MiniLEGO:
Efficient Secure Two-Party Computation From General Assumptions
(Full Version)

Tore Kasper Frederiksen[1], Thomas Pelle Jakobsen[1], Jesper Buus Nielsen[1], Peter Sebastian Nordholt[1], and Claudio Orlandi[1] ⋆

Department of Computer Science, Aarhus University
{jot2re|tpj|jbn|psn|orlandi}@cs.au.dk

Abstract One of the main tools to construct secure two-party computation protocols are Yao garbled circuits. Using the cut-and-choose technique, one can get reasonably efficient Yao-based protocols with security against malicious adversaries. At TCC 2009, Nielsen and Orlandi [NO09] suggested to apply cut-and-choose at the gate level, while previously cut-and-choose was applied on the circuit as a whole. This appealing idea allows for a speed up with practical significance (in the order of the logarithm of the size of the circuit) and has become known as the "LEGO" construction. Unfortunately the construction in [NO09] is based on a specific number-theoretic assumption and requires public-key operations per gate of the circuit.

The main technical contribution of this work is a new XOR-homomorphic commitment scheme based on oblivious transfer, that we use to cope with the problem of connecting the gates in the LEGO construction. Our new protocol has the following advantages:

1. It maintains the efficiency of the LEGO cut-and-choose.
2. After a number of seed oblivious transfers linear in the security parameter, the construction uses only primitives from Minicrypt (i.e., private-key cryptography) per gate in the circuit (hence the name MiniLEGO).
3. On the contrary of original LEGO, MiniLEGO is compatible with all known optimization for Yao garbled gates (row reduction, free-XORs, point-and-permute).

⋆ Partially supported by the European Research Commission Starting Grant 279447 and the Danish National Research Foundation and The National Science Foundation of China (grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation. Tore and Thomas are supported by Danish Council for Independent Research Starting Grant 10-081612.

Contents

1 Introduction

Secure computation allows two (or more) parties to compute a function of their inputs while ensuring security properties such as the privacy of the inputs and the correctness of the outputs. The first and arguably still most popular protocol for secure two-party computation is Yao's garbled circuit [Yao86, LP09].

In recent years there has been a significant effort to bring secure computation into practice. These efforts resulted in terrific improvements in terms of algorithmic complexity, efficiency of implementations etc. (see e.g., [ST04, MNPS04, LP07, BDNP08, IPS08, LPS08, KS08, IKOS08, NO09, PSSW09, HKS+10, LP11, LOP11, sS11, HEKM11, KsS12, DKL+12, NNOB12, HKE12, BHR12, FN13] and references therein). Perhaps the most interesting problem is how to achieve protocols with security against malicious adversaries that are efficient enough to be used in practice.

In a nutshell, Yao's protocol works as follows: A constructs an encrypted version of the circuit to be computed (the "garbled circuit") and sends it to B who evaluates the encrypted circuit on encrypted inputs, thus learning nothing but the output of the computation. One of the main problems of this protocol is that if A is malicious she can encrypt a circuit different than the one B agreed on computing, with dramatic consequences for the correctness of the result and the privacy of B's input. One of the main tools to cope with this is the so called *cut-and-choose* technique: A prepares many copies of the encrypted circuit and B checks some of them for correctness. This induces a probability on the unopened circuits to be correct. Nielsen and Orlandi [NO09] presented a twist on this approach, known as LEGO: their approach consists of performing a cut-and-choose test at the gate level (instead of at circuit level), and allows to save a factor $O(\log(s))$ with s being the circuit size, in the computation and communication complexity w.r.t., "standard" cut-and-choose at the circuit level. That is, A prepares many encrypted NAND gates, B checks some of them and, if no inconsistency is detected, the unopened gates are randomly permuted and partitioned into small "*buckets*". The NAND gates in each bucket are "*soldered*" together such that the bucket has 2 input wires, a left and right, and one output wire. The input wires are soldered to each of the NAND gates in the bucket's input wires, left to left and right to right. The output of the NAND gates in the bucket are soldered together to the output wires of the bucket such that if the output of the NAND gates are different the majority (if such exist) is going to be the value of the bucket's output wire. Thus a bucket represent a redundant version of a garbled NAND gate in the original circuit, but which computes the correct output in the presence of a minority faulty internal gates.

The LEGO approach has been praised for its novelty [Gol09], but did not have a practical impact for the efficiency of Yao-based protocols. There are several reasons for this:

1. *LEGO uses public-key primitives for each gate in the circuit:* Each gate has in fact associated three commitments to its input/output keys. Those commitments are used for the "soldering" and need to be homomorphic. For this purpose LEGO uses Pedersen commitments (based on the hardness of computing discrete logarithms in some group – e.g., the group of points of an elliptic curve). This is a drawback for the efficiency of the protocol (group operations, even in an elliptic curve, are orders of magnitude slower than symmetric primitives such as hash functions or private-key encryption). Moreover, this is also a drawback as LEGO can be only implemented using a single computational assumptions – it is unclear how to implement LEGO under a variety of computational assumptions (including assumptions that are believed to withstand quantum attacks).
2. *LEGO is not compatible with known optimization for Yao's protocol:* Keys in LEGO are element of $\mathbb{Z}_p$ for some prime p, while using binary strings $\{0,1\}^t$ is more natural and standard. Therefore, it is not possible to use the "free-XOR" trick with LEGO, nor many of the others optimizations that are tailored for bit-string keys.
3. *LEGO has too many bricks:* There are many different kind of objects in LEGO (key-filters, not-two gates, etc.) that make the use of LEGO complex to understand and implement.

1.1 Contributions

In this paper, we present a generalization and a simplification of the LEGO approach. The main technical difference is to replace the Pedersen commitments with some XOR-homomorphic commitments based on oblivious transfer (OT) which we believe is of independent interest and might be used in other applications. We take this direction as OT can be efficiently extended (both with passive [IKNP03] and active security [HIKN08, NNOB12]), the price is only a few private-key operations per OT (together with a small number of "real" seed OTs that use public-key technology used to bootstrap the process). Doing so allows us to:

1. Maintain LEGO's good complexity and achieve statistical security 2^{-k} when the replication factor is only $\rho = O(k/\log(s))$ against a replication factor of $\rho = O(k)$ for standard cut-and-choose such as the one in [LP11]
2. Implement a variant on LEGO whose security only relies on generic, symmetric primitives (except for the few seed OTs needed to bootstrap the OT extension).
3. Achieve a variant of LEGO that uses "standard" garbled gates (ANDs and free-XORs and NOTs), compatible with garbled gates optimizations.

To sum up, we propose the first real alternative to standard cut-and-choose for practical purposes Yao-based secure two-party computation. We believe that it is important to have a variety of different protocols from which implementers can choose from. Whether our proposed protocol will be more efficient in practice than protocols with standard cut-and-choose [LP07, PSSW09, sS11, KsS12, FN13] will only be decided by performing a serious comparison of similar implementations running on the same hardware-network configuration of our and other approaches. This is an interesting direction for future work.

Table 1. Comparison of the asymptotic complexity of our protocol with the competition. We let s denote the amount of non-XOR gates in a circuit, k denotes the statistical security parameter and t the computational security parameter. Finally ℓ is the amount of input bits from the players.

	Model	Symmetric	Asymmetric	Rounds	Communication (bits)
Our result	ROM, OT-hybrid	$O(s \cdot k/\log(s))$	$O(k)$	$O(1)$	$O(tks/\log(s))$
[NNOB12]	ROM, OT-hybrid	$O(s \cdot k/\log(s))$	$O(k)$	$O(d)$	$O(tks/\log(s) + t^2)$
[FN13]	ROM, OT-hybrid	$O(s \cdot k)$	$O(k)$	$O(1)$	$O(tks)$
[LP11], [sS11]	SM, DDH	$O(s \cdot k)$	$O(k\ell)$	$O(1)$	$O(tks)$
[NO09]	UC, OT-hybrid, DL, CoRH	$O(s \cdot k/\log(s))$	$O(s \cdot k/log(s))$	$O(1)$	$O(tks/\log(s))$

1.2 Technical Overview

The main idea of the protocol we present here is the same as in [NO09]: A prepares many garbled gates (NANDs in [NO09], while here we use ANDs) together with commitments to the input and output garbled keys. If A prepares a gate dishonestly we view it as a faulty gate, i.e., one that does not give the correct output on some inputs. B asks A to open a random subset of the AND gates and checks them for correctness. If the check goes through, B randomly permutes the unopened gates into buckets representing a redundant AND gate. He solders the gates within a given bucket together and then solders the buckets together to form a circuit that computes the function even in the presence of a minority of faulty gates within each bucket. As part of this soldering NOT gates can be injected thus the garbled AND gates and the soldering alone provides a universal set of Boolean gates.

As the gates have been generated independently, the output keys of the gates in one layer of the circuit cannot be directly fed as input to the next layer. Therefore, A reveals the difference between the output keys in the first layer with the corresponding input keys in the second layer (using the XOR-homomorphic properties of the commitment scheme). This allows B to "align" the input keys of the gates in one layer with the output keys of the gates in the previous layer. The main intuition for the security of LEGO cut-and-choose is as follows: If A had sent B k faulty gates, B would detect this with probability $1 - 2^{-k}$. Therefore, if B accepts the test, with very high probability there are only a few faulty gates among the unopened ones. As all gates are permuted at random and placed in random buckets in the circuit, only very little redundancy is needed to correct for all faulty gates.

The original LEGO construction in [NO09] used a set of specially tailored garbled gates and a complex design for the circuit used to deal with faulty gates. This meant that it was not possible to apply standard garbling techniques and optimizations. In contrast, the construction here can be instantiated with essentially any free-XOR compatible garbled gate scheme and is compatible with various state of the art optimizations (such as free-XOR, row-reduction, point-and-permute). Additionally, we can design the fault tolerant circuit in a very simple way: we implement each gate of the original circuit as a bucket consisting of ρ garbled gates and take the output of the bucket to be any output agreed upon by more than $\lfloor \rho/2 \rfloor$ of the replicated garbled gates it contains.

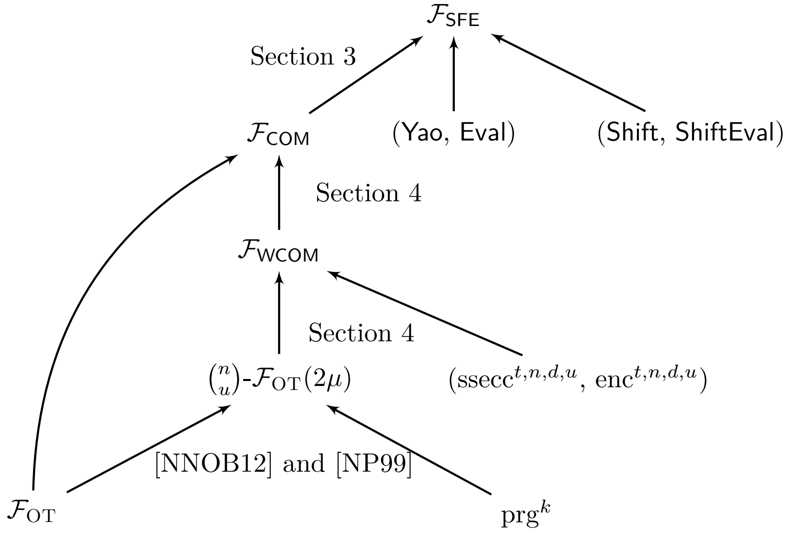
$\mathcal{F}_{\mathsf{SFE}}$
Section 3
$\mathcal{F}_{\mathsf{COM}}$
$(\mathsf{Yao}, \mathsf{Eval})$
$(\mathsf{Shift}, \mathsf{ShiftEval})$
Section 4
$\mathcal{F}_{\mathsf{WCOM}}$
Section 4
$\binom{n}{u}$-$\mathcal{F}_{\mathrm{OT}}(2\mu)$
$(\mathrm{ssecc}^{t,n,d,u}, \mathrm{enc}^{t,n,d,u})$
[NNOB12] and [NP99]
$\mathcal{F}_{\mathrm{OT}}$
prg^k

Figure 1. Diagram of the whole protocol and its construction.

1.3 Organization

We start with preliminaries and background in Section 2. We then continue to go through the overall description of the secure-two party computation protocol in Section 3. This is followed by Section 4 where we describe the main technical contribution of this paper; the XOR homomorphic commitments.

2 Background

In this section we formalize our goal in the (now standard) real/ideal-world simulation based universally composable security definitions for two-party computation (refer to textbooks such as [Gol04, HL10] for definitions). We furthermore list the basic building blocks of our protocol and quickly review their individual constructions.

2.1 The Ideal Functionality

In Fig. 2 the ideal functionality for secure function evaluation is presented (taken almost verbatim from [NO09]). Note that the functionality is insecure in the sense that A can try to guess B's input bits, but if her guess is wrong B is told that A is cheating. This models a standard problem in Yao based protocols known as "selective failure attack", that can be solved by modifying the circuit to be evaluated. For instance, to evaluate a circuit $\mathcal{C}\left((a_i)_{i\in[\ell]}, (b_i)_{i\in[\ell]}\right)$ securely one could instead evaluate $\mathcal{C}'\left((a_i)_{i\in[\ell]}, (b_{i,j})_{i\in[\ell],j\in[k]}\right) =$

$\mathcal{C}\left((a_i)_{i\in[\ell]}, (\oplus_{i\in[k]} b_{i,j})_{j\in[\ell]}\right)$ i.e., B encodes his real input bit in the parity of a k bit-long string, and the modified circuit first reconstructs the real input and then evaluates the original circuit. Now, in order to guess one of B's real input bits A needs to guess correctly the k random bits, so she will fail with probability $1-2^{-k}$.

