, y$, $\mathbb{REAL}^{P_1,P_2}(x,y) \approx \mathbb{IDEAL}^{\mathcal{T},P_1,\mathcal{S}}(x,y)$ by examining the following two cases:

1. If P_2 does abort in the real world execution, $\mathcal{S}$ will also abort in the ideal world execution because P_1 is honest so the only reason that an abort happens is that P_2 decides to leaves the protocol while the decision can only be made over transcripts that are computationally indistinguishable. This computational indistinguishability can be derived from the security of the garbled circuit protocol.
2. If P_2 does not abort in the real world execution, assuming the parameters T, B, n, ℓ, w are properly set, following Lemma 2 and Lemma 3, we know that P_1 and P_2 will be able to output $f(x,y)$ in the real world (except with probability less than 2^{-s}). In this case, the $\mathcal{S}$ runs in the ideal world will not abort either (following the security of the garbling scheme), so $\mathcal{S}$ will be able to extract y and use it to obtain $f(x,y)$ from the trusted party in the ideal world. Note that because $\mathcal{S}$ knows all w symbols of ρ's encoding watched by P_2 and $\ell > w$ (where ℓ is the number of symbols in rho'), a ρ' that matches $\langle\rho\rangle$ but differs from ρ in a single bit can be efficiently calculated from solving a linear system of $w+1$ equations. $\square$

5 Parameters and Analysis

Our protocol involves two flavors of cut-and-choose (one in the XOR-homomorphic interactive hash and the other for gate checking and evaluation) and many parameters. It involves novel approaches of analysis to figure out the best (in terms of efficiency) set of parameters subject to a concrete security guarantee. In this section, we propose a unique approach to efficiently automate the search for optimal parameter settings for every concrete security parameters (s,k).

Cost Metric. In this work, we focus on the cost of network bandwidth, which can be accurately calculated in our analysis. As faster hardware [3,32], new garbling mechanisms [18,3], and circuit parallelism [7,46,24,32] have been exploited to dramatically improve the speed of garbled circuit protocols, the high bandwidth cost of these protocols, especially in the malicious model, is increasingly prominent and widely recognized by the research community. In practice, bandwidth has been shown to be highly correlated with the time cost as a significant portion of the time is spent on data transmission and synchronization.

5.1 Optimal Parameters of the Interactive Hash

Here our goal is to determine the most bandwidth-efficient configuration of (σ,ℓ,n,w) that achieves s bits statistical security in binding and retains μ bits entropy per hashed message from the hash receiver. We model it as a non-linear optimization problem given in Figure 5. We introduce constants α,β to characterize the bandwidth cost in the main protocol associated with the parameters of the interactive hash scheme. We use $\alpha = 3, \beta = 3$ when searching for the parameters of interactive hash used on wire permutation randomness because every garbled AND gate comes with 3 wires, each having a random string to encode the permutation bit (so $3n\sigma$ bits sent in Hash and $3\ell\sigma$ sent in "open" the XOR difference); while $\alpha = 3, \beta = 5$ when determining the parameters for hashing the wire labels (note $\beta = 5$ because the two garbled rows have to be extended to $\ell\sigma$ bits each to enable P_2 to verify the output wire labels on its own.)

A naive brute-force search of all possible (σ,ℓ,n,w) values can hardly solve the non-linear programming problem because, apparently, the value ranges of n and ℓ appear to be quite large, whereas there are no explicit upper-bounds for σ and ℓ.

To this end, we identified a highly efficient solution through aggressive pruning (Figure 6). Our basic strategy is to scan through candidate S values from the bottom and return as soon as a viable (σ,ℓ,n,w) is found. We had two key insights that helps to dramatically speedup our search: (1) Fixing $S = S_0$ and $\sigma = \sigma_0$, if $w := \lfloor \ell - k/\sigma_0 \rfloor$ can't satisfy (1) then there cannot be a viable solution with $\sigma = \sigma_0$ that achieves S_0 (which saves us from trying out all possible w but only one); (2) Fixing $S = S_0, \sigma = \sigma_0, w = w_0$, because $\ell = (S_0/\sigma_0 - \alpha n)/\beta$, $\binom{\ell-1}{w} / \binom{n}{w}$ is a *convex* function of n and its value is solely determined by n (which allows us to decide whether a viable solution of (σ_0, w_0, ℓ, n) exists for S_0 by evaluating $\binom{\ell-1}{w} / \binom{n}{w}$ at logarithmic number of n values). Overall, given $S = S_0$ and $\sigma = \sigma_0$, it takes only logarithmic number of steps to search in a one-dimensional space to decide whether a viable solution achieving S_0 exists with $\sigma = \sigma_0$. We formally prove this as Lemma 5.

Let s, μ, α, β be fixed integer constants, $S = (\alpha n + \beta \ell)\sigma$.

$$\min S$$

subject to:

$$\binom{\ell - 1}{w} \Big/ \binom{n}{w} \leq 2^{-s} \tag{1}$$

$$\sigma(\ell - w) \geq \mu \tag{2}$$

$$2^{\sigma} \geq n \tag{3}$$

where n, ℓ, σ, w are all positive integers.

Fig. 5: The non-linear programming problem for optimizing the interactive hash's parameters. ((1) comes from part 2 of Lemma 1, (2) is related to part 1 of Lemma 1, (3) stems from the requirement on the encoding vectors (Figure 4, Step 4 of the Setup stage).)

```
1. Let $S = (\alpha n + \beta \ell)\sigma$. For $S$ ranging from 2 to infinity (but skipping
   all odd primes)
   (a) For $\sigma$ ranging from all possible divisors of $S$
       i. Set $w := \lfloor \ell - \mu/\sigma \rfloor$, $\ell := (S/\sigma - \alpha n)/\beta$, then search with
          discrete Newton algorithm for the integer $n$ that minimizes
          $Q(n) = \binom{\ell-1}{w} / \binom{n}{w}$.
       ii. If the current assignments of $\sigma, \ell, n, w$ satisfy (1), output
          $(\sigma, \ell, n, w)$ and stop; otherwise, continue to examining the
          next possible $\sigma$.
```

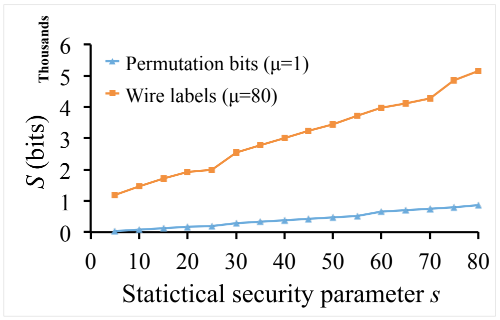
Fig. 6: Algorithm to find the bandiwidth-optimal parameters σ, ℓ, n, w of XOR-homomorphic interactive hash for every requirement specification s, μ, α, β.

Lemma 5. *Fixing $S = S_0$ and $\sigma = \sigma_0$, it takes only logarithmic number of steps to search in the one-dimensional space of n, to decide whether a viable solution (σ, w, ℓ, n) to the non-linear programming problem of Figure 5 achieving S_0 with $\sigma = \sigma_0$.*

Leveraging this parameter optimization algorithm, we plot the influence of the security parameters s, k over the bandwidth cost related to XOR-homomorphic interactive hash in our secure two-party computation protocol in Figure 7,

5.2 Fast Computation of $\mathbf{Pr}_{overall}(N, B, T, \tau, b)$

Computing the optimal parameters T, B based on s, k, N, τ is an even challenging non-linear programming problem. Our solution to that problem requires a highly efficient and accurate approach to compute the overall success probability of P_1. In this section, we describe our novel algorithm that computes $\Pr_{overall}(N, B, T, \tau, b)$ with unprecedented efficiency and accuracy.



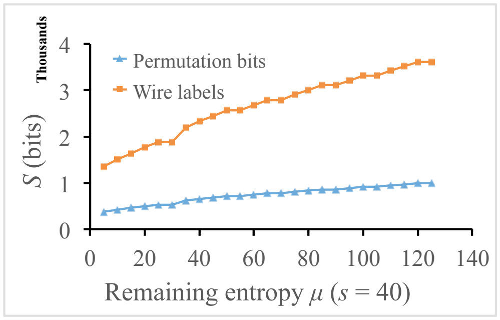


Fig. 7: Per gate bandwidth cost of XOR-homomorphic interactive hashes for two types of messages. (The cost is roughly linear in the security parameter s and the amount of remaining entropy μ, except for a few periodic jumps due to increased σ.)

Once the parameters N, B, T, τ, b are fixed, the exact probability of malicious P_1's attack success rate could be straightforwardly described as,

$$\Pr_{overall}(N,B,T,\tau,b) = \sum_{i=0}^{b} \Pr_c(N,B,T,\tau,b,i)\Pr_e(N,B,b-i)$$

$$\Pr_c(N,B,T,\tau,b,i) = (1-\tau)^i \frac{\binom{b}{i}\binom{T-b}{T-BN-i}}{\binom{T}{T-BN}} \tag{4}$$

$$\Pr_e(N,B,b) = \binom{b}{B} \bigg/ \binom{BN}{B} + \sum_{i=0}^{b-1} \Pr_e(N-1,B,b-i)\cdot \frac{\binom{b}{B-i}\binom{BN-b}{i}}{\binom{BN}{B}} \tag{5}$$

$$\Pr_e(N,B,b) = 0, \qquad \forall 0 \le b \le B$$

However, due to the large scale of N, T values (e.g., $N > 2^{30}$) and stringent requirement on the precision, e.g. end results less than $2^{40} \sim 2^{-80}$ (determined by the statistical security parameter), it is infeasible to accurately compute $\Pr_c$ (which involves calculating large binomial coefficients) and $\Pr_e$ (which involves exponential number of slow recursions) based on (4) and (5).

Thus, we propose novel approaches to compute $\Pr_e$ and $\Pr_c$, which naturally combines to yield a highly efficient and accurate approximation for $\Pr_{overall}(N,B,T,\tau,b)$. E.g., set $s = 40$, if $N = 50{,}000$, the error in our approximation of $\log \Pr_{overall}(N,B,T,\tau,b)$ (on any secure choice of (T,B) assuming $T > BN + 0.05T$) is less than 1 bit, while it decreases as N increases (following Lemma 6 and Lemma 7).