As our construction allows to compute XOR and NOT gates for free, this increases only marginally the time needed to construct and evaluate the circuit. On the other hand, this increases the number of OT's needed during the input phase by a factor k (but an OT extension can be used – so this is also not too bad). Better encodings can be used (See [LP07]) to reduce the size of the encoded input from $\ell \cdot k$ bits to $\max(4\ell, 8k)$ bits.

Circuit and inputs:

On input $(\texttt{init}, A, k)$ from A and input $(\texttt{init}, B, k)$ from B where $A = (\mathbf{a}, \mathcal{C}_A)$, $B = (\mathbf{b}, \mathcal{C}_B)$ proceed as follows:

1. Let k be a statistical security parameter and let $\mathcal{C}_A$ and $\mathcal{C}_B$ be descriptions of Boolean circuits consisting of NOT, XOR and AND gates computing the corresponding boolean functions f_A, respectively f_B.
2. Leak $\mathcal{C}_A$, $\mathcal{C}_B$ and k to the adversary.
3. If $\mathcal{C}_A \neq \mathcal{C}_B$, then the ideal functionality outputs `disagreement!` to both parties and terminates. Otherwise, let $\mathcal{C} = \mathcal{C}_A$ and parse $\mathcal{C}$ as $(\ell, \mathcal{C}')$, where $\ell \in \mathbb{N}$ and $\mathcal{C}'$ is a circuit with 2ℓ input wires and ℓ output wires. I.e., we potentially add blank wires to make sure that the size of A's input, B's input and the output are the same. Thus $\mathcal{C}'$ computes the boolean function $f = f_A$.
4. Finally parse $\mathbf{a}$ as $\mathbf{a} \in \{0,1\}^\ell$ and $\mathbf{b} \in \{0,1\}^\ell$ and return $(\ell, \mathcal{C}')$ to both A and B.

Corrupt A:

On input $(\texttt{corrupt})$ from A, let her be corrupt. She can then specify a set $\{(i, \beta_i)\}_{i\in I}$, where $I \subseteq \{1, \ldots, k\}$ and $\beta_i \in \{0,1\}$. If $\beta_i = b_i$ for $i \in I$, then output `correct!` to A. Otherwise, output `You were nicked!` to A and output `Alice cheats!` to B.

Evaluation:

If both parties are honest or A was not caught above, then on input $(\texttt{evaluate})$ from both A and B the ideal functionality computes $\mathbf{z} = f(\mathbf{a}, \mathbf{b})$ and outputs $\mathbf{z}$ to A. The adversary decides the time of delivery.

Figure 2. The ideal functionality, $\mathcal{F}_{\mathsf{SFE}}$, for secure function evaluation for two parties

2.2 Building Blocks

We here review the main building blocks of our protocol.

Generic Free-XOR Yao Gate. Our protocol works with every "free-XOR compatible" garbling scheme. In particular, it is possible to use very optimized garbling schemes. We now describe such a garbling scheme that combines the state of the art optimizations for Yao Gates i.e., free-XOR [KS08], permutation bits [NPS99], garbled row-reduction [NPS99] in the same way as [BHR12].

In particular this means that to garble a gate 4 evaluations of AES are needed, and a garbled gate consists of only 3 ciphertexts (therefore saving on communication complexity). The evaluation of the gate consists of a single AES evaluation.

– We have a (possibly randomized) algorithm $\mathsf{Yao}(L_0, R_0, \Delta, id)$ with id a unique gate identifier, a left input *zero-key* $L_0 \in \{0,1\}^t$, a right input zero-key $R_0 \in \{0,1\}^t$ and a global difference $\Delta \in \{0,1\}^t$ outputs a garbled gate gg and a output zero-key $O_0 \in \{0,1\}^t$.
– We have a (possibly randomized) algorithm $\mathsf{Eval}(gg, L', R')$ that on input a garbled gate gg, a left key $L' \in \{0,1\}^t$ and a right key $R' \in \{0,1\}^t$ outputs an output key $O' \in \{0,1\}^t \cup \{\perp\}$.
– We define the *one-keys* L_1,R_1,O_1 s.t. $L_0 \oplus L_1 = R_0 \oplus R_1 = O_0 \oplus O_1 = \Delta$.

The idea is that a garbled AND gate gg has a zero- and a one-key associated with each of its wires (left input, right input and output wire), and that these keys represent the bit values on those wires. E.g., if gg is a garbled AND gate generated as $(gg, O_0) \leftarrow \mathsf{Yao}(L_0, R_0, \varDelta, id)$ then $\mathsf{Eval}(gg, L_a, R_b)$ for any $a, b \in \{0,1\}$ should output $O_{a\wedge b}$.

Note that if A samples $\varDelta$ and a zero-key, say L_0, at random and give the key L_a to B then there is no way for B to infer the bit a from L_a. Furthermore, even if B learns a he cannot guess the key L_{1-a} with better probability than guessing $\varDelta$. For a garbling scheme to be secure we want that even if B learns gg and keys L_a and R_b for $a, b \in \{0,1\}$, and is able to evaluate $O_{a\wedge b} \leftarrow \mathsf{Eval}(gg, L_a, R_b)$, then he cannot guess L_{1-a}, R_{1-b} or $O_{1-a\wedge b}$ with better probability then guessing the random string $\varDelta$, even if he knows a and/or b.

Thus B can evaluate the garbled gate gg without knowing anymore about the output than he can infer from his knowledge of a and b. Furthermore, B cannot evaluate the gate on any other inputs. Thus if B sends back $O_{a\wedge b}$ to A, A can learn $a \wedge b$ (as she knows O_0 and $\varDelta$) and be confident that this is the correct result.

We formalize this intuition about correctness and security of a garbled gate in definition Def. 1.

Definition 1. *We say that* $(\mathsf{Yao}, \mathsf{Eval})$ *is a* Yao free-XOR garbling scheme *if the following holds:*

Correctness: *Let* $(gg, O_0) \leftarrow \mathsf{Yao}(L_0, R_0, \varDelta, id)$*, then for all* $a, b \in \{0,1\}$

$$\mathsf{Eval}(gg, L_a, R_b) = O_{a\wedge b}$$

with overwhelming probability over the choices of $L_0, R_0, \varDelta$ *and the random coins of* Yao *and* Eval*.*

Secrecy: *Consider the following indistinguishability under chosen input attack game for a stateful adversary* $\mathcal{A}$*.*

$$\begin{array}{l}
\mathsf{IND\text{-}CIA}^{\mathcal{A}}_{(\mathsf{Yao},\mathsf{Eval})}(t) \\
\hline
(a_0, b_0) \leftarrow \mathcal{A}(1^t),\ \textit{where}\ (a_0, b_0) \in \{0,1\}^{2k} \\
(a_1, b_1) \leftarrow \mathcal{A}(1^t),\ \textit{where}\ (a_1, b_1) \in \{0,1\}^{2k} \\
c \leftarrow \{0,1\},\ \varDelta \leftarrow \{0,1\}^t \\
\left(L_0^i, R_0^i\right) \leftarrow \{0,1\}^{2t},\ \textit{for}\ i = 1, \ldots k \\
d \leftarrow \mathcal{A}\left(\left\{\mathsf{Yao}\left(L_0^i, R_0^i, \varDelta, i\right), L^i_{a^i_c}, R^i_{b^i_c}\right\}_{i\in[k]}\right)
\end{array}$$

The adversary outputs two pairs of bit vectors $\left(a_0^i, b_0^i\right)_{i\in[k]}, \left(a_1^i, b_1^i\right)_{i\in[k]} \in \{0,1\}^{2k}$*. The game picks a uniformly random challenge* $c \in_{\mathrm{R}} \{0,1\}$*, samples* $\varDelta \in_{\mathrm{R}} \{0,1\}^t$ *and for* $i = 1, \ldots, k$ *it samples* $L^i, R^i \in_{\mathrm{R}} \{0,1\}^t$*, samples* $gg^i \leftarrow \mathsf{Yao}\left(L_0^i, R_0^i, \varDelta\right)$ *and then inputs* $\left(gg^i, L^i_{a^i_c}, R^i_{b^i_c}\right)_{i\in[k]}$ *to* $\mathcal{A}$*. Finally* $\mathcal{A}$ *outputs a bit* $d \in \{0,1\}$ *and wins if* $d = c$*. We say that the scheme is IND-CIA if for all PPT* $\mathcal{A}$*,* $\mathcal{A}$ *wins the* $\mathsf{IND\text{-}CIA}$ *game with at most negligible advantage in* t*.*

In Fig. 3, a free-XOR garbled gates construction, with optimizations, is presented. We briefly sketch its properties: for correctness, remember that the requirement for correctness is: Let $(gg, O_0) \leftarrow \mathsf{Yao}(L_0, R_0, \varDelta, id)$, then for all $a, b \in \{0,1\}$

$$\mathsf{Eval}(gg, L_a, R_b) = O_{a\wedge b}$$

except with negligible probability over the choices of $L_0, R_0, \varDelta$ and the random coins of Yao and Eval.

This is the case because, if we let $L' = L_a, R' = R_b$, then $lsb(L') = a \oplus p_L$ and $lsb(R') = b \oplus p_R$. Then by construction $\tau^j = a \wedge b$ for $j = 2p_L + p_R$ and

$$\alpha_j = O_{\tau^j} \oplus \mathsf{KDF}(L_a, R_b, id)$$

and therefore

$$O' = O_{\tau^j} = O_{(p_L \oplus a)\wedge(p_R \oplus b)}$$

This garbling scheme can be proven secure under the assumption that AES (with a fixed keys) behaves like a random permutation – see [BHR12] and references therein. Note that, as with any other free-XOR based construction, we need to assume that AES satisfies some kind of related key-attack security [CKKZ12].

We used the assumption that $lsb(\varDelta) = 1$. If this is not the case, then $p_{L_0} = p_{L_1}$ and therefore during the cut-and-choose phase of the protocol described in Section 3 when B challenges on $(a, b) \neq (0, 0)$ an error will occur.

Notation, Convention:

We call the least significant bit of every key the permutation bit and write $p_K = lsb(K)$ and we assume that $lsb(\Delta) = 1$ – see discussion at the end of this section.

Define, for some constant value *const*:

$$\mathsf{KDF}(L, R, id) = (L \oplus 2R \oplus id) \oplus \mathsf{AES}_{const}(L \oplus 2R \oplus id)$$

where $2R$ is multiplication in $GF(2^t)$.

Garbling, $\mathsf{Yao}(L_0, R_0, \Delta, id)$:

With $L_0, R_0, \Delta \in \{0,1\}^t$ and id a unique identifier do the following:

1. Define
$$\begin{aligned}\tau^0 &= p_L \wedge p_R\\ \tau^1 &= p_L \wedge \overline{p_R}\\ \tau^2 &= \overline{p_L} \wedge p_R\\ \tau^3 &= \overline{p_L} \wedge \overline{p_R}\end{aligned}$$
2. Compute $O_0 = \mathsf{KDF}\left(L_{p_L}, R_{p_R}, id\right) \oplus (\tau^0 \cdot \Delta)$;
3. Compute:
$$\begin{aligned}\alpha_1 &= O_{\tau^1} \oplus \mathsf{KDF}\left(L_{p_L}, R_{\overline{p_R}}, id\right)\\ \alpha_2 &= O_{\tau^2} \oplus \mathsf{KDF}\left(L_{\overline{p_L}}, R_{p_R}, id\right)\\ \alpha_3 &= O_{\tau^3} \oplus \mathsf{KDF}\left(L_{\overline{p_L}}, R_{\overline{p_R}}, id\right)\end{aligned}$$
4. Output a garbled gate $gg = (id, \alpha_1, \alpha_2, \alpha_3)$ and the zero output key O_0.

Evaluation, $\mathsf{Eval}(gg, L', R')$:

With $L', R' \in \{0,1\}^t$ do the following:

1. Parse $gg = (id, \alpha_1, \alpha_2, \alpha_3)$. Define $\alpha_0 = 0^t$.
2. Compute $p_{L'} = lsb(L')$ and $p_{R'} = lsb(R')$ and let $j = 2p_{L'} + p_{R'}$.
3. Return $O' = \alpha_j \oplus \mathsf{KDF}(L', R', id)$;

Figure 3. Free-XOR Garbled Gates Construction

Soldering. The idea for this component is the same as in [NO09], however, slightly changed to support a general gate garbling scheme.

When a garbled gate gg^1 has the same zero-key (and therefore also one-key) associated to one of its wires, as is associated with one of gg^2's wires, we say that the given wire of gg^1 is *soldered* to the given wire of gg^2. This is a useful concept when we want to build circuits of garbled gates. To see this consider a garbled gate gg^1 with its left input wire soldered to the output of gg^2, and its right input wire soldered to the output of gg^3. This means that if gg^2 and gg^3 has output zero-keys O_0^2 and O_0^3 respectively, then gg^1 has left and right zero-keys $L_0^1 = O_0^2$ and $R_0^1 = O_0^3$. Thus if we evaluate gg^2 and gg^3 on some input and obtain output keys O_a^2 and O_b^3 we can use this to further evaluate gg^1 on these outputs. The resulting output would be some output key $O^1_{a \wedge b}$.