Computing $\Pr_e(N,B,b)$ Our key idea is to use *generating functions* to efficiently calculate $\Pr_e$ as the ratio between the count of garbled gate assignments resulting a failure (i.e., at least one bucket is filled with B bad gates) and the total count of garbled gate assignments. First, we can use function $g(x,y) = (1+x)^B + (y-1)x^B$ to model the gate assignment process for a single bucket, where x denotes a gate is "bad" and 1 denotes a gate is "good" while the coefficient of x^i is the number of ways to assign i bad gates to a bucket. Note that we explicitly introduce the coefficient y for x^B to denote the event that "all B gates in a bucket are bad". Furthermore, we can use $G(x,y) = g(x,y)^N$ as the generating function to model the gate assignment process on the whole circuit with N buckets: the coefficient (which is a polynomial in y, hence written as $f_i(y)$) of x^i in $G(x,y)$ denotes the number of assignments that used i bad gates in total. Let $f_i(y) = \sum_{j=0}^{\infty} c_j y^j$, then $f_b(1) = \sum_{j=0}^{\infty} c_j$ is the total number of assignments with b bad gates used in the evaluation stage whereas $f_b(1) - f_b(0) = \sum_{j=1}^{\infty} c_j$ denotes the number of assignments that resulted at least one broken bucket. Hence, we calculate $\Pr_e(N,B,b) = [f_b(1) - f_b(0)]/f_b(1)$.

```
Input: $s, N, r_0$.
Output: $T_{min}, B_{T_{min}}$

1. Choose $b^u$ and $B^u$.
    (a) Compute $i_0 \coloneqq s+2$ and $b^u \coloneqq -(s+1) \Big/ \log(\frac{r_0}{2} + \frac{2}{2/(1-r_0)-i_0/N})$
    (b) Set $T_{min} \coloneqq \infty$ and $B^u \coloneqq \infty$.
2. For $B$ ranging from 2 to $B^u$,
    (a) Precompute $\Pr_e(N, B, b) = 1 - f_b(0)/f_b(1)$ for all integer $b \in [B, b^u]$.
    (b) Find the smallest $T_B$ such that $\Pr_{overall}(N, B, T_B, \tau, b) < 2^{-s}$
        for all $b \le b^u$. Note that such a $T_B$ exists for every $B \ge 2$.
        Since $\Pr_{overall}(N, B, T, \tau, b)$ is a monotonically decreasing function
        of $T$, the search of $T_B$ is highly efficient with a doubling-and-halving
        probe strategy.
    (c) If $T_{min} > T_B$, then update $T_{min} \coloneqq T_B$, $B_{T_{min}} \coloneqq B$, and $B^u \coloneqq \lceil (1-r_0)T_B/N \rceil$.
3. Output $T_{min}, B_{T_{min}}$.
```

Fig. 8: Determine bucket size (B) and the total number of gates to garble (T).

Note that we can further dramatically reduce the cost of computing the coefficients of the polynomial $G(x, y)$, by not distinguishing any terms y^{j_1} and y^{j_2} for any $j_1, j_2 \ge 1$. Therefore, multiplying $(u + vy)$ and $(w + ty)$ yields $uw + (ut + vw + vt)y$, keeping all c_js a linear formula of y no matter how big N and B are.

Computing $\Pr_c(N, B, T, \tau, b, i)$ Recall that typically T, N are large while b, i are far smaller than N. So the dominating cost in computing $\Pr_c$ is due to calculating $\binom{T-b}{T-BN-i} \Big/ \binom{T}{T-BN}$. Fortunately, we show that, according to Lemma 6, it can be accurately approximated by $\left(\frac{T-BN}{T}\right)^i \left(\frac{BN}{T-i}\right)^{b-i}$.

Lemma 6. *Let T, B, N, b, i be defined as above. Then*

$$\lim_{N->\infty} \frac{\binom{T-b}{T-BN-i}}{\binom{T}{T-BN}} = \left(\frac{T-BN}{T}\right)^i \left(\frac{BN}{T-i}\right)^{b-i}.$$

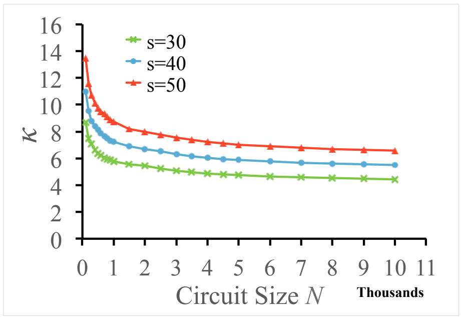
5.3 Parameters of Gate Checking and Bucketing

Here the goal is, for every statistical security requirement s and circuit size N, efficiently identify the smallest T that guarantees s bits of statistical security for a circuit of N buckets. To ensure security, a naive solution requires, for a candidate (T, B) configuration, calculating $\Pr_{overall}$ for all b ranging from 1 to T. In practice, this works only for circuits of moderate size, e.g., under a million. When N is large, this does not work due to the daunting cost of computing (and memorizing) the coefficients $f_j(y)$ in the generating function $G(x, y)$ for all $i \le T$ (as b is only loosely upper-bounded by T).

Thus, we consider a relaxed but practically meaningful problem for cases when N is large. Namely, assuming the check rate $r = (T - BN)/T > r_0$ for a positive constant r_0, find the (T, B) pair with minimal T (except for an asymptotically diminishing error due to approximating $\Pr_c$). Intuitively, strategies with low check rate do not perform well as a malicious P_1 would exploit it by injecting more bad gates, increasing its probability of successful attacks. In practice, we can set r_0 to however small positive values to attain stronger claims on the optimality of the T value output by the search.

A key insight of our (T, B)-searching algorithm (Figure 8) is that we can significantly save the cost of computing $G(x, y)$ by bounding b based on r_0: because we can show that if P_2 sets $r > r_0$, there exists a b_0 such that all of P_1's strategies using $b > b^u$ will be *dominated* by those with $b \le b^u$, thus we don't even bother to compute the coefficients of x^j in $G(x, y)$ for all $j > b^u$. We formally prove that a lower-bound of r implies a upper-bound of b as Lemma 7. Our algorithm finds the smallest T by enumerating all possible B values. Another useful observation, which enables to effectively prune our search space, is that a smaller viable T_B we found on the way stipulates an upper-bound on B values we need to enumerate.

Using the efficient search algorithm above, we discovered the optimal T for a range of N, s settings. Figure 9 depicts the the optimal $\kappa = T/N$ with respect to different N and s.



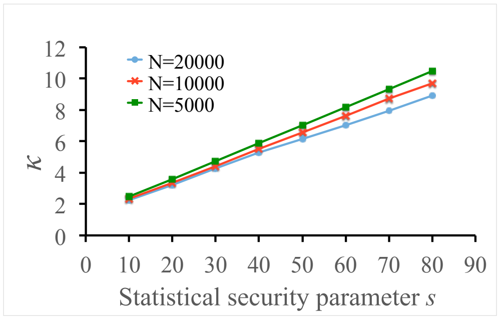


Fig. 9: Optimal κ with respect to different N and s values. (When $s = 40$, $\kappa = 6$ is easily achievable for circuits with more than 6000 gates. Note κ increases almost linearly with the statistical security parameter s.)

Lemma 7. *For any $s, r_0, \tau > 0$, there exists N_0 and b^u such that if $T \geq BN/(1 - r_0)$, then*

$$\Pr_{overall}(N, B, T, \tau, b) < 2^{-s}, \qquad \forall b > b^u.$$

5.4 Compare to Existing Two-Party Computation Protocols

Since our protocol has constant rounds whereas NNOB [47] and SPDZ [12] requires linear number of rounds (theoretically in circuit depth but practically in circuit size because only a bounded number of gates can be held in memory) and a heavy-weighted preparation stage, ours performs better for large scale computation when the impact of network latency is taken into account.

Our protocol easily beats any existing cut-and-choose based two-party computation protocols (such as [35,52,36,53,42]) that requires $3s$ copies of circuits to achieve s-bit of statistical security on any not-too-small circuits (e.g., $N \geq 6000$), because, say $s = 40$, the bandwidth of using 120 copies of the circuit is equivalent to a κ value of $120 * 128 * 2/4844 = 6.34$, while as Figure 9 shows, we can achieve $\kappa = 5.8$ for any circuit of size $N \geq 6000$.

Next, we consider The Lindell (CRYPTO, 2013)'s cheat-then-reveal protocol [33], the only protocol that achieves s bits of security using s copies of circuits in the non-amortized execution scenario. We will show that our protocol can outperform theirs in terms of bandwidth cost either when the $n_I^{P_1}/N$ ratio is large, or when N is sufficiently large.

First of all, we calculate the bandwidth cost of our protocol for $(s = 40, k = 128)$ as follows:

1. Determine the optimal parameter setting $n = 100, \ell = 46, w = 27, \sigma = 7$ for hashing wire labels. That is, it costs $n\sigma = 700$ bits to hash a wire label and $\ell\sigma = 322$ bits to reveal a wire label (or an XOR of several wire labels).
2. Determine the optimal parameter setting $n = 44, \ell = 19, w = 18, \sigma = 6$ for hashing permutation bits. So it costs $n\sigma = 264$ bits to hash a permutation bit and $\ell\sigma = 114$ bits to reveal a permutation bit (or an XOR of several permutation bits).
3. Each row in the garbled table should have 322 bits since the wire labels are extended.
4. To check a gate, the total bandwidth cost is $114 \times 3 + 322 \times 2 = 986$ because three permutation bits and two wire labels needs to be revealed.

5. To evaluate a gate, the total bandwidth cost is $114 \times 3 + 322 \times 3 = 1308$ because all three wires needs to be soldered once, each of which requires revealing an XOR of two permutation bits and an XOR of two wire labels.
6. Let $n_I^{P_1}$ be the bit length of x, P_1 needs to send a $\ell\sigma n_I^{P_1}$ bits in Step 3a; Let $n_I^{P_2}$ be the bit length of y, we use $\log n_I^{P_2} + n_I^{P_2} + s + s \cdot \max\left(\log(4n_I^{P_2}), \log(4s)\right)$ number of actively secure oblivious transfers (where efficient actively secure OT extension [27,1] can be used) in Step 3b following Shelat-Shen's optimization on handling P_2's input [53].