Regarding XORs notice that if gg^1 has, e.g., left, input zero-key $L_0^1 = O_0^2 \oplus O_0^3$ then

$$O_a^2 \oplus O_b^3 = O_0^2 \oplus O_0^3 \oplus (a \oplus b)\Delta = L_0^1 \oplus (a \oplus b)\Delta = L^1_{a \oplus b} \ .$$

In this case we say that the left input wire of gg^1 is soldered to the XOR of the output of gg^2 and gg^3. This is because by XOR'ing the outputs keys of gg^2 and gg^3 we get the left input key of gg^1 corresponding to XOR of the outputs of gg^2 and gg^3. This is also why we call the garbling *free-XOR*: we do not need to garble XOR gates, since this is handled by the soldering.

In our protocol we will first generate garbled gates where all zero-keys are picked independently, and then in a later stage we will *solder* the wires of the garbled gates to each other to form a garbled circuit.

For this purpose, we introduce a function $\mathsf{Shift}\left(gg, L^d, R^d, O^d\right)$ that on input a garbled gate gg, generated as $(gg, O_0) \leftarrow \mathsf{Yao}\left(L_0, R_0, \Delta, id\right)$, and three *differences* $L^d, R^d, O^d \in \{0,1\}^t$, outputs a new *shifted gate sgg*. The shifted gate sgg is the gate gg modified to have have input zero-keys $\left(L_0 \oplus L^d\right)$ and $\left(R_0 \oplus R^d\right)$ and output zero-key $\left(O_0 \oplus O^d\right)$.

This can be implemented by letting Shift output the concatenation of its inputs i.e., $sgg = \left(gg, L^d, R^d, O^d\right)$ and let the evaluation of a shifted gate sgg be defined by:

$$\mathsf{ShiftEval}\left(sgg, \hat{L}, \hat{R}\right) = \mathsf{Eval}\left(gg, \hat{L} \oplus L^d, \hat{R} \oplus R^d\right) \oplus O^d$$

where for all K we define $\bot \oplus K = \bot$. It is clear that a shifted gate is correct (with respect to the shifted zero-keys) if and only if a standard gate is correct, and clearly shifting a gate does not threaten its security property. A shifted gate can be shifted again: The Shift function will just update the values L^d, R^d, O^d accordingly.

Consider two garbled gates $\left(gg^1, O_0^1\right) \leftarrow \mathsf{Yao}\left(L_0^1, R_0^1, \Delta, 1\right)$ and $\left(gg^2, O_0^2\right) \leftarrow \mathsf{Yao}\left(L_0^2, R_0^2, \Delta, 2\right)$. The shifted gate $sgg^2 = \mathsf{Shift}\left(gg^2, \left(O_0^1 \oplus L_0^2\right), 0, 0\right)$ then becomes a garbled gate with left zero-key $L_0^2 \oplus \left(O_0^1 \oplus L_0^2\right) = O_0^1$. I.e. the output wire of gg^1 is now soldered to the left input wire of sgg^2.

Similarly we could have used the Shift function to solder the input of sgg^2 to the XOR of some other garbled gates.

If one wish to use NOT gates then these can be implemented as part of this shifting by a simply change in the the difference, i.e., to add a NOT gate to the soldering of the left wire of a gate we simply use $\neg L^d = L^d \oplus \Delta$ instead of just L^d.

To see this assume we want to put a NOT into the soldering between the output wire of gg^1 and the left wire of gg^2. In this case we would have $\neg {L^2}^d = {L^2}^d \oplus \Delta = O_0^1 \oplus L_0^2 \oplus \Delta$, i.e., $sgg^2 = \mathsf{Shift}\left(gg^2, \left(O_0^1 \oplus L_0^2 \oplus \Delta\right), 0, 0\right)$. Thus when the evaluator does

$$\mathsf{ShiftEval}\left(sgg^2, L^2, 0, 0\right) = \mathsf{Eval}\left(gg^2, L_a^2 \oplus (O_0^1 \oplus L_0^2 \oplus \Delta), R^2\right) \ .$$

If $a = 0$ we get the left input key for gg^2 is $L_0^2 \oplus (O_0^1 \oplus L_0^2 \oplus \Delta) = O_0^1 \oplus \Delta = O_1^1$ and similarly for $a = 1$ we get $L_1^2 \oplus (O_0^1 \oplus L_0^2 \oplus \Delta) = (L_0^2 \oplus \Delta) \oplus O_0^1 \oplus (L_0^2 \oplus \Delta) = O_0^1$. Thus we clearly see that the bit represented by the left input key (along with the key itself) for gg^2 has been flipped.

```
Shift $\left(gg, L^d, R^d, O^d\right)$:
    1. Assume $gg$ has been constructed by the call Yao$(L_0, R_0, \Delta, id)$ such that $gg = (id, \alpha_1, \alpha_2, \alpha_3)$.
    2. Output $sgg = \left(gg, L^d, R^d, O^d\right)$.
ShiftEval $\left(gg, \hat{L}, \hat{R}\right)$:
    1. Assume $sgg$ has been constructed by a call Shift $\left(gg, L^d, R^d, O^d\right)$, i.e., $sgg = \left(gg, L^d, R^d, O^d\right)$.
    2. Return Eval $\left(gg, \hat{L} \oplus L^d, \hat{R} \oplus R^d\right) \oplus O^d$.
```

Figure 4. The shifting procedures for free-XOR Garbled Gates.

Homomorphic commitments. To securely implement the soldering described above, we cannot simply have (potentially malicious) A send the differences needed to shift the gates. Instead we will have A give homomorphic commitments to all zero-keys of each gate, and then have her open the differences of the committed keys. Therefore we need a homomorphic commitment scheme. In Fig. 5 we state the ideal functionality $\mathcal{F}_{\mathsf{COM}}$ for such homomorphic commitments. We then show how to implement these in Section 4.

The functionality allows A to commit to messages and to later reveal those messages. In addition the functionality allows to reveal the XOR of two or more committed messages to B (without revealing any extra information about the original committed messages).

Setup

On input $(\mathtt{init}, ID, W)$ from the adversary, with $|ID| = \mu$, $|W| \leq \kappa$ and $W \subset ID$, output ID to both A and B and let $\mathcal{J} = \emptyset$. If A is honest, then $W = \emptyset$.

Commit

On input $(\mathtt{commit}, j \in ID, m_j)$ with $m_j \in \{0,1\}^t$ from A, and where no value of the form $(j, \cdot)$ is stored, store (j, m_j). If $j \in ID \setminus W$, add $J = \{j\}$ to $\mathcal{J}$ and associate with J the equation $\mathtt{X}_j = m_j$. Then output $(\mathtt{commit}, j)$ to B.

Open

On input $(\mathtt{open}, J \subset ID)$ from A, where for all $j \in J$ a pair (j, m_j) is stored do the following:

- If A is honest, output $(\mathtt{open}, J, \oplus_{j \in J} m_j)$ to B.
- If A is corrupted wait for A to input $(\mathtt{corrupt\text{-}open}, J, m_J)$. Then add J to $\mathcal{J}$, associate the equation $\oplus_{j \in J} \mathtt{X}_j = m_J$ to J, and check that the equation system $\{\oplus_{j \in J} \mathtt{X}_j = m_J\}_{J \in \mathcal{J}}$ has a solution. If so, output $(\mathtt{open}, J, m_J)$ to B. Otherwise, output `Alice cheats` to B and terminate.

Oblivious-Opening

On input $(\mathtt{OT\text{-}choose}, \mathit{otid}, b)$ with $b \in \{0,1\}$ from B output $(\mathtt{OT\text{-}choose}, \mathit{otid})$ to A. On input $(\mathtt{OT\text{-}open}, \mathit{otid}, J_0, J_1)$ from A with $J_0, J_1 \subset ID$ where for all $j \in J_0, J_1$ a pair (j, m_j) is stored and $(\mathtt{OT\text{-}choose}, \mathit{otid}, *)$ was input before by B do the following:

- If A is honest, output $(\mathtt{OT\text{-}open}, \mathit{otid}, J_b, \oplus_{j \in J_b} m_j)$ to B (Note that B does not learn the set of ids J_{1-b}).
- If A is corrupted, wait for A to input $(\mathtt{guess}, g)$ with $g \in \{0, 1, \perp\}$. If $g \in \{0,1\}$ and $g \neq b$ output `Alice cheats` to B and terminate. Otherwise, proceed to wait for A to input $(\mathtt{corrupt\text{-}open}, J_0, J_1, m_{J_0}, m_{J_1})$. Add J_b to $\mathcal{J}$ and associate the equation $\oplus_{j \in J_b} \mathtt{X}_j = m_{J_b}$ to J_b. Check that the equation system still has a solution as described above. If so, output $(\mathtt{OT\text{-}open}, J_b, m_{J_b})$ to B. Otherwise output `Alice cheats` to B.

Or-Opening

For up to ω Or-Openings, that must all occur before the first Oblivious-Opening, do the following: On input $(\mathtt{OR\text{-}open}, J_0, J_1, a)$ from A, with $J_0, J_1 \subset ID, a \in \{0,1\}$ where for all $j \in J_0, J_1$ a pair (j, m_j) is stored do the following:

- If A is honest, output $(\mathtt{OR\text{-}open}, J_0, J_1, \oplus_{j \in J_a} m_j)$ to B.
- If A is corrupted, and if $J_a \cap W \neq \emptyset$, wait for corrupt A to input $(\mathtt{corrupt\text{-}open}, J_a, m_{J_a})$, add J_a to $\mathcal{J}$ and associate $\oplus_{j \in J_a} \mathtt{X}_j = m_{J_a}$ to J_a. Check if the equation system still has a solution as described above. If so, output $(\mathtt{OR\text{-}open}, J_0, J_1, m_{J_a})$ to B. Otherwise output `Alice cheats` to B.

before the first Oblivious-Opening.

Figure 5. The ideal functionality, $\mathcal{F}_{\mathsf{COM}}$, for the commitment scheme used by π_{LEGO}.

The functionality is "insecure", in the sense that A can choose a set of up to κ wildcard commitments where she can change her mind about the committed value at opening time. However, openings need to be consistent. More specifically, the $\mathcal{F}_{\mathsf{COM}}$ functionality stores a system of linear equations. Initially these equations simply specify that non-wildcard commitments must be opened to the value, they were commitments to. Every time A performs an opening involving wildcard commitments this defines a new linear equation, which is stored in the ideal functionality. For an opening of a wildcard commitment to be successful the set of linear equations stored in the ideal functionality must be consistent.

If the set of equations stored in the ideal functionality restricts the opening of a commitment in such a way that it can only be opened to *one* value, we say that the commitment is *fixed* to that value. Note, that all non-wildcard commitments are fixed, and a fixed wildcard commitment can essentially be viewed as a non-wildcard commitment.

As we are treating the commitments as an ideal functionality, we need to push into the ideal functionalities two extra commands (in a way similar to the commit-and-prove functionality in [CLOS02])): apart from the regular openings the functionality allows to open (the XOR of) committed messages in two alternative ways: In an *Oblivious-Opening*, B can choose between two sets of committed messages and learn the XOR of the messages in one of them. In an *Or-Opening* we allow A to open the XOR of one out of two sets of committed messages without revealing which one. For technical reasons there can only be a total of ω Or-Openings and all Or-Openings must be done before the first Oblivious-Opening. Also, note that there is a build-in selective failure attack in the Oblivious-Opening. However, this is not a problem as we will only use this type of opening to handle B's input where, as discussed above, the $\mathcal{F}_{\mathsf{SFE}}$ functionality already allows a selective failure attack.

Commitment from B to A. Additional to the $\mathcal{F}_{\mathsf{COM}}$ functionality we are going to use an extractable commitment Com. This commitment is used only once by B to commit to his challenge in the cut-and-choose phase and extraction is needed for simulation (to avoid selective opening issues). Since this commitment does not need to be homomorphic it can be easily implemented in the $\mathcal{F}_{\mathrm{OT}}$-hybrid model.

3 The MiniLEGO Protocol

We now show how to use the ingredients outlined in the previous section in order to construct the MiniLEGO protocol.

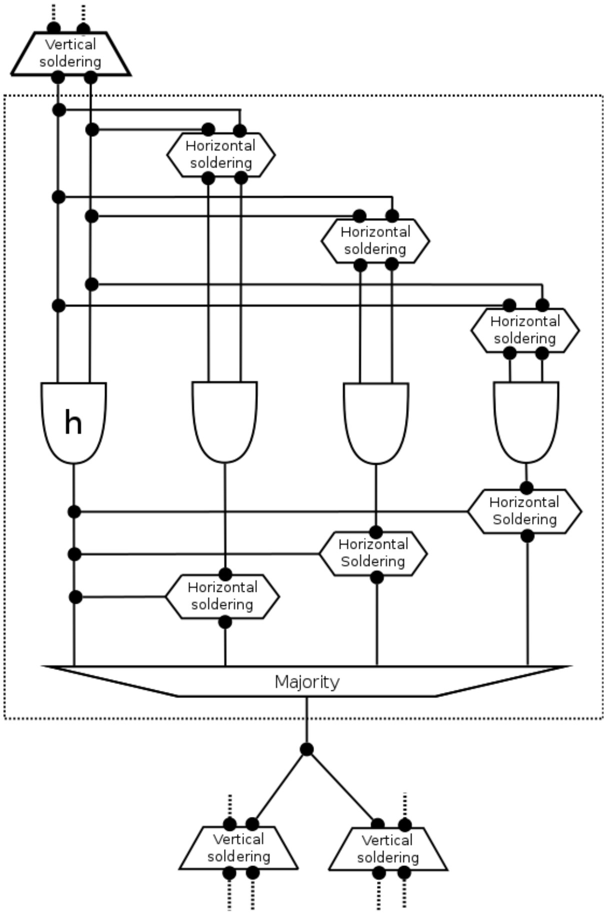

Figure 6. Illustration of a bucket (computing AND) when $\rho = 4$ and how the horizontal and vertical soldering are connected.