Taking a conservative estimation, every garbled gate will cost no more than $(700 + 264) \times 3 + (322 + 114) \times 3 + 322 \times 2 = 4844$ bits, no matter whether the gate is used for checking or evaluation. Note that Lindell (CRYPTO, 2013)'s protocol [33] will generate $128 \times 2 \times 40 = 10240$ bits of traffic per gate (excluding the cost of the special handling of input and output wires). Since the same overhead allows us to implement $5120/4844 = 2.11$ gates per bucket. Therefore, according to the results of Section 6.2, setting κ to any value between 2 and 2.11, our approach can outperform theirs when N is sufficiently large.

When N is not sufficiently large, the cost of proper treatment of the initial input wires cannot be ignored. In this case, our protocol has significant advantage over the state-of-the-art actively secure two-party computation protocols, since ours requires only sending a $\ell\sigma$-bit message per bit of P_1's input x, whereas the state-of-the-art protocols spend significantly more bandwidth in the input consistency proofs. Let $c_{\text{ours}}(n_I)$ and $c_{\text{prev}}(n_I)$ be the cost of handling the initial inputs in our work and a previous work, respectively. Let $c_{\text{N,ours}}$ and $c_{\text{N,prev}}$ be the per gate cost of our work and a previous work, respectively. Our protocol will outperform when

$$c_{\text{ours}}(n_I) + c_{\text{N,ours}} \cdot N < c_{\text{prev}}(n_I) + c_{\text{N,prev}} \cdot N,$$

that is, when $c_{\text{N,ours}} < c(n_I)/N + c_{\text{N,prev}}$ where $c(n_I) = c_{\text{prev}}(n_I) - c_{\text{ours}}(n_I)$. Because $c_{\text{N,prev}}$ is a constant solely determined by the security parameters while $c_{\text{N,ours}}$ decreases as N grows, for any fixed $c(n_I)/N$, our protocol will outperform traditional cut-and-choose protocols if N is larger than a threshold $N_{\text{break-even}}$. Figure 10 quantitatively depicts $N_{\text{break-even}}$ with respect to $c(n_I)/N$. Our protocol uses less bandwidth for all $(c(n_I)/N, N)$ configurations in the area above the curve.

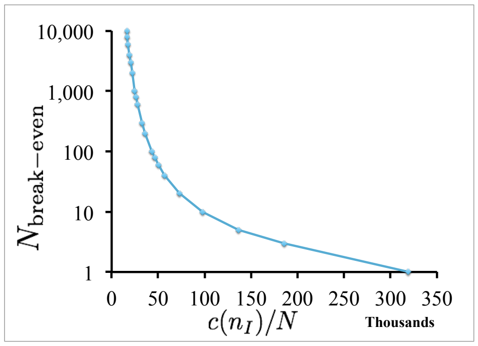


Fig. 10: Quantifying $N_{\text{break-even}}$ with respect to $c(n_I)/N$, the cost ratio of input handling and circuit size.

Comparing to Lindell (CRYPTO, 2013)'s protocol [33], we have significant performance advantage on processing input-wires corresponding to the P_1's input, because in addition to the input consistency proof, they require actively-secure computing a cheating-punishment circuit whose bandwidth cost grows with the length of P_1's input and is orders-of-magnitude more expensive than ours. (We stress that although they proposed an optimization that effectively reduces the size of the cheating-punishment circuit, it does not reduce the number of input-wires, which equals to the number of bits in a wire label). For many interesting linear and sub-linear algorithms, $c(n_I)/N$ for Lindell (CRYPTO, 2013)'s protocol is so large that ours has clear advantage, e.g., $c(n_I)/N$ is 50K bits for computing the Hamming distance between two strings ($(n_I^{P_1} = n_I^{P_2} = N/2)$, 101K bits for Integer comparison ($n_I^{P_1} = n_I^{P_2} = N$), and 3K bits for private AES cipher ($n_I^{P_1} = n_I^{P_2} = 128, N \approx 10K$). So, our protocol can easily outperform theirs for Hamming distance and Integer comparison but will not be more bandwidth-efficient for computing private AES when $N \leq 10^{10}$. In general, our scheme loses to Lindell (CRYPTO, 2013) for most super-linear algorithms (except for sufficiently large N), whose $c(n_I)/N$ values are small.

Most recently, Arash-Mohassel-Pinkas-Riva [2] proposed a lightweight cheating-punishment technique based on Lindell's and applied a bandwidth-saving technique that requires little more than 19 copies of garbled circuits being transmitted to obtain 40-bit security. Our protocol cannot outperform theirs in bandwidth but it would be an interesting and promising research to develop XOR-homomorphic interactive hash schemes that are more bandwidth-efficient to outperform theirs.

6 The Tight Bound on The Benefit

In LEGO-protocols, $\kappa = T/N$ (could be a decimal). When the circuit size N approaches to infinity, (following the efficiency claim in LEGO-style protocols [13,48,14]) one may intuitively think κ goes to 0, as $O(\kappa) = O(T/N) = O((skN/\log N) \cdot 1/N) = O(sk/\log N)$. However, this is not the case! We will show shortly that two is the tight bound of κ, i.e., $\kappa \leq 2$ can never be securely achieved by any LEGO-protocol; while any $\kappa > 2$ can be achieved (at least with the protocol proposed in this paper) when N is sufficiently large.

In the proof of this section, we generalize the bucket size B to decimal values, which suggests different buckets can have different (integer) size.

6.1 Any $\kappa \leq 2$ Is Impossible to Achieve

Let $\Pr_{overall}$ be the overall success rate of P_1 attacking a LEGO-style protocol. Our goal is to show that any LEGO-style protocol will be insecure in an asymptotic sense. Namely, there exists a positive constant c and N_0 such that $\Pr_{overall} > c$ holds for all circuit sizes larger than N_0 (Theorem 3). We achieve this by first restricting our attention to the insecurity of LEGO-style protocols with $\kappa = 2$ that use only size-1 and size-2 buckets (Lemma 8), which can be proved through a sequence of mathematical relaxation (Lemma 10, 11, 12) on the success probabilities of a malicious generator. Then we generalize the result of Lemma 8 to schemes of arbitrary size buckets (Lemma 9) and settings of $\kappa < 2$ (Theorem 3).

Theorem 3. *If $\kappa \leq 2$, there exist a constant $c > 0$ and an integer N_0 such that $\Pr_{overall} > c$ for all $N > N_0$.*

Proof First, we will show a proposition Prop that if $\kappa = 2$, there exist a constant $c > 0$ and an integer N_0 such that $\Pr_{overall} > c$ for all $N > N_0$. This can be derived directly through combining the facts of Lemma 8 and Lemma 9: it suffices to consider schemes involving only size-1 and size-2 buckets because, according to Lemma 9, they work at least as good as any scheme with the same (T, N) but using other bucket sizes; while Lemma 8 proves Prop with respect to using buckets of sizes 1 and 2.

Second, to generalize Prop to $\kappa < 2$, it suffices to see that comparing to the setting of $\kappa = 2$, with the circuit size N being fixed, reducing κ leads to a smaller T, hence some garbled gates have to be dropped either in the verification stage, or in the evaluation stage, or both, all of which further undermine the overall security guarantee of the original cut-and-choose scheme. Therefore, Prop also holds for all $\kappa < 2$. $\square$

Lemma 8. *Assume only size-1 buckets and size-2 buckets are used. Let $\Pr_{overall}(N, x, b)$ be the probability that a cheating P_1, who generates b bad gates, successfully attacks a LEGO-style protocol that computes a circuit with N buckets, of which x percent is of size 2 and $(1-x)$ percent is of size 1 (hence $x \in [1/N, 1-1/N]$). If $\kappa = 2$, there exists a constant $c > 0$ and an integer N_0 such that $\Pr_{overall}(N, x, b) > c$ for all $N > N_0$.*

Lemma 9. *If $\kappa = 2$ and T, N are fixed, a scheme that uses only size-1 and size-2 buckets results in lower rate of successful attacks by a malicious generator than one using any number of buckets whose size is greater than 2.*

Lemma 10. *Fix $\kappa = 2$ and assume a total of N buckets are used, of which x percent is of size 2 and $(1-x)$ percent is of size 1. Let $\Pr_c(N,x,b)$ be the probability that P_1 survives the gate verification stage, generating b bad gates. Then*

$$\Pr_c(N,x,b) \geq \left(\frac{1+x}{2}\right)^b \left[1 - \frac{(1-x)b^2}{(1+x)(2N-b)}\right].$$

Lemma 11. *Assume a total of N buckets are used, of which x percent is of size 2 and $(1-x)$ percent is of size 1. Let $\Pr_e(N,x,b)$ be the probability of P_1 successfully cheats in the evaluation stage, with b bad gates selected for evaluation. Then*

$$\Pr_e(N,x,b) \geq 1 - \left(\frac{2x}{1+x}\right)^b$$

Lemma 12. *There exists a constant $c > 0$ and a constant N_0 such that for all integer $N > N_0, \forall x \in [\frac{1}{N}, 1-\frac{1}{N}]$, there exists a positive integer b such that*

$$\left(\frac{1+x}{2}\right)^b \left(1 - \frac{(1-x)b^2}{(1+x)(2N-b)}\right) \left(1 - \frac{2x}{1+x}\right)^b > c$$

Last, we remark that our discussion above only concerns the *asymptotic* notion of security. Therefore, the aforementioned theorem and proofs should not be interpreted as contradiction to the fact that a particular $\kappa \leq 2$ could work with certain *concrete* security parameter settings, e.g., a $\kappa \leq 2$ may be achievable for 3 bits of statistical security. However, we stress that any $\kappa \leq 2$ can never be achieved with more than 5 bits of statistical security, evidenced by picking $c_2 = 0.48, c_3 = 0.13, c_1 = 0.85$, and $N_0 = 3$ in the proof of Lemma 12 such that $c > 0.05$ while $\log 0.05^{-1} \approx 4.32$.