Plaintext Circuit. We denote by $\mathcal{C}'$ the Boolean circuit to be evaluated. We assume $\mathcal{C}'$ to be composed of NOT, XOR and AND gates. The XOR gates are allowed to have unbounded fan-in while the AND gates have fan-in 2. With each AND gate in $\mathcal{C}'$ we associate a unique label and we let gates be the set of all these labels. A subset $\mathsf{inputGates} \subset \mathsf{gates}$ of size 2ℓ are specially marked as input gates. The AND gates in inputGates should be given the same bit on both input wires, so that the gate simply computes the identity function. A subset in $\mathsf{Ainputs} \subset \mathsf{inputGates}$ of size ℓ are taken to be A's inputs. The remaining ℓ gates in $\mathsf{Binputs} = \mathsf{inputGates} \setminus \mathsf{Ainputs}$ are B's inputs (for convenience assume that $\mathsf{Binputs} = [\ell]$). A has input bits $(a_1, \ldots, a_\ell)$, while B has input bits $(b_1, \ldots, b_\ell)$.

A subset $\mathsf{outputGates} \subset \mathsf{gates}$ of size ℓ are marked as output gates. The output of these gates are taken to be the output of the circuit. Note that this means that all output gates are AND gates. However, this is without loss of generality: Any circuit with one or more XOR gates as output gates can easily be modified to an equivalent circuit with AND gates as output gates by adding at most ℓ AND gates. The ℓ output bits are denoted $(z_1, \ldots, z_\ell)$.

The wiring of the circuit $\mathcal{C}'$ is described by two functions $\mathsf{lp}, \mathsf{rp} : \mathsf{gates} \setminus \mathsf{inputGates} \to 2^{\mathsf{gates} \cup \{\mathbb{1}\}}$. We call $\mathsf{lp}(j)$ the left parents of j (resp. $\mathsf{rp}(j)$ the right parents of j), and take the left (resp. right) input of j to be the XOR of the output bits of all gates in $\mathsf{lp}(j)$ (resp. $\mathsf{rp}(j)$). Thus, the XOR gates of $\mathcal{C}'$ are implicitly defined by the lp and rp functions. If the special symbol $\mathbb{1}$ is included in the set returned by lp, rp, this means that a NOT gate is inserted between gate j and its parent gate (i.e., the input is XORed with the constant 1). We assume that $\mathcal{C}'$ does not have loops and it is ordered i.e., $\max(\mathsf{lp}(j)) < j$ and $\max(\mathsf{rp}(j)) < j$.

If $\mathsf{lp}(j)$ (resp. $\mathsf{rp}(j)$) is not a singleton, it is intended that the input value of gate j is the XOR of the outputs of the gates in $\mathsf{lp}(j)$ (resp. $\mathsf{rp}(j)$). This is just a convenient notation for our construction, given that XOR gates can be evaluated for free.

Garbled Circuit. Let $\Gamma = 2\rho s$ for $s = |\mathsf{gates}|$ and some replication factor $\rho \in \mathbb{N}$. For our protocol A will construct Γ garbled gates. She constructs twice as many garbled gates as is needed to build the garbled circuit, because half the gates are going to be checked during the cut-and-choose phase. We choose to check exactly half for the sake of presentation but, as in [sS11], this could be changed to any fraction in order to optimize concrete efficiency.

Bucket Notation. In the protocol individual garbled gates are combined together in "buckets" of gates. We introduce here some convenient notation that allow us to address the gates in a bucket, the bucket of a gate etc. Let $\mathcal{B}$ be the family of ρ-to-1, ρ-wise independent functions from a set $U \subset [\Gamma]$ of size ρs to gates. For a function $\mathsf{BucketOf} \in \mathcal{B}$ let Bucket be the function that, for all $j \in \mathsf{gates}$ outputs the set $\{i \in U | \mathsf{BucketOf}(i) = j\}$. Let $\mathsf{BucketHead}(j)$ be the function that returns the "first" (in lexicographic order) element of $\mathsf{Bucket}(j)$.

There are $\Gamma' = 3\Gamma + 1$ keys in the protocol, because every constructed AND gate has a left, right and output key and in addition there is a global difference Δ. The key index is written as a superscript while subscripts are in $\{0, 1\}$ and describe the value carried by the key i.e., $K_b^i = K^i \oplus (b\Delta)$. Let id be a function that on input a key $K_0^j \in \{0,1\}^t$ returns a unique label for that key. We will sometimes abuse notation and write $\mathsf{id}\left(K_1^j\right)$ to denote the set $\left\{\mathsf{id}\left(K_0^j\right), \mathsf{id}\left(\Delta\right)\right\}$. This will simplify the notation when using the $\mathcal{F}_{\mathsf{COM}}$ functionality.

3.1 Protocol Specification

The protocol π_{LEGO} in Fig. 7 progresses in six phases: **Setup**, **Garbling**, **Cut-and-choose**, **Soldering**, **Input** and **Evaluation**. Here we describe these phases one by one.

During **Setup**, A or B initialize the $\mathcal{F}_{\mathsf{COM}}$ functionality by calling $(\mathtt{init}, ID, W)$ with $|ID| = \Gamma'$ and $|W| \leq k$. For the remainder of the protocol if $\mathcal{F}_{\mathsf{COM}}$ outputs `Alice cheats`, B will abort the protocol. Then A samples the global difference Δ and commits to it using the `commit` command in $\mathcal{F}_{\mathsf{COM}}$. B samples his challenge for the cut-and-choose phase and the $\mathsf{BucketOf}$ function as described above, and commits to both using the extractable commitment Com. B also "commits" to his input using the `OT-choose` command of the $\mathcal{F}_{\mathsf{COM}}$ functionality. These commitments of B's are needed to avoid selective opening issues in the cut-and-choose phase and reduce the security of the protocol to the $\mathsf{IND\text{-}CIA}$ game.

In **Garbling**, A constructs the candidate garbled gates $(gg^i)_{i \in [\Gamma]}$ and commits to the input/output zero-keys of each garbled gate using $\mathcal{F}_{\mathsf{COM}}$.

In **Cut-and-choose**, B reveals his challenge. The challenge consists of a set of indices $T \subset [\Gamma]$ of size ρs and a sequence of bits $(u_i, v_i)_{i \in T}$, indicating that B wants to test garbled gate gg^i on input (u_i, v_i). A opens the corresponding input and output keys for the test gates, allowing B to check for correctness. Note that B only tests one set of inputs for each gate – otherwise he will learn Δ.

In the remainder of the protocol the garbled gates that are not checked in **Cut-and-choose**, those with indices in $U = [\Gamma] \setminus T$, are used to build a garbled circuit according to the following fault tolerant circuit design: With each gate $j \in \mathsf{gates}$ we associate a *bucket* of ρ AND gates. To evaluate gate j we will evaluate each gate in the bucket of j on the inputs given to j. If more than $\lfloor \rho/2 \rfloor$ of the gates in the bucket agree on their output bit, we take this bit to be the output of j (otherwise the output is $\perp$). Clearly if there are more than $\lfloor \rho/2 \rfloor$ non-faulty gates in each bucket the output obtained in this way is correct. See Fig. 6 for a pictorial description.

To build such a garbled circuit the gates that were not checked $(gg^i)_{i \in U}$ are assigned into buckets using the $\mathsf{BucketOf}$ function. Then B uses the Shift function as described in Section 2 to solder the wires of the garbled gates. Note that since A may be malicious we cannot simply have her sent the XOR's of zero-keys that B needs for soldering. Instead A reveals the XOR's by opening the corresponding commitments to the zero-keys.

In this way the garbled circuit is constructed in **Soldering** in three different soldering steps: For all $j \in \mathsf{gates}$ **Horizontal Soldering** solders all wires of all gates in $(gg^i)_{i\in\mathsf{Bucket}(j)}$ to the corresponding wires of $gg^{\mathsf{BucketHead}(j)}$. This allows to evaluate all the gates in the same bucket on the same input keys and get the same output keys. I.e., if A is honest, after the horizontal soldering all the gates in one bucket have exactly the same keys. For all $j \in \mathsf{gates}$ **Vertical Soldering** solders the left input wire of $gg^{\mathsf{BucketHead}(j)}$ to the XOR of the output wires of $\left(gg^{\mathsf{BucketHead}(i)}\right)_{i\in\mathsf{lp}(j)}$, and the right input wire of $gg^{\mathsf{BucketHead}(j)}$ to the XOR of the output wires of $\left(gg^{\mathsf{BucketHead}(i)}\right)_{i\in\mathsf{rp}(j)}$ (and we use the convention $O^{\mathbb{1}} = \Delta$ to deal with NOT gates – note that Δ can be seen as the 1 key of a special wire with zero-key equal to 0^t). Note that since **Horizontal Soldering** made all garbled gates in a bucket have the same input keys, this essentially means soldering *all* the gates in the bucket to the output wires of gates in $(\mathsf{Bucket}(i))_{i\in\mathsf{lp}(j)}$ and $(\mathsf{Bucket}(i))_{i\in\mathsf{rp}(j)}$. I.e., vertical soldering is "functional", in the sense that it make sure that the garbled circuit computes the right circuit, $\mathcal{C}'$. For all $j \in \mathsf{inputGates}$ **Input Soldering** simply solders the left and right input wire of garbled gates in $\mathsf{Bucket}(j)$ to each other. This means that the gates in $\mathsf{inputGates}$ simply compute the identity function.

In **Input**, for all $j \in \mathsf{Ainputs}$ A uses the **Or-Opening** command of $\mathcal{F}_{\mathsf{COM}}$ to open the input key to the garbled gates in $\mathsf{Bucket}(j)$ corresponding to her input bit. For all $j \in \mathsf{Binputs}$ B also learns the input key to the garbled gates in $\mathsf{Bucket}(j)$ corresponding to his input bit, using the **Oblivious-Opening** command.

Given the initial input keys in **Evaluation** B evaluates each bucket of garbled gates in the following way: He evaluates each gate in the bucket on the left and right input keys for that bucket. If a key appears more than $\lfloor\rho/2\rfloor$ times as the output key of the garbled gates in the bucket, he takes this to be the output key of the bucket. If no such key exists B aborts. Note that by the way we soldered the garbled circuit, this corresponds exactly to the fault tolerant circuit we described above. Finally B provides A with the output keys. Knowing Δ, A can decipher the output keys and obtain the output values.

Theorem 1. *Let k be the security parameter, $\rho = O(k/\log(s))$. If* $(\mathsf{Yao}, \mathsf{Eval})$ *is an* $\mathsf{IND\text{-}CIA}$ *secure* Yao free-XOR garbling scheme *then the protocol π_{LEGO} in Fig. 7 UC, active, static securely implements $\mathcal{F}_{\mathsf{SFE}}$ in the $(\mathcal{F}_{\mathsf{COM}})$-hybrid model (initialized with $(\boldsymbol{init}, ID, W)$ for $|ID| = 3\Gamma + 1$ and $|W| \le k$).*

Analysis. A full proof of Theorem 1 can be found in Appendix A. However, we quickly sketch the idea of the proof. First considering a corrupted B and then considering a corrupted A.

Corrupted B. B does not receive any output nor has any real way of cheating in the protocol (in the output phase, if B changes the output key in a way that makes A accept, then he must have guessed Δ, thus breaking the $\mathsf{IND\text{-}CIA}$ game). Essentially, we only need to argue that his view does not leak any information, thanks to the $\mathsf{IND\text{-}CIA}$ security of the garbling scheme. Note that in the protocol B starts by committing to his input and challenge for the cut-and-choose phase. This allows the simulator S to extract all this information at the beginning of the simulation (and provide input on behalf of corrupted B to the ideal functionality). Then we reduce the security of the protocol to the $\mathsf{IND\text{-}CIA}$ security of the garbling scheme: The simulator knows in fact T and U before it sends the gates to B, therefore S will place honestly constructed gates in T (for which it knows the openings and therefore can easily simulate the cut-and-choose test – remember that the simulator fully controls $\mathcal{F}_{\mathsf{COM}}$) and the challenge garbled gates from the $\mathsf{IND\text{-}CIA}$ game in U: that is, the simulator produces a view such that distinguishing between a real and a simulated execution is equivalent to winning the $\mathsf{IND\text{-}CIA}$ game.

Corrupted A. Essentially, the proof of security boils down to proving correctness. By the design of the garbled circuit correctness follows when there is more than $\lfloor\rho/2\rfloor$ correct gates in each of the buckets.