6.2 Any $\kappa > 2$ Is Achievable

When N is sufficiently enough, LEGO-style protocols can indeed reduce the circuit duplication factor κ to values arbitrarily close to 2. Our proof of this fact (Theorem 4) is constructive, which essentially uses the protocol we will explain in Section 4 hence inherits its proof of security (see Section 4.4) except for a claim on the feasibility of upper-bounding a malicious generator's probability of successful attacks, which we prove here as Lemma 13. The proof is based on a simple cut-and-choose scheme where all buckets are of size 2 and $T - 2N$ gates are used for verification. The key idea is to show that, for a fixed $\kappa > 2$ and a (however small) constant ε, we can find an integer N_0 such that for all $N > N_0$, a malicious generator's successful attack probability is upper-bounded by ε.

Theorem 4. *For any $\kappa > 2$, statistical security parameter s and computational security parameter k, there exist an integer N_0 and a LEGO-like two-party computation protocol Π^{κ}_{LEGO} of circuit duplication factor κ such that for Π^{κ}_{LEGO} is actively-secure for all circuits of size $N > N_0$.*

Lemma 13. *Assume that, to compute a boolean circuit of size N, P_1 generates a total of T garbled gates, $T - 2N$ of which are used for gate verification. Assume all buckets used in evaluation stage are of size 2. Let $\Pr_{overall}(N,b)$ be the probability that P_1 succeeds in an attack by generating b bad gates. For any $\kappa > 2$ and any $\varepsilon > 0$, there exists N_0 such that*

$$\Pr_{overall}(N,b) < \varepsilon, \quad (\forall\, N > N_0)$$

Lemma 14. *Let $\Pr_e(N,b)$ be the probability that, conditioned on passing the verification stage, P_1 succeeds with b bad gates actually used in the evaluation stage. Then*

1. *For every fixed N, $\Pr_e(N,b)$ is strictly* increasing *with respect to b.*
2. *For any b and $\varepsilon > 0$, there exists an N_0 such that $\Pr_e(N,b) < \varepsilon$.*

Lemma 15. *If T, N, b, i are non-negative integers such that $T > 2N$, $T \geq b$, and $i \leq b$, then*

$$\frac{\binom{T-b}{T-2N-i}}{\binom{T}{T-2N}} \leq \left(\frac{T-2N}{T}\right)^i \left(\frac{2N}{T-i}\right)^{b-i}.$$

Acknowledgememnt

Yan Huang would like to thank Feng-Hao Liu for helpful discussions on some security proofs, and thank Jesper Nielsen for an initial chat that provoked this research topic. This work is supported by NSF award #1464113.

References

1. G. Asharov, Y. Lindell, T. Schneider, and M. Zohner. More efficient oblivious transfer extensions with security for malicious adversaries. *Advances in Cryptology – EUROCRYPT*, 2015.
2. A. Afshar, P. Mohassel, B. Pinkas, and B. Riva. Non-interactive secure computation based on cut-and-choose. *Advances in Cryptology – EUROCRYPT*, 2014.
3. M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway. Efficient garbling from a fixed-key blockcipher. *IEEE Symposium on Security and Privacy*, 2013.
4. M. Bellare, V. T. Hoang, and P. Rogaway. Foundations of garbled circuits. *19th Conference on Computer and Communications Security*, 2012.
5. M. Bellare, T. Ristenpart, P. Rogaway, and T. Stegers. Format-preserving encryption. *SAC: 16th Annual International Workshop on Selected Areas in Cryptography*, 2009.
6. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. *1st Conference on Computer and Communications Security*, 1993.
7. N. Buescher and S. Katzenbeisser. Faster secure computation through automatic parallelization. *24th USENIX Security Symposium*, 2015.
8. J. Camenisch, G. Neven, and a. shelat. Simulatable adaptive oblivious transfer. *Advances in Cryptology – EUROCRYPT*, 2007.
9. R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 2000.
10. H. Chen and R. Cramer. Algebraic geometric secret sharing schemes and secure multi-party computations over small fields. *Advances in Cryptology – CRYPTO*, 2006.
11. I. Damgård, M. Keller, E. Larraia, C. Miles, and N. P. Smart. Implementing AES via an actively/covertly secure dishonest-majority MPC protocol. *International Conference on Security in Communication Networks*, 2012.
12. I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. *Advances in Cryptology – CRYPTO*, 2012
13. T. K. Frederiksen, T. P. Jakobsen, J. Nielsen, P. S. Nordholt, and C. Orlandi. MiniLEGO: Efficient secure two-party computation from general assumptions. *Advances in Cryptology – EUROCRYPT*, 2013.
14. T. K. Frederiksen, T. P. Jakobsen, J. Nielsen, and R. Trifiletti. TinyLEGO: An interactive garbling scheme for maliciously secure two-party computation. ePrint Archive, 2015. http://eprint.iacr.org/2015/309.
15. J. A. Garay, P. MacKenzie, and K. Yang. Efficient and universally composable committed oblivious transfer and applications. *Theory of Cryptography Conference*, 2004.
16. O. Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.
17. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. *ACM Symposium on Theory of Computing*, 1987.
18. S. Gureron, Y. Lindell, A. Nof, and B. Pinkas. Fast garbling of circuits under standard assumptions. *Conference on Computer and Communications Security*, 2015.
19. W. Henecka, S. Kögl, A.-R. Sadeghi, T. Schneider, and I. Wehrenberg. TASTY: tool for automating secure two-party computations. *Conference on Computer and Communications Security*, 2010.
20. A. Holzer, M. Franz, S. Katzenbeisser, and H. Veith. Secure two-party computations in ANSI C. *Conference on Computer and Communications Security*, 2012.
21. Y. Huang, D. Evans, J. Katz, and L. Malka. Faster secure two-party computation using garbled circuits. *USENIX Security Symposium*, 2011.
22. Y. Huang, J. Katz, and D. Evans. Efficient secure two-party computation using symmetric cut-and-choose. *Advances in Cryptology – CRYPTO*, 2013.
23. Y. Huang, J. Katz, V. Kolesnikov, R. Kumaresan, and A. J. Malozemoff. Amortizing garbled circuits. *Advances in Cryptology – CRYPTO*, 2014.
24. N. Husted, S. Myers, A. Shelat, and P. Grubbs. GPU and CPU parallelization of honest-but-curious secure two-party computation. *Annual Computer Security Applications Conference*, 2013.

25. Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer - efficiently. *Advances in Cryptology – CRYPTO*, 2008.
26. S. Jarecki and V. Shmatikov. Efficient two-party secure computation on committed inputs. *Advances in Cryptology – EUROCRYPT*, 2007.
27. M. Keller, E. Orsini, and P. Scholl. Actively secure OT extension with optimal overhead. *Advances in Cryptology – CRYPTO*, 2015.
28. M. S. Kiraz and B. Schoenmakers. A protocol issue for the malicious case of yao's garbled circuit construction. *Symposium on Information Theory in the Benelux*, 2006.
29. M. S. Kiraz, B. Schoenmakers, and J. Villegas. Efficient committed oblivious transfer of bit strings. *International Conference on Information Security*, 2007.
30. V. Kolesnikov and T. Schneider. Improved garbled circuit: Free XOR gates and applications. *International Colloquium on Automata, Languages and Programming*, 2008.
31. B. Kreuter, B. Mood, A. Shelat, and K. Butler. Pcf: A portable circuit format for scalable two-party secure computation. *USENIX Security Symposium*, 2013.
32. B. Kreuter, A. Shelat, and C. hao Shen. Billion-gate secure computation with malicious adversaries. *USENIX Security Symposium*, 2012.
33. Y. Lindell. Fast cut-and-choose based protocols for malicious and covert adversaries. *Advances in Cryptology – CRYPTO*, 2013.
34. Y. Lindell, E. Oxman, and B. Pinkas. The IPS compiler: Optimizations, variants and concrete efficiency. *Advances in Cryptology – CRYPTO*, 2011.
35. Y. Lindell and B. Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. *Advances in Cryptology – EUROCRYPT*, 2007.
36. Y. Lindell and B. Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. *Journal of Cryptology*, 2012.
37. Y. Lindell and B. Riva. Cut-and-choose Yao-based secure computation in the online/offline and batch settings. *Advances in Cryptology – CRYPTO*, 2014.
38. Y. Lindell and B. Riva. Blazing fast 2pc in the offline/online setting with security for malicious adversaries. *Computer and Communication Security*, 2015.
39. C. Liu, Y. Huang, E. Shi, J. Katz, and M. W. Hicks. Automating efficient RAM-model secure computation. *IEEE Symposium on Security and Privacy*, 2014.
40. D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay — a secure two-party computation system. *USENIX Security Symposium*, 2004.
41. P. Mohassel and M. Franklin. Efficiency tradeoffs for malicious two-party computation. *International Conference on Theory and Practice of Public Key Cryptography*, 2006.
42. P. Mohassel and B. Riva. Garbled circuits checking garbled circuits: More efficient and secure two-party computation. *Advances in Cryptology – CRYPTO*, 2013.
43. B. Morris, P. Rogaway, and T. Stegers. How to encipher messages on a small domain. *Advances in Cryptology – CRYPTO*, 2009.
44. M. Naor and B. Pinkas. Efficient oblivious transfer protocols. *ACM-SIAM Symposium on Discrete Algorithms*, 2001.
45. M. Naor and O. Reingold. On the construction of pseudorandom permutations: Luby-Rackoff revisited. *Journal of Cryptology*, 1999.
46. K. Nayak, X. S. Wang, S. Ioannidis, U. Weinsberg, N. Taft, and E. Shi. GraphSC: Parallel secure computation made easy. *IEEE Symposium on Security and Privacy*, 2015.
47. J. Nielsen, P. S. Nordholt, C. Orlandi, and S. S. Burra. A new approach to practical active-secure two-party computation. *Advances in Cryptology – CRYPTO*, 2012.
48. J. Nielsen and C. Orlandi. LEGO for two-party secure computation. *Theory of Cryptography Conference*, 2009.
49. C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. *Advances in Cryptology – CRYPTO*, 2008.
50. B. Pinkas, T. Schneider, N. P. Smart, and S. C. Williams. Secure two-party computation is practical. *Advances in Cryptology – ASIACRYPT*, 2009.
51. I. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society of Industrial and Applied Mathematics*, 1960.
52. a. shelat and C.-H. Shen. Two-output secure computation with malicious adversaries. *Advances in Cryptology – EUROCRYPT*, 2011.
53. a. shelat and C.-H. Shen. Fast two-party secure computation with minimal assumptions. *Conference on Computer and Communications Security*, 2013.

54. E. M. Songhori, S. U. Hussain, A.-R. Sadeghi, T. Schneider, and F. Koushanfar. TinyGarble: Highly compressed and scalable sequential garbled circuits. *IEEE Symposium on Security and Privacy*, 2015.
55. D. P. Woodruff. Revisiting the efficiency of malicious two-party computation. *Advances in Cryptology – EUROCRYPT*, 2007.
56. A. C.-C. Yao. How to generate and exchange secrets (extended abstract). *Annual Symposium on Foundations of Computer Science*, 1986.
57. S. Zahur, M. Rosulek, and D. Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. *Advances in Cryptology – EUROCRYPT*, 2015.

A Proofs

Proof of Lemma 1:

1. Since $\boldsymbol{m}$ is sampled uniform-random, every one of the ℓ symbols in $\boldsymbol{m}$ can equally-likely take any one of the 2^σ values of $\mathbb{F}_{2^\sigma}$. As $\langle\boldsymbol{m}\rangle$ is the product of $\boldsymbol{m}$ and w linearly independent vectors, P_2 can eliminate w unknown symbols of $\boldsymbol{m}$ using the w equations implied by $\langle\boldsymbol{m}\rangle$, while the remaining $\ell - w$ symbols are still completely unconstrained, i.e., for every $\langle\boldsymbol{m}_0\rangle$,
$$\Pr\left(\boldsymbol{m} = \boldsymbol{m}_0 | \langle\boldsymbol{m}\rangle = \langle\boldsymbol{m}_0\rangle\right) = 2^{-(\ell-w)\sigma}.$$
Therefore,
$$\begin{aligned}\mathbf{H}_{min}(\boldsymbol{m}|\langle\boldsymbol{m}\rangle) &= -\sum_{\boldsymbol{m}_0}\Pr\left(\boldsymbol{m} = \boldsymbol{m}_0|\langle\boldsymbol{m}\rangle = \langle\boldsymbol{m}_0\rangle\right)\cdot\log\Pr\left(\boldsymbol{m} = \boldsymbol{m}_0|\langle\boldsymbol{m}\rangle = \langle\boldsymbol{m}_0\rangle\right)\\ &= -\sum_{\boldsymbol{m}_0}2^{-(\ell-w)\sigma}\cdot[-(\ell-w)\sigma]\\ &= 2^{-(\ell-w)\sigma}\cdot[(\ell-w)\sigma]\cdot 2^{(\ell-w)\sigma} = (\ell-w)\sigma.\end{aligned}$$
2. Let $(h_{i_1}, \dots, h_{i_w})$ be a tuple of w symbols. The linear system of equations, $\boldsymbol{v}_{i_t}\cdot\boldsymbol{m} = h_{i_t}, \forall 1 \le t \le w$, will always has some solutions for the ℓ-symbol vector $\boldsymbol{m}$ because $\boldsymbol{v}_{i_1}, \dots, \boldsymbol{v}_{i_w}$ are linearly independent and $w < \ell$. That is, for any $\boldsymbol{m}$ satisfies the w equations above, $(h_{i_1}, \dots, h_{i_w})$ is a hash of $\boldsymbol{m}$.
3. We know $n > \ell$, and P_1 computes the n-symbol encoding of $\boldsymbol{m}$ as $(\boldsymbol{v}_1\boldsymbol{m}, \dots, \boldsymbol{v}_n\boldsymbol{m})$. If $\boldsymbol{m} \neq \boldsymbol{m'}$, their n-symbol encodings will be identical in at most $\ell - 1$ symbols (otherwise, we can use the ℓ identical symbols to show $\boldsymbol{m} = \boldsymbol{m'}$ because any ℓ vectors of $\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_n\}$ are linearly independent). Since w (out of n) symbols of the encoding are randomly selected and watched, the change can evade P_2's detection with probability no more than $\binom{\ell-1}{w}\Big/\binom{n}{w}$. □

Proof of Lemma 2: Following the analysis of Section 5.3, fixing $N, B (B \le 2), T$ can be set to a value based on N, B such that it does not make sense for P_1 to generate more than b^u bad gates (where b^u is a constant fixed by N, B, s, τ because $\forall b > b^u, \Pr_{overall}(N, B, T, \tau, b) < 2^{-s}$. Moreover, because the probability that all b bad gates are consumed in the checking stage is $\binom{T-2N}{b}\Big/\binom{T}{b}$, which approaches 1 when T approaches infinity, there exists a constant T_0, the probability that all bad circuits are consumed by the checking stage is less than 2^{-s}. Thus, we can setup T, B, N to achieve a even stronger guarantee, i.e., all gates in every bucket are correctly garbled, except with probability 2^{-s}. □

Proof of Lemma 3: It suffices to consider the impacts of incorrectly garbled gates. Evaluating an incorrectly garbled gate yields a output wire label that matches the hash $\langle w_o^{p_o}\rangle$, or $\langle w_o^{p_o}\rangle \oplus \langle \varDelta\rangle = \langle w_o^{p_o} \oplus \varDelta\rangle$, or neither of them.

1. If the output label matches with neither hashes, the evaluation result will be discarded (step 3(c)iiG);
2. If the output label matches with a hash and represents the same plain-text value as the output label obtained from evaluating the correct garbled gate in the same bucket, the corrupted garbled gate does not affect the evaluation.

3. If the output label matches with a hash but represents the opposite plain-text value as the output label obtained from evaluating the correct garbled gate, P_2 learns Δ^* at step 3(c)iii. Then P_2 will be able to learn P_1's input x except with negligible probability, because
   (a) Case I ($\Delta^* = \Delta$): P_2 can find out every bit of x through completely decrypt the correctly garbled gates (every bucket has at least one) in the buckets directly operating on the bits of x.
   (b) Case II ($\Delta^* \neq \Delta$): this implies either Δ^* is corrupted (contradicting with the assumption and reasoning above) or Δ is corrupted (contradicting with the fact that no bad gate is identified in the check stage), neither of which can happen except with negligible probability.

In all three cases, P_2 can correctly output $f(x, y)$ except with negligible probability. □

Proof of Lemma 4: Using the specification of EvlAND, the four equations above can be translated into

$$\begin{aligned}
H(w_l^0) \oplus \mathsf{lsb}\,(w_l^0)T_G \oplus H(w_r^0) \oplus \mathsf{lsb}\,(w_r^0)(T_E \oplus w_l^0) &= w_o^{p_o}\\
H(w_l^1) \oplus \mathsf{lsb}\,(w_l^1)T_G \oplus H(w_r^0) \oplus \mathsf{lsb}\,(w_r^0)(T_E \oplus w_l^1) &= w_o^{p_o}\\
H(w_l^0) \oplus \mathsf{lsb}\,(w_l^0)T_G \oplus H(w_r^1) \oplus \mathsf{lsb}\,(w_r^1)(T_E \oplus w_l^0) &= w_o^{p_o}\\
H(w_l^1) \oplus \mathsf{lsb}\,(w_l^1)T_G \oplus H(w_r^1) \oplus \mathsf{lsb}\,(w_r^1)(T_E \oplus w_l^1) &= w_o^{p_o} \oplus \Delta.
\end{aligned}$$

That is,

$$\begin{aligned}
\mathsf{lsb}\,(w_l^0)T_G \oplus w_o^{p_o} \oplus \mathsf{lsb}\,(w_r^0)T_E &= H(w_l^0) \oplus H(w_r^0) \oplus \mathsf{lsb}\,(w_r^0)w_l^0\\
\mathsf{lsb}\,(w_l^1)T_G \oplus w_o^{p_o} \oplus \mathsf{lsb}\,(w_r^0)T_E &= H(w_l^1) \oplus H(w_r^0) \oplus \mathsf{lsb}\,(w_r^0)w_l^1\\
\mathsf{lsb}\,(w_l^0)T_G \oplus w_o^{p_o} \oplus \mathsf{lsb}\,(w_r^1)T_E &= H(w_l^0) \oplus H(w_r^1) \oplus \mathsf{lsb}\,(w_r^1)w_l^0\\
\mathsf{lsb}\,(w_l^1)T_G \oplus w_o^{p_o} \oplus \mathsf{lsb}\,(w_r^1)T_E &= H(w_l^1) \oplus H(w_r^1) \oplus \mathsf{lsb}\,(w_r^1)w_l^1 \oplus \Delta.
\end{aligned}$$

which can be viewed as a linear system of four equations over three variables T_G, T_E, and $w_o^{p_o}$. Note that all coefficients on the left-hand side and all constants on the right-hand side of the equations are fixed by the seven statistically-binding hashes known to P_2. Also note that any three out of the four equations are linearly independent except with negligible probability, because $\mathsf{lsb}\,(w_l^0) \oplus \mathsf{lsb}\,(w_l^1) = 1$, $\mathsf{lsb}\,(w_r^0) \oplus \mathsf{lsb}\,(w_r^1) = 1$, $w_l^0 \oplus w_l^1 = w_r^0 \oplus w_r^1 = \Delta$ and H is modeled as a random oracle. Thus, if any three of the four equations hold, the fourth one will be automatically satisfied as it is simply a linear combination of the other three. In addition, we know there must be one solution to the system of four equations if P_1 follows the specification of GenAND. Therefore, if $T_G, T_E, w_o^{p_o}$ take some corrupted values such that any one equation does not hold, there has to be at least one other equality that does not hold, (otherwise, $T_G, T_E, w_o^{p_o}$ have to satisfy all four equations). Namely, if the gate is corrupted, at least two out of the four equations evidence the corruption. Hence, P_2 detects the corruption with probability at least $1/2$ as it randomly checks one equations at step 4c. □

Proof of Lemma 5: Fixing $\sigma = \sigma_0$, (2) stipulates that for any ℓ, $w \leq \lfloor \ell - \mu/\sigma_0 \rfloor$. Since for all ℓ, n where $n > \ell > w$, $\binom{\ell-1}{w} / \binom{n}{w}$ decreases as w increases (a proof is easy to obtain similarly to that in part 1 of Lemma 14). If setting $w := \lfloor \ell - \mu/\sigma_0 \rfloor$ still can't satisfy (1), there is no other w that can work with σ_0 to meet (1).