The LEGO approach ensures that if A passes the cut-and-choose test, then with overwhelming probability there are at most k faulty gates left in U. Those faulty gates are then randomly assigned into buckets, this means that with overwhelming probability each bucket will have a majority of correct gates.

Setup

Choose $\rho = O(k/\log(s))$ and $\mathit{\Gamma} = 2\rho s$ where $s = |\mathsf{gates}|$ in $\mathcal{C}'$. Let $\mathit{\Gamma}' = 3\mathit{\Gamma} + 1$ and proceed as follows:

1. A and B initialize a $\mathcal{F}_{\mathsf{COM}}$ functionality by having either of them call $(\mathtt{init}, ID, W)$ with $|ID| = \mathit{\Gamma}'$ and $|W| \leq k$.
2. A samples $\mathit{\Delta} \in_{\mathrm{R}} \{0,1\}^t$ and inputs $(\mathtt{commit}, \mathsf{id}(\mathit{\Delta}), \mathit{\Delta})$ to $\mathcal{F}_{\mathsf{COM}}$.
3. B samples a random $T \subset [\mathit{\Gamma}]$ of size ρs, and for all $i \in T$ samples $u_i, v_i \in_{\mathrm{R}} \{0,1\}$. Let $U = [\mathit{\Gamma}] \setminus T$.
4. B samples $\mathsf{BucketOf} \in \mathcal{B}$ as described in Section 2.
5. B sends $C_T = \mathsf{Com}(T, (u_i, v_i)_{i \in T}, \mathsf{BucketOf}, r_T)$ to A.
6. For each $j \in \mathsf{Binputs}$, B inputs $(\mathtt{OT\text{-}choose}, j, b_j)$ to $\mathcal{F}_{\mathsf{COM}}$.

Garbling

1. For all $i \in [\mathit{\Gamma}]$, A samples $L_0^i, R_0^i \in_{\mathrm{R}} \{0,1\}^t$, computes $\left(gg^i, O_0^i\right) \leftarrow \mathsf{Yao}\left(L_0^i, R_0^i, \mathit{\Delta}, i\right)$ and sends $GG = \left(gg^i\right)_{i \in [\mathit{\Gamma}]}$ to B.
2. A inputs $\left(\mathtt{commit}, \mathsf{id}\left(L_0^i\right), L_0^i\right)$, $\left(\mathtt{commit}, \mathsf{id}\left(R_0^i\right), R_0^i\right)$ and $\left(\mathtt{commit}, \mathsf{id}\left(O_0^i\right),\ O_0^i\right)$ to $\mathcal{F}_{\mathsf{COM}}$.

Cut-and-choose

1. B sends T, $(u_i, v_i)_{i \in T}$, $\mathsf{BucketOf}$ and randomness r_T to A.
2. If this is not a valid opening of C_T A aborts. Otherwise, for all $i \in T$ A inputs to $\mathcal{F}_{\mathsf{COM}}$ $\left(\mathsf{open}, \mathsf{id}\left(L_{u_i}^i\right)\right)$, $\left(\mathsf{open}, \mathsf{id}\left(R_{v_i}^i\right)\right)$, $\left(\mathsf{open}, \mathsf{id}\left(O_{u_i \wedge v_i}^i\right)\right)$. Let $\hat{L}^i, \hat{R}^i, \hat{O}^i$ be the values output to B by $\mathcal{F}_{\mathsf{COM}}$.
3. B aborts if there is an $i \in T$ so that $\hat{O}^i \neq \mathsf{Eval}\left(gg^i, \hat{L}^i, \hat{R}^i\right)$.

Soldering

1. **Horizontal Soldering:** For all $j \in \mathsf{gates}$, let $h = \mathsf{BucketHead}(j)$: For all $i \neq h \in \mathsf{Bucket}(j)$ A inputs $\left(\mathsf{open}, \left\{\mathsf{id}\left(L^h\right), \mathsf{id}\left(L^i\right)\right\}\right)$, $\left(\mathsf{open}, \left\{\mathsf{id}\left(R^h\right), \mathsf{id}\left(R^i\right)\right\}\right)$, and $\left(\mathsf{open}, \left\{\mathsf{id}\left(O^h\right), \mathsf{id}\left(O^i\right)\right\}\right)$ to $\mathcal{F}_{\mathsf{COM}}$. Let $L^{i^d}, R^{i^d}, O^{i^d}$ be the keys output to B from $\mathcal{F}_{\mathsf{COM}}$ and $sgg^i = \mathsf{Shift}\left(gg^i, L^{i^d}, R^{i^d}, O^{i^d}\right)$.
2. **Vertical Soldering:** For all $j \in \mathsf{gates} \setminus \mathsf{inputGates}$, let $h = \mathsf{BucketHead}(j)$: A inputs $\left(\mathsf{open}, \left\{\mathsf{id}\left(L^h\right)\right\} \cup \left\{\mathsf{id}\left(O^{\mathsf{BucketHead}(i)}\right)\right\}_{i \in \mathsf{lp}(j)}\right)$ and $\left(\mathsf{open}, \left\{\mathsf{id}\left(R^h\right)\right\} \cup \left\{\mathsf{id}\left(O^{\mathsf{BucketHead}(i)}\right)\right\}_{i \in \mathsf{rp}(j)}\right)$ to $\mathcal{F}_{\mathsf{COM}}$. Let L^{h^d}, R^{h^d} be the keys output to B by $\mathcal{F}_{\mathsf{COM}}$ and $sgg^h = \mathsf{Shift}\left(gg^h, L^{h^d}, R^{h^d}, 0^t\right)$.
3. **Input Soldering:** For all $j \in \mathsf{inputGates}$, let $h = \mathsf{BucketHead}(j)$: A inputs $\left(\mathsf{open}, \left\{\mathsf{id}\left(L^h\right), \mathsf{id}\left(R^h\right)\right\}\right)$ to $\mathcal{F}_{\mathsf{COM}}$. Let R^{h^d} be the key output to B by $\mathcal{F}_{\mathsf{COM}}$ and $sgg^h = \mathsf{Shift}\left(sgg^h, 0^t, R^{h^d}, 0^t\right)$.

Input

1. For all $j \in \mathsf{Ainputs}$ let $h = \mathsf{BucketHead}(j)$, A inputs $\left(\mathtt{OR\text{-}open}, \mathsf{id}\left(L_0^h\right), \mathsf{id}\left(L_1^h\right), a_j\right)$ to $\mathcal{F}_{\mathsf{COM}}$.
2. For all $j \in \mathsf{Binputs}$ let $h = \mathsf{BucketHead}(j)$, A inputs $\left(\mathtt{OT\text{-}open}, j, \mathsf{id}\left(L_0^h\right), \mathsf{id}\left(L_1^h\right)\right)$ to $\mathcal{F}_{\mathsf{COM}}$.
3. In both cases let $\hat{L}^h$ be the key output from $\mathcal{F}_{\mathsf{COM}}$ to B. B computes a list of candidate keys $\mathsf{Cand}_j = \left(\mathsf{ShiftEval}\left(sgg^i, \hat{L}^h, \hat{L}^h\right)\right)_{i \in \mathsf{Bucket}(j)}$. If any key appears more than $\lfloor \rho/2 \rfloor$ times in Cand_j name it $\hat{O}^j$, otherwise B aborts.

Evaluate

1. For each gate $j \in \mathsf{gates} \setminus \mathsf{inputGates}$, B computes:
   (a) Left and right keys $\hat{L}^j = \bigoplus_{l \in \mathsf{lp}(j)} \hat{O}^l$ and $\hat{R}^j = \bigoplus_{l \in \mathsf{rp}(j)} \hat{O}^l$.
   (b) The list of output keys $\mathsf{Cand}_j = \left(\mathsf{ShiftEval}\left(sgg^i, \hat{L}^j, \hat{R}^j\right)\right)_{i \in \mathsf{Bucket}(j)}$.
   (c) If a key appears more than $\lfloor \rho/2 \rfloor$ times in Cand_j name it $\hat{O}^j$ and proceed, otherwise abort.
2. For all $j \in \mathsf{outputGates}$, B sends $\hat{O}^j$ to A.
3. For all $j \in \mathsf{outputGates}$, A outputs $z_j = 0$ if $\hat{O}^j = O_0^{\mathsf{BucketHead}(j)}$, $z_j = 1$ if $\hat{O}^j = O_1^{\mathsf{BucketHead}(j)}$ and aborts otherwise.

Figure 7. The Protocol π_{LEGO} implementing $\mathcal{F}_{\mathsf{SFE}}$.

However, as opposed to [NO09] where all commitments were binding, here we have also k wildcard commitments to deal with. This is in problematic, as wildcard commitments can be opened to anything, and we need make sure that this does not break correctness.

To be more specific we say that a garbled gate gg^i is *faulty* if the commitments to its input and output zero-keys are fixed to values L_0^i, R_0^i and O_0^i respectively, and there exists some $a, b \in \{0,1\}$ so that

$\mathsf{Eval}\left(gg^i, L_a^i, R_b^i\right)$ does not output $O_{a\wedge b}^i$ with overwhelming probability. If a gate gg^i has a wire where the commitment to the associated zero-key is not fixed, then we say that this wire is faulty, and gg^i has *faulty wiring*. We say that gg^i is *fault free* if it is neither faulty nor has faulty wiring. If a garbled gate gg^i is faulty, fault free or has faulty wiring, we say the same of any shifted gate sgg^i resulting from shifting gg^i.

Gates gg^i with faulty wiring are problematic for the cut-and-choose test: If $i \in T$ A can choose to let gg^i act as a fault free gate by opening the wildcard commitments consistently with the actual zero-keys used to generate gg^i. On the other hand, if $i \in U$ A can make sgg^i faulty by opening the commitment inconsistently in **Soldering**[1].

In Lemma 2 we show that, with overwhelming probability, there will be a majority of fault free gates in $\left(gg^i\right)_{i\in\mathsf{Bucket}(j)}$ for all $j \in \mathsf{gates}$. It is easy to verify that this means that after **Horizontal Soldering** all commitments to zero-keys are fixed. I.e., the commitment to the zero-key of a faulty wire will be fixed to open as one specific value. If this value is not consistent with the zero-keys used to generate the associated garbled gate, then that gate becomes faulty.

Note however, that for all $j \in \mathsf{gates}$ all fault free shifted gates $\left(sgg^i\right)_{i\in\mathsf{Bucket}(j)}$ resulting from **Horizontal Soldering** will have identical input and output keys, as required of the garbled circuit, even if some gates in $\left(gg^i\right)_{i\in\mathsf{Bucket}(j)}$ had faulty wiring. I.e., the effect of a garbled gate gg^i having faulty wiring is *at worst* that shifted gate sgg^i after **Soldering** is faulty. Since we use a $\mathcal{F}_{\mathsf{COM}}$ functionality with at most k wildcard commitments we still have at most k faulty gates in **Evaluation**. Since these faulty gates are placed in random buckets we can still guarantee correctness with overwhelming probability.

Note that the faulty wires are also the reason for gate replication on the input layer, to not let A change her or B's input by using the wildcard commitments.

4 Commitments

In this section we present our novel construction of XOR-homomorphic commitment based solely on OT. To the best of our knowledge this is the first kind of homomorphic cryptographic primitive that can be obtained under general assumption. We also describe how to transform this general construction of XOR-homomorphic commitments into the specific type we need for the MiniLEGO protocol described inFig. 7.

4.1 The Ideal Functionality

We name our ideal functionality $\mathcal{F}_{\mathsf{WCOM}}$ and describe it in Fig. 8. The functionality allows A to commit to up to μ messages and to later reveal those messages. In addition the $\mathcal{F}_{\mathsf{WCOM}}$ allows to reveal the XOR of two or more committed messages to B (without revealing any extra information about the original committed messages). In the context of Fig. 7 this is the same as $\mathcal{F}_{\mathsf{COM}}$ except without the methods for **Oblivious-Opening**, **Or-Opening** and (which will become apparent later on) contains more wildcards than we can handle in the MiniLEGO protocol describedFig. 7.

We formalize $\mathcal{F}_{\mathsf{WCOM}}$ in the following way: First we let the adversary specify a set of identifiers ID with $|ID| = \mu$ used to identify each of the μ commitments (for technical reasons ID will be a subset of $[2\mu]$). In addition the adversary gives a set $W \subset ID$, of size at most κ, to identify the wildcard commitments. $\mathcal{F}_{\mathsf{WCOM}}$ stores a set of linear equations on μ variables $(\mathtt{X}_i)_{i\in ID}$ (one for each commitment). Initially this set is empty. Each time A commits to a message m_j using a non-wildcard commitment (i.e., $j \in ID\backslash W$) $\mathcal{F}_{\mathsf{WCOM}}$ will store the equation $\mathtt{X}_j = m_j$. For wildcard commitments no such equation are stored when A makes the commitment. If A instructs $\mathcal{F}_{\mathsf{WCOM}}$ to open the XOR of the set of commitments $J \subset ID$ we let corrupted A input a message $m_J \in \{0,1\}^t$ she wants to open to. The functionality then adds the equation $\bigoplus_{j\in J} \mathtt{X}_j = m_J$ to the set of stored equations and checks that this set of equations has a solution, i.e., if there is an assignment of values in $\{0,1\}^t$ to each variable $\mathtt{X}_j$ such that all equations are satisfied. If so, the functionality permanently stores the equation $\bigoplus_{j\in J} \mathtt{X}_j = m_J$ and opens the XOR of the set of commitments J as m_J. Otherwise, $\mathcal{F}_{\mathsf{WCOM}}$

[1] By *inconsistently* we mean inconsistent with the actual keys used for gg^i, not inconsistent with the equations stored in $\mathcal{F}_{\mathsf{COM}}$.

will output `Alice cheats` to B and terminate. Note that if $J \cap W = \emptyset$ then for all $j \in J$ the functionality has stored the equation $\mathtt{X}_j = m_j$, and therefore a corrupt A can only open the commitment successfully if $m_J = \bigoplus_{j \in J} m_j$. Note also that if, e.g., A has made commitments $i \in W$ and $j \in ID \setminus W$, and opens the XOR of commitments i and j as m' then for all later openings A can only successfully open the wildcard commitment i as $m'_i = m_j \oplus m'$. In these cases, when a commitment can only successfully be opened to one value, we say that the commitment is *fixed* to that value. Non-wildcard commitments are always fixed; when A opens the XOR of a wildcard commitments and non-wildcard commitments, a wildcard commitment can become fixed. When a wildcard commitment has been fixed it can essentially be viewed as a non-wildcard commitment.