For any given $S = S_0, \sigma = \sigma_0, w = w_0$, because $\ell = (S_0/\sigma_0 - \alpha n)/\beta$, the value of $\binom{\ell-1}{w} / \binom{n}{w}$ is solely decided by n, hence we denote $Q(n) = \binom{\ell-1}{w} / \binom{n}{w} = \binom{(S_0/\sigma_0 - \alpha n)/\beta - 1}{w_0} \Big/ \binom{n}{w_0}$.

Without loss of generality, assume $d = \mathsf{gcd}(\alpha, \beta) = 1$ (otherwise it suffices to minimize $S = (\alpha' n + \beta' \ell)\sigma d$ where $\alpha' = \alpha/d, \beta' = \beta/d$ and $\mathsf{gcd}(\alpha', \beta') = 1$), because $S, n, l, \sigma, \alpha, \beta$ are all integers, n can only take values at length β intervals. Let $S' = S/\sigma$, we have

$$\frac{Q(n+\beta)}{Q(n)} = \frac{\left(n - \frac{S'-\alpha n}{\beta} + 2 + (\ell - w - 1)\right) \cdots \left(n - \frac{S'-\alpha n}{\beta} + \beta + \alpha + 1 + (\ell - w - 1)\right)}{\left(\frac{S'-\alpha n}{\beta}\right) \cdots \left(\frac{S'-\alpha n}{\beta} + \alpha - 1\right) \cdot (n+1) \cdots (n+\beta)}$$

and also that

$$d\log\frac{Q(n+\beta)}{Q(n)}\Big/dn = \frac{1+\alpha/\beta}{n-\frac{S'-\alpha n}{\beta}+2+(\ell-w-1)}+\cdots$$
$$+\frac{1+\alpha/\beta}{n-\frac{S'-\alpha n}{\beta}+\beta+1+\alpha+(\ell-w-1)}+\frac{\alpha/\beta}{\frac{S'-\alpha n}{\beta}}+\cdots$$
$$+\frac{\alpha/\beta}{\frac{S'-\alpha n}{\beta}+\alpha-1}-\frac{1}{n+1}-\cdots-\frac{1}{n+\beta}>0,$$

since $(S'-\alpha n)/\beta=\ell>\ell-w$ so for all non-negative integer x, $n-(S'-\alpha n)/\beta+2+x+(\ell-w-1)<n+1+x$. This implies $Q(n)$ is a convex function of n. Therefore, we can apply a discrete Newton algorithm to find its minimum in logarithmic number of steps. If the minimal $Q(n)$ satisfies (1), we found a viable solution that achieves S_0; otherwise, there won't be any n to work with the parameter setting σ_0 to achieve S_0. □

Proof of Lemma 6: There exists N_0 such that if $N>N_0$,

$$\begin{aligned}
\frac{\binom{T-b}{T-BN-i}}{\binom{T}{T-BN}} &= \frac{(T-b)!(T-BN)!(BN)!}{T!(T-BN-i)!(BN-b+i)!}\\
&= \frac{((T-BN-i+1)\cdots(T-BN))((BN-b+i+1)\cdots BN)}{(T-b+1)\cdots T}\\
&= \frac{(T-BN-i+1)\cdots(T-BN)}{(T-i+1)\cdots T}\cdot\frac{(BN-b+i+1)\cdots BN}{(T-b+1)\cdots(T-i)}\\
&\le \left(\frac{T-BN}{T}\right)^i\left(\frac{BN}{T-i}\right)^{b-i}\triangleq U.
\end{aligned}$$

Similarly, we have, there exists N_1 such that if $N>N_1$,

$$\frac{\binom{T-b}{T-BN-i}}{\binom{T}{T-BN}}\ge\left(\frac{T-BN-i+1}{T-i+1}\right)^i\left(\frac{BN-b+i+1}{T-b+1}\right)^{b-i}\triangleq L.$$

So, we know that, for sufficiently large N,

$$\begin{aligned}
U\Big/\frac{\binom{T-b}{T-BN-i}}{\binom{T}{T-BN}} \le \frac{U}{L} &= \left(\frac{T-BN}{T-BN-i+1}\cdot\frac{T-i+1}{T}\right)^i\left(\frac{BN}{BN-b+i+1}\cdot\frac{T-b+1}{T-i}\right)^{b-i}\\
&= \left[\frac{(T-BN)T-(i-1)T+(i-1)BN}{(T-BN)T-(i-1)T}\right]^i\cdot\\
&\qquad\left[\frac{(BN-b+i+1)(T-i)+(b-i-1)(T-BN-i)}{(BN-b+i+1)(T-i)}\right]^{b-i}\\
&= \left[1+\frac{(i-1)BN}{(T-BN)T-(i-1)T)}\right]^i\left[1+\frac{(b-i-1)(T-BN-i)}{(BN-b+i+1)(T-i)}\right]^{b-i}\\
&\le \left(1+\frac{i-1}{T-BN-i+1}\right)^i\left(1+\frac{b-i-1}{BN-b+i+1}\right)^{b-i} \qquad (6)\\
&\le \left(1+\frac{i-1}{T-BN-i+1}\right)^i\left(1+\frac{b-1}{BN-b+1}\right)^{b} \qquad (7)
\end{aligned}$$

Note that the inequality (6) holds because $T>BN$. Thanks to the upper-bound of b (Lemma 7) and hence on i (recall $i\le b$), $\lim_{N\to\infty}\left(1+\frac{i-1}{T-BN-i+1}\right)^i\left(1+\frac{b-1}{BN-b+1}\right)^b=1$. □

Proof of Lemma 7: We know that there exists i_0, N_0 such that $(1-r)^{i_0} < 2^{-s-1}$ for all $r > r_0$. Following Lemma 13, we also know that

$$\begin{aligned}\Pr_{overall}(N,B,T,b) &\leq \left((1-\tau)\frac{T-2N}{T} + \frac{2N}{T-i_0}\right)^b \Pr_e(N,b) + 2^{-s-1}\\ &\leq \left((1-\tau)\frac{T-2N}{T} + \frac{2N}{T-i_0}\right)^b + 2^{-s-1}\end{aligned}$$

Because for all $N > N_0$, $\lim_{b\to\infty} \Pr_{overall}(N,B,T,b) = 2^{-s+1}$, there exists b^u such that

$$\Pr_{overall}(N,B,T,b) < 2^{-s}, \qquad \text{for all } b > b^u.$$

□

Proof of Lemma 8: We assume for simplicity that in the gate verification stage, if a bad gate is indeed selected for verification, P_2 is able to detect the fact that "the gate is bad" with probability 1 (i.e., $\tau = 1$). (If $\tau < 1$, it only makes it easier for a cheating P_1 to succeed.)

Let $\Pr_c(N,x,b)$ be the probability that P_1 survives the gate verification stage; and $\Pr_e(N,x,b)$ be the probability that, P_1 succeeds in the evaluation stage given that it already passes the verification stage. Because $\tau = 1$, in a successful attack, no bad gates will be "consumed" in the verification stage. Therefore, there exists a positive constant c and a constant N_0 such that for all $N > N_0$

$$\begin{aligned}\Pr_{overall}(N,x,b) &= \Pr_c(N,x,b)\cdot\Pr_e(N,x,b)\\ &\geq \left(\frac{1+x}{2}\right)^b\left[1-\frac{(1-x)b^2}{(1+x)(2N-b)}\right]\cdot\Pr_e(N,x,b) && \text{[Lemma 10]}\\ &\geq \left(\frac{1+x}{2}\right)^b\left[1-\frac{(1-x)b^2}{(1+x)(2N-b)}\right]\left(1-\left(\frac{2x}{1+x}\right)^b\right) && \text{[Lemma 11]}\\ &> c && \text{[Lemma 12]}\end{aligned}$$

This completes the proof. □

Proof of Lemma 9: First, we show that, if $B > 2$, a scheme S_1 that uses at least a size-1 bucket and a size-B bucket can always be improved (in terms of thwarting attacks from a cheating generator) into a scheme S_2 by replacing the two bucket by a size-2 bucket and a size-$(B-1)$ bucket. Since the only difference between S_1 and S_2 is this pair of buckets, the change actually preserves the total number of garbled gates being used. We say S_2 is at least as good as S_1 because for any positive integer x_b (which denotes the number of bad gates falling into the $B+1$ slots in these two buckets), a malicious generator is less or equally likely to succeed with S_2 than S_1. This fact is evident from counting the number of ways to place the x_b bad gates that will not result a successful attack: assume $B > 2$,

1. If $x_b < B-1$, the count is $\binom{B}{x_b}$ for S_1 (as the only way to fail the attack is to put the x_b bad gates in the size-B bucket) and $\binom{B-1}{x_b} + \binom{2}{1}\binom{B-1}{x_b-1}$ for S_2 (as the x_b bad gates are either all put into the size-$(B-1)$ bucket, or 1 put in the size-2 bucket while the rest $x_b - 1$ put in the size-$(B-1)$ bucket). Hence,

$$\binom{B-1}{x_b} + \binom{2}{1}\binom{B-1}{x_b-1} = \binom{B}{x_b} + \binom{B-1}{x_b-1} > \binom{B}{x_b}, \quad \forall\, 0 < x_b < B-1$$

   implies S_2 is better than S_1.
2. If $x_b = B-1$, the count is B for S_1 and $2(B-1)$ for S_2. Since $2(B-1) > B$, S_2 is better than S_1.
3. If $x_b > B-1$, the count is 0 for both S_1 and S_2: both are destined to successful attacks.

Through recursively applying the above observation to *any* LEGO-style cut-and-choose scheme, we will derive an *improved* scheme that either

1. **uses only buckets of size 2 or above:** because at least 1 gate needs to be used for verification, the overall κ has to be greater than 2. The scope of this claim ($\kappa = 2$) already eliminates this case. Or,
2. **uses only buckets of size 1 and 2.**

This completes the proof. $\square$

Proof of Lemma 10: Since a total of $(1+x)N$ garbled gates are used in evaluation while the rest $T-(1+x)N$ gates are used for checking, we have

$$\begin{aligned}
\Pr_c(N,x,b) &= \binom{T-b}{T-(1+x)N} \Big/ \binom{T}{T-(1+x)N} \\
&= \binom{2N-b}{2N-(1+x)N} \Big/ \binom{2N}{2N-(1+x)N} \qquad (8)\\
&= \binom{2N-b}{(1-x)N} \Big/ \binom{2N}{(1-x)N} \\
&= \frac{(2N-b)(2N-b-1)\cdots[2N-b-(1-x)N+1]}{2N(2N-1)\cdots[2N-(1-x)N+1]} \\
&= \frac{[2N-(1-x)N]\cdots[2N-b-(1-x)N+1]}{2N(2N-1)\cdots[2N-b+1]} \\
&= \frac{[2N-(1-x)N]}{2N}\cdot\dots\cdot\frac{[2N-b-(1-x)N+1]}{[2N-b+1]} \\
&= \frac{[(1+x)N]}{2N}\cdot\dots\cdot\frac{[(1+x)N-b+1]}{[2N-b+1]} \\
&\geq \left[\frac{(1+x)N-b}{2N-b}\right]^b \qquad (9)\\
&= \left(\frac{1+x}{2}\right)^b\left(1-\frac{(1-x)b}{(1+x)(2N-b)}\right)^b \\
&\geq \left(\frac{1+x}{2}\right)^b\left(1-\frac{(1-x)b^2}{(1+x)(2N-b)}\right) \qquad (10)
\end{aligned}$$

where equality (8) holds because $\kappa = T/N = 2$; the inequality (9) holds because every of the b fractions is larger than or equal to $[(1+x)N-b]/(2N-b)$; and (10) can be derived from the binomial inequality (i.e., $\forall x \in \mathbb{R}, x > -1, \text{and} \forall n \in \mathbb{N}, (1+x)^n \geq 1+nx$) and the fact that $\frac{(1-x)b}{(1+x)(2N-b)} < 1$ when $0 \leq b \leq \kappa N = 2N$, $\frac{1}{N} \leq x \leq 1-\frac{1}{N}$. $\square$

Proof of Lemma 11: We assume that P_1 fails an attack if there is at least one good gate in every bucket (the proposed protocol in Section 4 indeed achieves this), which happens with probability $2^b\binom{xN}{b}/\binom{(1+x)N}{b}$, where $2^b\binom{xN}{b}$ is the number of ways to place b bad gates into the xN size-2 buckets subject to at most one bad gate per bucket, and $\binom{(1+x)N}{b}$ is the total number of ways to place the b bad gates without any restriction. Therefore,

$$\begin{aligned}
\Pr_e(N,x,b) &= 1-\frac{2^b\binom{xN}{b}}{\binom{(1+x)N}{b}} = 1-\frac{2^b[xN-(b-1)][xN-(b-2)]\cdots[xN]}{[(1+x)N-(b-1)][(1+x)N-(b-2)]\cdots[(1+x)N]} \\
&= 1-\frac{2xN-2(b-1)}{(1+x)N-(b-1)}\cdot\frac{2xN-2(b-2)}{(1+x)N-(b-2)}\cdot\dots\cdot\frac{2xN}{(1+x)N} \\
&\geq 1-\left(\frac{2x}{1+x}\right)^b
\end{aligned}$$

where the inequality holds because every of the b fractions is greater than or equal to $(2x)/(1+x)$. $\square$

Proof of Lemma 12: It suffices to show that there exists $c_1, c_2, c_3 > 0$ and $N_0 > 0$ such that $\forall N > N_0, \forall x \in [1/N, 1-1/N]$, there exists a positive integer b that satisfy all of the three inequality below,

$$\left(\frac{1+x}{2}\right)^b > c_1 \tag{11}$$

$$1 - \frac{(1-x)b^2}{(1+x)(2N-b)} > c_2 \tag{12}$$

$$1 - \left(\frac{2x}{1+x}\right)^b > c_3 \tag{13}$$

Because (11) holds as long as $b < \log c_1 \big/ \log \frac{1+x}{2}$, (12) holds as long as

$$b < \sqrt{2N\frac{1+x}{1-x}(1-c_2) + \frac{1}{4}\left(\frac{1+x}{1-x}\right)^2(1-c_2)^2} - \frac{1}{2}\cdot\frac{1+x}{1-x}\cdot(1-c_2),$$

(13) holds as long as $b > \log(1-c_3)\big/\log\frac{2x}{1+x}$, and b needs to be a positive integer, it suffices to show that there exist positive c_1, c_2, c_3, and N_0 such that the following two inequalities hold for all $N > N_0$

$$\frac{\log(1-c_3)}{\log\frac{2x}{1+x}} + 1 < \frac{\log c_1}{\log\frac{1+x}{2}} \tag{14}$$

$$\frac{\log(1-c_3)}{\log\frac{2x}{1+x}} + 1 < \sqrt{2N\frac{1+x}{1-x}(1-c_2) + \frac{1}{4}\cdot\left(\frac{1+x}{1-x}\right)^2(1-c_2)^2} - \frac{1}{2}\cdot\frac{1+x}{1-x}\cdot(1-c_2) \tag{15}$$

We note that (14) is equivalent to

$$\frac{1}{\log c_1}\log\frac{1+x}{2x} > \frac{1}{\log(1-c_3)}\log\frac{2}{1+x} + \log\frac{1+x}{2x}\log\frac{2}{1+x},$$

which will always hold as long as

$$\frac{1}{\log c_1} - \frac{1}{\log(1-c_3)} - \log 2 > 0 \tag{16}$$

because $\frac{1+x}{2x} > \frac{2}{1+x}$ and $\log\frac{2}{1+x} < \log 2$ hold for all $x \in [1/N, 1-1/N]$. Since (16) doesn't involve x, it is easy to find a c_1 based on (any value of) c_3 such that (16) is satisfied.

Next, we note that (15) is equivalent to

$$\frac{2N\frac{1+x}{1-x}(1-c_2)}{\sqrt{2N\frac{1+x}{1-x}(1-c_2) + \frac{1}{4}(\frac{1+x}{1-x})^2(1-c_2)^2} + \frac{1}{2}\cdot\frac{1+x}{1-x}\cdot(1-c_2)} > 1 + \frac{\log\frac{1}{1-c_3}}{\log\frac{1+x}{2x}},$$

which can be simplified, by defining $c_2' = 1 - c_2$ and $y = (1+x)/(1-x)$ (hence, $y \in [1+1/N, 2N]$ and $(1+x)/(2x) = 1 + 1/y$), to

$$\left(\frac{2N}{\sqrt{\frac{2N}{yc_2'} + \frac{1}{4}} + \frac{1}{2}} - 1\right)\log\left(1 + \frac{1}{y-1}\right) > \log\frac{1}{1-c_3}.$$

Now we analyze this inequality in two cases.

Case I ($y < 3$): If $y < 3$,

$$\left(\frac{2N}{\sqrt{\frac{2N}{yc_2'}+\frac{1}{4}}+\frac{1}{2}}-1\right)\log\left(1+\frac{1}{y-1}\right) \geq (\frac{2N}{\sqrt{\frac{2N}{c_2'}+\frac{1}{4}}+\frac{1}{2}}-1)\log(1+\frac{1}{y-1})$$
$$\geq (\frac{2N}{\sqrt{\frac{2N}{c_2'}+1}}-1)\log(1+\frac{1}{y-1})$$
$$\geq (\frac{2N}{\sqrt{\frac{2N}{c_2'}+1}}-1)\log\frac{3}{2}.$$

Because there exists an integer N_0 such that for all $N > N_0$,

$$\left(\frac{2N}{\sqrt{\frac{2N}{c_2'}+1}}-1\right)\log\frac{3}{2} > \log\frac{1}{1-c_3} \tag{17}$$

(15) can also be satisfied when $N > N_0$, regardless of the values of c_2' and c_3.

Case II ($y \geq 3$): If $y \geq 3$, then $0 < 1/(y-1) < 1/2$, and (because $y = \log(1+x)$ is concave function when $x \in [0, 1/2]$) we have

$$\log[1+1/(y-1)] \geq [2\log(3/2)]/(y-1) > \frac{2\log(3/2)}{y}.$$

In addition, for all $N > N_0$ (where N_0 is defined as above), we have

$$\frac{2N}{\sqrt{\frac{2N}{yc_2'}+\frac{1}{4}}+\frac{1}{2}}-1 > \frac{2N}{\sqrt{\frac{2N}{c_2'}+1}}-1 > 0.$$

Thus, for all $N > N_0$, we know

$$\left(\frac{2N}{\sqrt{\frac{2N}{yc_2'}+\frac{1}{4}}+\frac{1}{2}}-1\right)\log\left(1+\frac{1}{y-1}\right) \geq \left(\frac{2N}{\sqrt{\frac{2N}{yc_2'}+\frac{1}{4}}+\frac{1}{2}}-1\right)\frac{2\log(3/2)}{y}$$
$$= 2\log(3/2)\left(\frac{2N}{\sqrt{\frac{2Ny}{c_2'}+\frac{1}{4}y^2}+\frac{1}{2}y}-\frac{1}{y}\right)$$
$$\geq 2\log(3/2)\left(\frac{2N}{\sqrt{\frac{4N^2}{c_2'}+N^2}+N}-\frac{1}{3}\right)$$
$$= 2\log(3/2)\left(\frac{2}{1+\sqrt{\frac{4}{c_2'}+1}}-\frac{1}{3}\right),$$

where the second inequality holds because $3 \leq y \leq 2N$. Therefore, it is easy to find c_3 based on (any value of) c_2' to satisfy

$$2\log(3/2)\left(\frac{2}{1+\sqrt{\frac{4}{c_2'}+1}}-\frac{1}{3}\right) > \log\frac{1}{1-c_3}, \tag{18}$$

which will guarantee (15) holds.