Notice that the terminology can become a little confusing because of the wildcard commitments: When we say that A opens the XOR of some set of commitments $J \subset ID$ to a value m_J, then we cannot guarantee that $m_J = \bigoplus_{j \in J} m_j$, when $J \cap W \neq \emptyset$.

Setup

On input $(\mathtt{init}, ID, W)$ with $|ID| = \mu$, $|W| \leq \kappa$ and $W \subset ID$ from the adversary output ID to both parties and let $\mathcal{J} = \emptyset$. If A is honest, then $W = \emptyset$.

Commit

On input $(\mathtt{commit}, j \in ID, m_j)$ with $m_j \in \{0,1\}^t$ from A, and where no value of the form $(j, \cdot)$ is stored, store (j, m_j). If $j \in ID \setminus W$, add $J = \{j\}$ to $\mathcal{J}$ and associate with J the equation $\mathtt{X}_j = m_j$. Then output $(\mathtt{commit}, j)$ to B.

Open

On input $(\mathtt{open}, J \subset ID)$ from A, where for all $j \in J$ a pair (j, m_j) is stored do the following:

- If A is honest, output $(\mathtt{open}, J, \oplus_{j \in J} m_j)$ to B.
- If A is corrupted wait for A to input $(\mathtt{corrupt\text{-}open}, J, m_J)$. Then add J to $\mathcal{J}$, associate the equation $\oplus_{j \in J} \mathtt{X}_j = m_J$ to J, and check that the equation system $\{\oplus_{j \in J} \mathtt{X}_j = m_J\}_{J \in \mathcal{J}}$ has a solution[a]. If so, output $(\mathtt{open}, J, m_J)$ to B. Otherwise, output `Alice cheats` to B and terminate.

[a] I.e., there should be an assignment of values to the wildcard commitments such that all stored openings can be explained by this assignment.

Figure 8. The ideal functionality, $\mathcal{F}_{\mathsf{WCOM}}$, for our basic commitment scheme consisting.

4.2 Building blocks

Here we give the building blocks from which we implement $\mathcal{F}_{\mathsf{WCOM}}$.

Oblivious Transfer. We use a $\binom{n}{u}$-Oblivious Transfer functionality with message strings of length 2μ. We denote this functionality $\binom{n}{u}$-$\mathcal{F}_{\mathrm{OT}}(2\mu)$. On input `start` from both A and B the $\binom{n}{u}$-$\mathcal{F}_{\mathrm{OT}}(2\mu)$ functionality picks n message strings $S_1, \ldots, S_n \in_{\mathrm{R}} \{0,1\}^{2\mu}$, a uniformly random set $I \subseteq \{1, \ldots, n\}$ with $|I| = u$ and outputs $(S_i)_{i \in [n]}$ to A and $(I, (S_i)_{i \in I})$ to B. We can implement $\binom{n}{u}$-$\mathcal{F}_{\mathrm{OT}}(2\mu)$ for any $2\mu = \mathrm{poly}(k)$ using a $\mathcal{F}_{\mathrm{OT}}$ functionality and a pseudo random generator (prg), where k is the security parameter, using the same technique as in [NNOB12]: Simply use the $\mathcal{F}_{\mathrm{OT}}(k)$ functionality to send seeds to the prg and then use the prg to expand those seeds to 2μ bits. One can then construct a $\binom{n}{u}$-$\mathcal{F}_{\mathrm{OT}}(2\mu)$ functionality from $\binom{2}{1}\mathcal{F}_{\mathrm{OT}}(2\mu)$ e.g., as described in [NP99].

Error Correcting Codes. We also need an error correcting code (ECC), which encodes an t-bit string as an n-bit string with minimal distance at least d using some ϕ-bits of randomness. It should at the same time be a secret sharing scheme in that seeing u random positions of a random codeword does not leak information on the message. We denote this scheme by $\mathrm{ssecc}^{t,n,d,u}$. We use $\mathrm{enc}^{t,n,d,u}$ to denote the encoding function and we use $\mathrm{dec}^{t,n,d,u}$ to denote the decoding function. Both should be PPT and we drop parameters for notational convenience. The code should have the following properties:

Error correction For all $m \in \{0,1\}^t$, $r \in \{0,1\}^\phi$ and error vectors $e \in \{0,1\}^n$ with $\mathrm{hw}(e) < d/2$ it holds that $\mathrm{dec}(\mathrm{enc}(m;r) \oplus e) = m$, where hw is the Hamming weight in $\{0,1\}^n$. We assume that $\mathrm{dec}(C) = \bot$ when C has distance more than $d/2$ to all codewords and we assume that there exists an efficient algorithm ncw such that $\mathrm{ncw}(\mathrm{enc}(m;r) \oplus e) = \mathrm{enc}(m;r)$ when $\mathrm{hw}(e) < d/2$[2].

Privacy There exists a PPT function xpl which can explain any codeword as being a codeword of any message to anyone who knows at most u positions of the codeword. Formally, for all $I \subset [n]$, $|I| = u$ and all $m, m' \in \{0,1\}^t$ the distributions D_0 and D_1 described below are statistically close. The distribution D_0 is generated as follows: Sample $r \in_{\mathrm{R}} \{0,1\}^\phi$, let $c = \mathrm{enc}(m;r)$ and output $((c_i)_{i \in I}, m, r)$. The distribution D_1 is generated as follows: Sample $r' \in_{\mathrm{R}} \{0,1\}^\phi$, let $c = \mathrm{enc}(m';r')$, sample $r \leftarrow \mathrm{xpl}(I, m', r', m)$ and output $((c_i)_{i \in I}, m, r)$.

Linearity For all $m, m' \in \{0,1\}^t$ and $r, r' \in \{0,1\}^\phi$ it holds that $\mathrm{enc}(m;r) \oplus \mathrm{enc}(m';r') = \mathrm{enc}(m \oplus m'; r \oplus r')$.

Note that **Error correction** implies that the minimal distance is at least d, i.e., for all $m \neq m' \in \{0,1\}^t$ and $r, r' \in \{0,1\}^\phi$, $c = \mathrm{enc}(m;r)$ and $c' = \mathrm{enc}(m';r')$ it holds that $\mathrm{ham}(c, c') \geq d$ where ham is the Hamming distance.

We further require of the parameters of ssecc that: $n = \Theta(k)$, $u = \Theta(n)$ and $d = \Theta(n)$. I.e., both the privacy and minimum distance of ssecc must be a constant fraction of the length of codewords, and the code should have constant rate. Codes that satisfy the desired properties can be found in [CC06].

4.3 Protocol Specification

Here we describe the ideas behind the protocol π_{WCOM} (described in Fig. 9) implementing $\mathcal{F}_{\mathsf{WCOM}}$ (described in Fig. 8).

Let $v \in \{0,1\}^n$ and $I = \{i_1, i_2, \ldots, i_u\} \subseteq [n]$. We define the function $w_I : \{0,1\}^n \to \{0,1\}^u$ so that $w_I(v) = (v_{i_1}, v_{i_2}, \ldots, v_{i_u}) \in \{0,1\}^u$, i.e., $w_I(v)$ is the u-bit string consisting of the u bits in v indexed by I.

In the protocol a commitment to a message m is a one-time pad of m with some key T. Clearly this is hiding but not binding. To make the commitment binding we allow the receiver of the commitment (B) to learn $w_I(m)$ for some secret set $I \subsetneq [n]$. We denote $w_I(m)$ the *watch bits* of the commitment. To open the commitment to m A sends m' to B and B checks if $w_I(m') = w_I(m)$. If this is not the case B rejects the opening.

The watch bits give some degree of binding since A can only open the commitment to some message $m' \neq m$ if $w_I(m') = w_I(m)$. I.e., if $u = |I|$ is large enough A can only hope to change a few bits of m without getting caught. On the other hand the watch bits clearly compromises the hiding property of the commitment. To avoid this we use the code ssecc to encode the message m and commit to the encoded m instead. I.e., a commitment to m becomes $\mathrm{enc}(m;r) \oplus T$. By privacy of ssecc m is now hidden.

The encoding additionally strengthens the binding of the commitment: codewords c and c' encoding to two different messages m and m' must be different in many bit positions. Thus for A to open a commitment to m to m' none of these positions must be in the watch bits.

More precisely let $d = 2w + 1$ be the minimum distance of ssecc for some $w < \frac{n}{2}$ and suppose a corrupt A gives the commitment, $c \oplus T$. Note that when A is corrupt c does not have to be a codeword. In that case we have that $c = \mathrm{ncw}(c) \oplus e$ for some *error vector* $e \in \{0,1\}^n$, and we say the commitment has $\mathrm{hw}(e)$ *errors.*

Regardless of the number of errors, consider what it takes for A to be able to open this commitment to two different messages m' and m'', with codewords c' and c'' respectively: For any two different codewords c' and c'' one of them has distance at least w to c, say c'. In other words c' has at least w bit positions different from c. If A tries to open the commitment to m', B only accepts the opening if none of these bit positions

[2] Above and in the following we assume that ncw gives the closest codeword on all inputs, also those with distance more than $d/2$ to all codewords.

are in his u watch bits for the commitment. Thus for any commitment (possibly with errors) the probability that a cheating A can open the commitment to two different messages m' and m'' is at most

$$\begin{aligned}\binom{n-u}{w}\binom{n}{w}^{-1} &= \prod_{i=0}^{w-1}\frac{n-u-i}{s-i}\prod_{i=0}^{w-1}\frac{w-i}{n-i}\\ &= \prod_{i=0}^{w-1}\frac{n-u-i}{n-i}\\ &= \prod_{i=0}^{w-1} 1-\frac{u}{n-i}\ .\end{aligned}$$

Assume that $w = w'n$ and $u = u'n$ for constants $0 < u', w' < 1$, then the probability of A breaking the binding property is at most

$$\begin{aligned}\prod_{i=0}^{w-1}\left(1-\frac{u}{n-i}\right) &\leq \prod_{i=0}^{w-1}\left(1-\frac{u'n}{n}\right)\\ &= (1-u')^{w'n}\ .\end{aligned}$$

Thus with k being the security parameter, $n = \Theta(k)$ and for any positive constants u', w' with $0 < u', w' < 1$ A will have negligible probability of breaking binding.

Notice, that while c' will have distance at least w to c it could be that c'' is much closer to c. E.g., c'' could be the nearest codeword to c. In this case, by a similar calculation as the one above, we have that, if the commitment has *very few* errors, a cheating A *could* open the commitment to m'' with noticeable probability (say, if the number of errors were constant in k). This is not a problem since such a "slightly wrong" commitment can be seen as a commitment to the message m'' encoded by the nearest codeword c''.

To get XOR-homomorphic commitments, more work has to be done. The problem being that the XOR of several commitments with errors, may become a commitment that breaks binding, even if the individual commitments only have a few errors. Consider a number of commitments made with non-codewords c_i with nearest codewords c'_i. The XOR the non-codewords $c = \bigoplus_i c_i$ may then be very far from the XOR of their nearest codewords $c' = \bigoplus_i c'_i$. In fact c might be so far away from c' that it gets very close to some other codeword c''. Hence the XOR of the commitments can be opened to a message different from the XOR of the message associated with the individual commitments. This would break the binding property.

To deal with this problem, the protocol starts by letting A commit to 2μ random messages. We then do a cut-and-choose test to check that half of these commitments can be opened correctly. If A passes the test we have that, with overwhelming probability, the remaining commitments only have a few errors. Additionally, those errors must be isolated to a few common bit positions. Thus the result of XOR'ing these commitments will at most have a few a errors in the same positions.

Thus if A passes the cut-and-choose test we use the un-tested random commitments to implement the actual commitments. The resulting commitment will have exactly the same errors as the random commitment (if any).