To sum up, we have shown that if we pick an arbitrary positive number c_2, then find c_3 based on (18) and c_2, find c_1 based on (16) and c_3, and finally find N_0 based on (17) and c_2, c_3, then for all $N > N_0$, all three inequalities, (11), (12), (13) should hold. This completes the proof. $\square$

Proof of Theorem 4: The proof, which is based on the construction of our protocol in Section 4.2, can be derived from the proof in Section 4.4, by substituting the use of Lemma 2 and Lemma 3 with the following Lemma 13, in the case of P_2 being corrupted (while the proof for corrupted P_1 works unaffected). $\square$

Proof of Lemma 13: Let $0 < \tau \leq 1$ be the probability that P_2 detects the abnormality in checking garbled gate g conditioned on g is indeed bad. We have

$$\Pr_{overall}(N,b) = \sum_{i=0}^{b}(1-\tau)^i \frac{\binom{b}{i}\binom{T-b}{T-2N-i}}{\binom{T}{T-2N}} \Pr_e(N, b-i)$$

where $(1-\tau)^i\binom{b}{i}\binom{T-b}{T-2N-i} \Big/ \binom{T}{T-2N}$ is the probability that P_1 who generates b bad gates survives the gate verification stage with i bad gates selected for verification (but P_2 fails to detect any of them). Because there exists i_0 such that $(1-\tau)^{i_0} < \varepsilon/2$,

$$\begin{aligned}
\Pr_{overall}(N,b) &= \sum_{i=0}^{b}(1-\tau)^i \frac{\binom{b}{i}\binom{T-b}{T-2N-i}}{\binom{T}{T-2N}} \Pr_e(N, b-i) \\
&= \sum_{i=0}^{i_0}(1-\tau)^i \frac{\binom{b}{i}\binom{T-b}{T-2N-i}}{\binom{T}{T-2N}} \Pr_e(N, b-i) + \sum_{i=i_0+1}^{b}(1-\tau)^i \frac{\binom{b}{i}\binom{T-b}{T-2N-i}}{\binom{T}{T-2N}} \Pr_e(N, b-i) \\
&\leq \sum_{i=0}^{i_0}(1-\tau)^i \frac{\binom{b}{i}\binom{T-b}{T-2N-i}}{\binom{T}{T-2N}} \Pr_e(N, b-i) + (1-\tau)^{i_0}\sum_{i=i_0+1}^{b} \frac{\binom{b}{i}\binom{T-b}{T-2N-i}}{\binom{T}{T-2N}} \Pr_e(N, b-i) \\
&\leq \sum_{i=0}^{i_0}(1-\tau)^i \frac{\binom{b}{i}\binom{T-b}{T-2N-i}}{\binom{T}{T-2N}} \Pr_e(N, b-i) + \frac{\varepsilon}{2}\sum_{i=i_0+1}^{b} \frac{\binom{b}{i}\binom{T-b}{T-2N-i}}{\binom{T}{T-2N}} \\
&\leq \sum_{i=0}^{i_0}(1-\tau)^i \frac{\binom{b}{i}\binom{T-b}{T-2N-i}}{\binom{T}{T-2N}} \Pr_e(N, b-i) + \frac{\varepsilon}{2}\sum_{i=1}^{b} \frac{\binom{b}{i}\binom{T-b}{T-2N-i}}{\binom{T}{T-2N}} \\
&\leq \sum_{i=0}^{i_0}(1-\tau)^i \frac{\binom{b}{i}\binom{T-b}{T-2N-i}}{\binom{T}{T-2N}} \Pr_e(N, b-i) + \frac{\varepsilon}{2}\cdot 1 \\
&\leq \sum_{i=0}^{i_0}(1-\tau)^i \binom{b}{i}\left(\frac{T-2N}{T}\right)^i \left(\frac{2N}{T-i}\right)^{b-i} \Pr_e(N, b-i) + \frac{\varepsilon}{2} \qquad \text{[Lemma 15]} \\
&\leq \sum_{i=0}^{i_0}(1-\tau)^i \binom{b}{i}\left(\frac{T-2N}{T}\right)^i \left(\frac{2N}{T-i}\right)^{b-i} \Pr_e(N, b) + \frac{\varepsilon}{2} \\
&\leq \sum_{i=0}^{i_0}(1-\tau)^i \binom{b}{i}\left(\frac{T-2N}{T}\right)^i \left(\frac{2N}{T-i_0}\right)^{b-i} \Pr_e(N, b) + \frac{\varepsilon}{2} \\
&\leq \sum_{i=0}^{b}(1-\tau)^i \binom{b}{i}\left(\frac{T-2N}{T}\right)^i \left(\frac{2N}{T-i_0}\right)^{b-i} \Pr_e(N, b) + \frac{\varepsilon}{2} \\
&= \left((1-\tau)\frac{T-2N}{T} + \frac{2N}{T-i_0}\right)^b \Pr_e(N,b) + \frac{\varepsilon}{2}.
\end{aligned}$$

Since $T = \kappa N$, we have

$$\lim_{N\to\infty} \frac{(1-\tau)(T-2N)}{T} + \frac{2N}{T-i_0} = \lim_{N\to\infty} \frac{(1-\tau)(\kappa-2)}{\kappa} + \frac{2}{\kappa - i_0/N} = 1 - \frac{\tau(\kappa-2)}{\kappa} < 1.$$

Therefore, there exists N_1 such that for all $N > N_1$, $(1-\tau)(T-2N)/T + 2N/(T-i_0) < 1$. Hence, for every $\varepsilon > 0$, we can find a b_0 such that,

1. for all $b > b_0$,
$$\begin{aligned}\Pr_{overall}(N,b) &\le \big[(1-\tau)(T-2N)/T + 2N/(T-i_0)\big]^b \Pr_e(N,b) + \varepsilon/2 \\ &< \varepsilon \Pr_e(N,b)/2 + \varepsilon/2 < \varepsilon/2 + \varepsilon/2 = \varepsilon.\end{aligned}$$
2. for all $b \le b_0$, by Lemma 14, we can further find an integer N_2 (according to the proof of Lemma 14) such that for all $N > N_2$,
$$\begin{aligned}\Pr_{overall}(N,b) &\le \big[(1-\tau)(T-2N)/T + 2N/(T-i_0)\big]^b \Pr_e(N,b) + \varepsilon/2 \\ &< \big[1-\tau(\kappa-2)/\kappa\big]^b \varepsilon/2 + \varepsilon/2 < \varepsilon/2 + \varepsilon/2 = \varepsilon.\end{aligned}$$

Thus, setting $N_0 = \max(N_1, N_2)$ completes the proof. □

Proof of Lemma 14: Since all buckets are of size 2, the following can be derived similarly to the proof of Lemma 11 (by setting $x = 1$),

$$\begin{aligned}\Pr_e(N,b) &= 1 - \frac{2^b\binom{N}{b}}{\binom{2N}{b}} \\ &= 1 - \frac{2N-2(b-1)}{2N-(b-1)} \cdot \frac{2N-2(b-2)}{2N-(b-2)} \cdot \cdots \cdot \frac{2N}{2N} \\ &\le 1 - \left(\frac{2N-2b+2}{2N-b+1}\right)^b = 1 - \left(\frac{2-2(b-1)/N}{2-(b-1)/N}\right)^b\end{aligned}$$

1. From the second equality above, we know $\Pr_e(N,b)$ is a strictly increasing function of b because the larger b is, the more multiplicative fractions (that are smaller than 1) are in the product form.
2. The above inequality indicates that for all b and ε, there exists an N_1 such that for all $N > N_1$, $\Pr_e(N,b) < \varepsilon$. □

Proof of Lemma 15:

$$\begin{aligned}\frac{\binom{T-b}{T-2N-i}}{\binom{T}{T-2N}} &= \frac{(T-b)!(T-2N)!(2N)!}{T!(T-2N-i)!(2N-b+i)!} \\ &= \frac{\big[(T-2N-i+1)\cdots(T-2N)\big]\big[(2N-b+i+1)\cdots 2N\big]}{(T-b+1)(T-b+2)\cdots T} \\ &= \frac{(T-2N-i+1)\cdots(T-2N)}{(T-i+1)\cdots T} \cdot \frac{(2N-b+i+1)\cdots 2N}{(T-b+1)\cdots(T-i)} \le \left(\frac{T-2N}{T}\right)^i \left(\frac{2N}{T-i}\right)^{b-i}\end{aligned}$$

□

B Half-Gates Garbling and Evaluating Algorithms

Algorithm 1 GenAND(i, Δ)

```
$w_l \leftarrow \{0,1\}^{\lambda_w}$; $w_r^0 \leftarrow \{0,1\}^{\lambda_w}$
$q_l :=$ lsb$(w_l^0)$; $q_r :=$ lsb$(w_r^0)$
$j := 2i$;    $j' := 2i+1$
$T_G := H(w_l^0, j) \oplus H(w_l^1, j) \oplus q_r\Delta$
$w_g^0 := H(w_l^0, j) \oplus q_l T_G$
$T_E := H(w_r^0, j') \oplus H(w_r^1, j') \oplus w_l^0$
$w_e^0 := H(w_r^0, j') \oplus q_r(T_E \oplus w_l^0)$
$w_o^0 := w_g^0 \oplus w_e^0$
return $(w_l^0, w_r^0, w_o^0, T_G, T_E)$
```

Algorithm 2 EvlAND(w_l, w_r, T_G, T_E)

```
$s_l :=$ lsb$(w_l)$; $s_r :=$ lsb$(w_r)$;
$j := 2i$;    $j' := 2i+1$
$w_g := H(w_l, j) \oplus s_l T_G$
$w_e := H(w_r, j') \oplus s_r(T_E \oplus w_l)$
$w_o := w_g \oplus w_e$
return $w_o$
```