Theorem 2. *Let k be the security parameter and use a code with $n = \Theta(k)$, $u = \Theta(n)$, $d = \Theta(n)$ and $k < d/2$ as, e.g., given by [CC06]. Then the protocol in Fig. 9 UC, active, static securely implements $\mathcal{F}_{\mathsf{WCOM}}$ in the $\left(\binom{n}{u}\text{-}\mathcal{F}_{\mathrm{OT}}(2\mu)\right)$-hybrid model when initialized on $(\boldsymbol{init}, ID, W)$ with $|ID| = \mu$ and $|W| \leq nk + k$.*

Analysis. Simulating when no party is corrupted or both parties are corrupted is straight forward. Simulating when B is corrupted is also quite simple, and can be done using standard techniques from simulation in secure multi-party computation based on secret sharing. Thus we will only sketch the proof for corrupted B, and focus the case of corrupted A. A sketch of the proof of security against a malicious A is included below and the formal proof can be found in Appendix B.

Setup

To set up the scheme A and B run the following.

1. A and B run a $\binom{n}{u}$-$\mathcal{F}_{\mathrm{OT}}(2\mu)$ functionality and get as output $(R_i)_{i\in[n]}$ and $(I, (R_i)_{i\in I})$ respectively where $R_1, \ldots, R_n \in_{\mathrm{R}} \{0,1\}^{2\mu}$ and I is a uniformly random subset of $[n]$ with $|I| = u$.
2. A lets $R \in \{0,1\}^{n\times 2\mu}$ be the matrix with R_i as the i'th row and lets $T_j \in \{0,1\}^n$ be the j'th column of R[a].
3. For $j = 1, \ldots, 2\mu$ A samples $x_j \in_{\mathrm{R}} \{0,1\}^t$, $r_j \in_{\mathrm{R}} \{0,1\}^\phi$ and sends the commitment $c_j = (\mathrm{enc}(x_j; r_j) \oplus T_j, j)$ to B. We let $c_j = (U_j, j)$ denote the value received by B.
4. B sends a uniformly random subset $C \subset [2\mu]$. This also defines $ID = \bar{C}$.
5. For $j \in C$, A opens c_j by sending $o_j = (x_j, r_j, j)$.
6. For $j \in C$, B receives (x'_j, r'_j, j) and checks that
$$w_I\left(\mathrm{enc}\left(x'_j; r'_j\right)\right) = w_I(U_j) \oplus w_I(T_j) \ ,$$
if not B terminates the protocol.

Commit

To commit to m_j for $j \in ID$ A sends (y_j, j) to B where y_j is the *correction value* $y_j = x_j \oplus m_j$.

Open

To open the XOR of commitments $J \subset ID$ the parties do the following.

1. For $j \in J$, let $c_j = (\mathrm{enc}(x_j; r_j) \oplus T_j, j)$ be the commitments sent in initialization and y_j the value sent during commitment. A computes the opening of $\bigoplus_{j\in J} m_j$ as $o_J = \left(\bigoplus_{j\in J} x_j, \bigoplus_{j\in J} r_j, J\right)$, and sends it to B.
2. If an opening of J was done previously, B uses the previous m_J, otherwise he proceeds as follows: Let $c_j = (U_j, j)$ be the commitments received during **Setup**. B accepts $o_J = (x_J, r_J, J)$ iff
$$w_I(\mathrm{enc}(x_J; r_J)) = w_I\left(\bigoplus_{j\in J} U_j\right) \oplus w_I\left(\bigoplus_{j\in J} T_j\right) \ ,$$
where
$$w_I\left(\bigoplus_{j\in J} U_j\right) = \bigoplus_{j\in J} w_I(U_j)$$
and
$$w_I\left(\bigoplus_{j\in J} T_j\right) = \bigoplus_{j\in J} w_I(T_j) \ .$$
If B accepts he outputs $x_J \oplus y_J$, where $y_J = \bigoplus_{j\in J} y_j$. Otherwise, B rejects the opening and terminates the protocol.

[a] Notice B can use $(R_i)_{i\in I}$ to compute $w_I(T_j)$ for all $j \in [2\mu]$.

Figure 9. The protocol π_{WCOM} implementing $\mathcal{F}_{\mathsf{WCOM}}$.

Corrupted B. The simulator commits to 0^t in all commitments. When asked to open such a commitment U_j to a given $m_j \in \{0,1\}^t$ it uses the efficient algorithm xpl to explain the commitment as $U_j = \mathrm{enc}(x_j; r_j) \oplus T_j$ for $x_j = y_j \oplus m_j$. The only non-trivial detail is that if the simulator is asked to open a commitment, where the value of the opening follows from previous openings (i.e., using some linear equation), it computes the opening as a linear combination of the previous simulated openings. As an example, if the simulator opened U_j as $U_j = \mathrm{enc}(x_j; r_j) \oplus T_j$ and opened U_i as $U_i = \mathrm{enc}(y_i; r_i) \oplus T_i$. Then it will open $U_j \oplus U_i$ as $U_j = \mathrm{enc}(x_j \oplus x_i; r_j \oplus r_i) \oplus T_j \oplus T_i$.

Corrupted A. Intuition of the proof when A is corrupted is that the cut-and-choose test will catch A if there are many indices i for which there exists a commitment that has an error in position i. This is because if the errors of the commitments are very spread out, with high probability, many of them will be in the watch bits positions. As mentioned above, this means that almost all errors must be isolated in a few positions. Therefore XOR's of commitments will also have errors only in these position, so the XOR's will also be close to their "correct" codeword. The formal proof is complicated by the fact that a few commitments with many errors, or errors outside isolated few positions, may pass the cut-and-choose. These commitments will be the wildcards. It can be shown that not even a commitment with many errors can be opened to two different values, as it would give a codeword encoding a non-zero value which is 0 in all the watch bits, which happens with negligible probability by the watch bits being random and the minimal distance high. This translates into it being impossible to make any combination of openings of linear equations yielding inconsistent outputs.

4.4 Completing construction

We now give a few simple modifications to $\mathcal{F}_{\mathsf{WCOM}}$ and π_{WCOM} that we will need for our concrete use in the MiniLEGO protocol described in Fig. 7. In particular we show the protocol π_{COM} which extends π_{WCOM} and implements the functionality $\mathcal{F}_{\mathsf{COM}}$ already defined in Fig. 5.

First, as we want to treat the commitments as an ideal functionality, we need to push into the ideal functionality two extra commands (in a way similar to the commit-and-prove functionality in [CLOS02]). The first one allows to perform *Oblivious-Opening* i.e., B can choose between two sets and learn the XOR of the committed messages in one of them, without learning anything about the other set. Additionally, we need a command to allow A to open the XOR of committed messages in one out of two sets to B without revealing which one. In other words, A proves that one of the two sets of committed messages XORs to the opened message, without revealing which one. We call this an *Or-Opening*. Note, that for technical reasons there can at most be ω Or-Openings an all Or-Openings must be done before the first Oblivious-Opening. The parameter ω can be any polynomial of the security parameter but it must be known when setting up the functionality.

Second, we need to strengthen the protocol to have only k wildcard commitments instead of $nk + k$ as the protocol $\mathcal{F}_{\mathsf{WCOM}}$.

Finally, note that we already described the $\mathcal{F}_{\mathsf{WCOM}}$ functionality with these commands added ($\mathcal{F}_{\mathsf{COM}}$) in Fig. 5 as we used it in a black box manner in the MiniLEGO protocol in Fig. 7.

Oblivious-Opening. The Oblivious-Opening can be implemented very easily in the $\mathcal{F}_{\mathrm{OT}}$-hybrid model from any protocol implementing $\mathcal{F}_{\mathsf{WCOM}}$ with a non-interactive opening. That is, any protocol that implements the opening part of $\mathcal{F}_{\mathsf{WCOM}}$ by sending a single message o from A to B: A inputs as messages to $\mathcal{F}_{\mathrm{OT}}$ the openings o_{J_0} and o_{J_1} corresponding to the sets of commitments J_0 and J_1. B inputs b to $\mathcal{F}_{\mathrm{OT}}$ and receives o_{J_b}, which he can then open in the regular way.

This solution has an inherent selective failure attack: A corrupt A could compute one opening honestly and the other in a way that is sure to make B reject the opening. This way A will either learn b or she will be caught. However, in our concrete use of $\mathcal{F}_{\mathsf{COM}}$ in Section 3 it turned out that we can live with this attack. Therefore, instead of trying to handle the attack we make it explicit in the $\mathcal{F}_{\mathsf{COM}}$ functionality that such attacks are allowed for a corrupted A.

Note also that the actual Oblivious-Opening described in Fig. 5 is of a form where B first inputs his bit b and then only later does A instruct $\mathcal{F}_{\mathsf{COM}}$ to open the XOR of J_b to B. This is needed for our concrete use of $\mathcal{F}_{\mathsf{COM}}$ in Section 3. Again this can be implement very simply: A One Time Pads (OTP) the openings before she inputs them to $\mathcal{F}_{\mathrm{OT}}$. Later when she wants to open to B she sends both OTP keys to B, which allows B to recover o_{J_b}.

We give a formal description of the implementation extended with the Oblivious-Opening in Fig. 10, and we prove the following Theorem in Appendix C.

Theorem 3. *Let k be the security parameter and use an ECC with $n = \Theta(k)$, $u = \Theta(n)$, $d = \Theta(n)$ and $k < d/2$ as, e.g., given by [CC06]. Then the protocol Fig. 10 UC, active, static securely implements the **Oblivious-Opening** command of $\mathcal{F}_{\mathsf{COM}}$ described in Fig. 5 in the $\left(\binom{n}{u}\text{-}\mathcal{F}_{\mathrm{OT}}(2\mu), \mathcal{F}_{\mathrm{OT}}\right)$-hybrid model.*

Setup

As in Fig. 9.

Commit

As in Fig. 9.

Open

As in Fig. 9.

Oblivious-Opening

To do an Oblivious-Opening with the sets of commitments J_0 and J_1 the parties does the following.

1. A computes openings o_{J_0} and o_{J_1} of the XOR of commitments in the sets J_0 and J_1 respectively as she would for regular openings described in step 1 of the **Open** phase of Fig. 9. She then samples $M_0 \in_{\mathrm{R}} \{0,1\}^{|o_{J_0}|}$ and $M_1 \in_{\mathrm{R}} \{0,1\}^{|o_{J_1}|}$ (where $|x|$ denotes the bit-length of x), and inputs to $(M_0 \oplus o_{J_0}, M_1 \oplus o_{J_1}) = (O_0, O_1)$ to $\mathcal{F}_{\mathrm{OT}}$.
2. B inputs b to $\mathcal{F}_{\mathrm{OT}}$ and receives $O_b = M_b \oplus o_{J_b}$.
3. Later A can open the XOR of commitments J_b by sending (M_0, M_1) to B. B computes the opening $o_{J_b} = O_b \oplus M_b$ and opens as a regular opening described in step 2 of the **Open** phase of Fig. 9.

Figure 10. The Oblivious Opening extension of protocol Fig. 9

Less Wildcards. We now show how our scheme can be strengthened to have k wildcard commitments instead of $nk + k$. A uses $\mathcal{F}_{\mathsf{WCOM}}$ to commit to 2μ random values x_i (instead of just μ values). Then we randomly pair all the commitments into μ pairs, open the XOR of each pair, and use the left element in each pair as the commitment of $\mathcal{F}_{\mathsf{WCOM}}$, sending just a correction value y_i as the actual commitment (similar to the protocol in Fig. 9). The idea being that if any wildcard commitment i is paired with a non-wildcard commitment j it will become fixed by opening the XOR of the pair. Therefore, the resulting commitments are only wildcard if they were in a pair where both commitments were wildcard commitments. If we take $\mu > (nk + k)^2$, then the probability that there are more than k such pairs of wildcard commitments is negligible. We describe the protocol formally in the **Setup**, **Commit** and **Open** phases of Fig. 11 and give a proof of the following Theorem in Appendix C.

Theorem 4. *Let k be the security parameter and $\mu' = \mu + 6k\omega > (nk + k)^2$. Then the protocol in Fig. 11 UC, active, static securely implements the **Setup**, **Commit**, **Open** and **Oblivious-Opening** commands of $\mathcal{F}_{\mathsf{COM}}$ in the ($\mathcal{F}_{\mathsf{WCOM}}$)-hybrid model when $\mathcal{F}_{\mathsf{COM}}$ is initialized as $(\textbf{\textit{init}}, ID, W)$ with $|ID| = \mu'$ and $|W| \leq k$ and $\mathcal{F}_{\mathsf{WCOM}}$ initialized as $(\textbf{\textit{init}}, ID', W')$ with $|ID'| = 2\mu'$ and $|W'| \leq nk + k$.*

Or-Opening. The Or-Opening can be implemented in a black box way given the $\mathcal{F}_{\mathsf{WCOM}}$ functionality (without the Or-Opening command): A sends $m = \oplus_{i \in J_a} m_i$ to B, and then proves that it is a correct value, i.e., that $m = \bigoplus_{i \in J_0} m_i \vee m = \bigoplus_{i \in J_1} m_i$. To do this, she makes two new commitments, to $m_{J_0} = \bigoplus_{j \in J_0} m_j$ respectively $m_{J_1} = \bigoplus_{j \in J_1} m_j$. She will do this by sampling a bit $p \in \{0,1\}$ and then use $\mathcal{F}_{\mathsf{WCOM}}$ to commit to values $m'_0 = m_{J_{0 \oplus p}}$ and $m'_1 = m_{J_{1 \oplus p}}$ without revealing p to B. Then B will challenge A with a bit c. If $c = 0$, A reveals p and uses $\mathcal{F}_{\mathsf{WCOM}}$ to open $(\oplus_{j \in J_0} m_j) \oplus m_{J_0}$ to 0 and $(\oplus_{j \in J_1} m_j) \oplus m_{J_1}$ to 0, proving that the messages m_{J_0} and m_{J_1} where computed correctly. If $c = 1$, A opens the commitment to $m'_{a \oplus p} = m$. If we disregard wildcard commitments, this is a zero-knowledge proof with soundness $\frac{1}{2}$. By repeating k times we get negligible soundness error.

However, if A commits to m'_0 and m'_1 using wildcard commitments she can open the commitments as she pleases, and soundness drops to 0. To deal with this, we let B randomly choose the indices used by A to commit to m'_0 and m'_1. Now with high probability there will be k repetitions where neither m'_0 nor m'_1 are committed to using wildcard commitments.

We formally describe the protocol used to implement $\mathcal{F}_{\mathsf{COM}}$ with the **Or-Opening** command in Fig. 11 and prove the following Theorem in Appendix C.

Theorem 5. *Let k be the security parameter, $\mu > 6\left(nk^2 + k^2\right)$ and $\mu' = 2(\mu + 6k\omega)$, where $\omega = poly(k)$ is the number of Or-Openings and k is the number of wildcard commitments. Then the protocol in Fig. 11*

UC, active, static securely implements $\mathcal{F}_{\mathsf{COM}}$ in the $(\mathcal{F}_{\mathsf{WCOM}})$-hybrid model when $\mathcal{F}_{\mathsf{COM}}$ is initialized as $(\boldsymbol{init}, ID, W)$ for $|ID| = \mu$ and $|W| \leq k$ and $\mathcal{F}_{\mathsf{WCOM}}$ as $(\boldsymbol{init}, ID', W')$ with $|ID'| = \mu'$ and $|W'| \leq nk + k$.

In conclusion, to implement the $\mathcal{F}_{\mathsf{COM}}$ functionality we first use $\binom{n}{u}$-$\mathcal{F}_{\mathrm{OT}}(2(2(\mu + 6k\omega)))$ to get $\mathcal{F}_{\mathsf{WCOM}}$ with $|ID| = 2(\mu + 6k\omega)$ and $|W| \leq nk+k$ (Theorem 2). We then use this to implement the Oblivious-Opening functionality (Theorem 3). From that we reduce the amount of wildcards from $nk + k$ to k at the expense of half the commitments (Theorem 4). Finally, we add the Or-Opening command (Theorem 5) and get the full $\mathcal{F}_{\mathsf{COM}}$ functionality for $|ID'| = \mu$ and $|W'| \leq k$. This is summarized in the following corollary:

Corollary 1. *Let k be the security parameter, $\mu > (nk + k)^2 > 6k^2$, $\omega = poly(k)$ and assume a code* ssecc *with $n = \Theta(k)$, $u = \Theta(n)$, $d = \Theta(n)$ and $k < d/2$ as, e.g., [CC06]. Then $\mathcal{F}_{\mathsf{COM}}$ is UC, active and static secure in the $\left(\binom{n}{u}\text{-}\mathcal{F}_{\mathrm{OT}}(2(2(\mu + 6k\omega)), \mathcal{F}_{\mathrm{OT}}\right)$-hybrid model when $|ID| = \mu$ and $|W| \leq k$.*

Setup

1. The adversary initializes a $\mathcal{F}_{\mathsf{WCOM}}$ functionality by calling $(\texttt{init}, ID', W')$ with $|ID'| = 2(\mu + 6k\omega)$ and $|W'| \leq nk + k$, which outputs ID' to both A and B.
2. For all $i \in [2(\mu + 6k\omega)]$ A samples $x_i \in_{\mathrm{R}} \{0,1\}^t$ and inputs $(\texttt{commit}, i, x_i)$ to $\mathcal{F}_{\mathsf{WCOM}}$.
3. After receiving $(\texttt{commit}, i)$ for all $i \in [2(\mu + 6k\omega)]$ from $\mathcal{F}_{\mathsf{WCOM}}$ B picks a uniformly random pairing π (a permutation $\pi : ID' \to ID'$ where $\forall i, \pi(\pi(i)) = i$) and sends it to A.
4. A sets $ID = \{i \in ID' | i < \pi(i)\}$ and then inputs $(\texttt{open}, \{i, \pi(i)\})$ to $\mathcal{F}_{\mathsf{WCOM}}$. She then outputs ID, with $|ID| = \mu + 6k\omega$, as the set of commitment id's.
5. On receiving $(\texttt{open}, \{i, \pi(i)\}, z_i = x_i \oplus x_{\pi(i)})$ for all $i \in ID$ B outputs ID. Additionally B sends to A ω uniformly random but disjoint sets $D_1, \ldots, D_\omega \subset ID$ so that each D_i has size $6k$. These id's are set aside only for use in the Or-Opening.
6. For the remainder of the protocol A is not allowed to use id's in $ID' \setminus ID$. If she does so B terminates the protocol.

Commit

To commit to m_j for $j \in ID$ A sends (y_j, j) to B where y_j is the *correction value* $y_j = x_j \oplus m_j$.

Open

To open the XOR of commitments $J \subset ID$ the parties do the following.

1. For $j \in J$, let $c_j = (\mathrm{enc}(x_j; r_j) \oplus T_j, j)$ be the commitments sent in initialization and y_j the value sent during commitment. A computes the opening of $\bigoplus_{j \in J} m_j$ as $o_J = \left(\bigoplus_{j \in J} x_j, \bigoplus_{j \in J} r_j, J\right)$, and sends it to B.
2. If an opening of J was done previously, B uses the previous m_J, otherwise he proceeds as follows: Let $c_j = (U_j, j)$ be the commitments received during **Setup**. B accepts $o_J = (x_J, r_J, J)$ iff $w_I(\mathrm{enc}(x_J; r_J)) = w_I\left(\bigoplus_{j \in J} U_j\right) \oplus w_I\left(\bigoplus_{j \in J} T_j\right)$, where $w_I\left(\bigoplus_{j \in J} U_j\right) = \bigoplus_{j \in J} w_I(U_j)$ and $w_I\left(\bigoplus_{j \in J} T_j\right) = \bigoplus_{j \in J} w_I(T_j)$. If B accepts he outputs $m_J = x_J \oplus y_J$, where $y_J = \bigoplus_{j \in J} y_j$. Otherwise, B rejects the opening and terminates the protocol.

Oblivious-Opening

To do an Oblivious-Opening with the sets of commitments J_0 and J_1 the parties does the following:

1. A computes openings o_{J_0} and o_{J_1} of the XOR of commitments in the sets J_0 and J_1 respectively as she would for regular openings described in **Open** step 1. She then samples $M_0 \in_{\mathrm{R}} \{0,1\}^{|o_{J_0}|}$ and $M_1 \in_{\mathrm{R}} \{0,1\}^{|o_{J_1}|}$ (where $|x|$ denotes the bit-length of x), and inputs to $(M_0 \oplus o_{J_0}, M_1 \oplus o_{J_1}) = (O_0, O_1)$ to $\mathcal{F}_{\mathrm{OT}}$.
2. B inputs b to $\mathcal{F}_{\mathrm{OT}}$ and receives $O_b = M_b \oplus o_{J_b}$.
3. Later A can open the XOR of commitments J_b by sending (M_0, M_1) to B. B computes the opening $o_{J_b} = O_b \oplus M_b$ and opens as a regular opening described in **Open** step 2.

Or-Opening

To perform the d'th Or-Opening the parties do the following (recall that no Or-Openings are done after the first Oblivious-Opening):

1. A computes $m_{J_0} = \oplus_{j \in J_0} m_j$ and $m_{J_1} = \oplus_{j \in J_1} m_j$ and sends $(J_0, J_1, m = m_{J_a})$ to B.
2. B parses D_d as $D_d = \{j_0^1, j_1^1, \ldots j_0^{3k}, j_1^{3k}\} \subset ID$.
3. For each $i \in [3k]$ A samples $p^i \in_{\mathrm{R}} \{0,1\}$ and inputs $\left(\texttt{commit}, j_0^i, m_{J_{0 \oplus p^i}}\right)$ and $\left(\texttt{commit}, j_1^i, m_{J_{1 \oplus p^i}}\right)$ to $\mathcal{F}_{\mathsf{WCOM}}$.
4. After receiving $\left(\texttt{commit}, j_0^i\right)$ and $\left(\texttt{commit}, j_1^i\right)$ from $\mathcal{F}_{\mathsf{WCOM}}$ for all $i \in [3k]$, B sends $\left(c^i\right)_{i \in [3k]} \in_{\mathrm{R}} \{0,1\}^{3k}$ to A.
5. For each $i \in [3k]$ A does the following:
   - If $c^i = 0$: A inputs $\left(\texttt{open}, J_{0 \oplus p^i} \cup \{j_0^i\}\right)$ and $\left(\texttt{open}, J_{1 \oplus p^i} \cup \{j_1^i\}\right)$ to $\mathcal{F}_{\mathsf{WCOM}}$ and sends p^i to B.
   - If $c^i = 1$: A inputs $\left(\texttt{open}, \{j_{a \oplus p^i}^i\}\right)$ to $\mathcal{F}_{\mathsf{WCOM}}$.
6. For each $i \in [3k]$ B does the following:
   - If $c^i = 0$: B checks that he receives both $\left(\texttt{open}, J_{0 \oplus p^i} \cup \{j_0^i\}, 0\right)$ and $\left(\texttt{open}, J_{1 \oplus p^i} \cup \{j_1^i\}, 0\right)$ from $\mathcal{F}_{\mathsf{WCOM}}$.
   - If $c^i = 1$: B checks that he receives $\left(\texttt{open}, \{j^i\}, m\right)$ from $\mathcal{F}_{\mathsf{WCOM}}$ for some $j^i \in \left\{j_0^i, j_1^i\right\}$.

   If any of the checks fail B rejects the opening and terminates the protocol. Otherwise, B outputs $(\texttt{open}, J_0, J_1, m)$.

Figure 11. The Protocol π_{COM} implementing $\mathcal{F}_{\mathsf{COM}}$.

References

[BDNP08] Assaf Ben-David, Noam Nisan, and Benny Pinkas. FairplayMP: a system for secure multi-party computation. In *ACM Conference on Computer and Communications Security*, pages 257–266, 2008.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *ACM Conference on Computer and Communications Security*, pages 784–796, 2012.

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

[CKKZ12] Seung Geol Choi, Jonathan Katz, Ranjit Kumaresan, and Hong-Sheng Zhou. On the security of the "free-XOR" technique. In *TCC*, pages 39–53, 2012.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *STOC*, pages 494–503, 2002.

[DKL$^+$12] Ivan Damgård, Marcel Keller, Enrique Larraia, Christian Miles, and Nigel P. Smart. Implementing AES via an actively/covertly secure dishonest-majority mpc protocol. In *SCN*, pages 241–263, 2012.

[FN13] Tore Kasper Frederiksen and Jesper Buus Nielsen. Fast and maliciously secure two-party computation using the gpu. Cryptology ePrint Archive, Report 2013/046, 2013. `http://eprint.iacr.org/`.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004.

[Gol09] Oded Goldreich. Comments on eight TCC'09 talks. Manuscript, 2009. `http://www.wisdom.weizmann.ac.il/~oded/X/tcc09.ps`.

[HEKM11] Yan Huang, David Evans, Jonathan Katz, and Lior Malka. Faster secure two-party computation using garbled circuits. In *USENIX Security Symposium*, 2011.

[HIKN08] Danny Harnik, Yuval Ishai, Eyal Kushilevitz, and Jesper Buus Nielsen. OT-combiners via secure computation. In *TCC*, pages 393–411, 2008.

[HKE12] Yan Huang, Jonathan Katz, and David Evans. Quid-pro-quo-tocols: Strengthening semi-honest protocols with dual execution. In *IEEE Symposium on Security and Privacy*, pages 272–284, 2012.

[HKS$^+$10] Wilko Henecka, Stefan Kögl, Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. TASTY: tool for automating secure two-party computations. In *ACM Conference on Computer and Communications Security*, pages 451–462, 2010.

[HL10] Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols: Techniques and Constructions*. Springer-Verlag, 2010.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *CRYPTO*, pages 145–161, 2003.

[IKOS08] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In *STOC*, pages 433–442, 2008.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO*, pages 572–591, 2008.

[KS08] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In *ICALP (2)*, pages 486–498, 2008.

[KsS12] Benjamin Kreuter, shelat abhi, and Chih-hao Shen. Billion-gate secure computation with malicious adversaries. USENIX Security. Available at Cryptology ePrint Archive, Report 2012/179, 2012. `http://eprint.iacr.org/`.

[LOP11] Yehuda Lindell, Eli Oxman, and Benny Pinkas. The IPS compiler: Optimizations, variants and concrete efficiency. In *CRYPTO*, pages 259–276, 2011.

[LP07] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *EUROCRYPT*, pages 52–78, 2007.

[LP09] Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *J. Cryptology*, 22(2):161–188, 2009.

[LP11] Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. In *TCC*, pages 329–346, 2011.

[LPS08] Yehuda Lindell, Benny Pinkas, and Nigel P. Smart. Implementing two-party computation efficiently with security against malicious adversaries. In *SCN*, pages 2–20, 2008.

[MNPS04] Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay - secure two-party computation system. In *USENIX Security Symposium*, pages 287–302, 2004.

[NNOB12] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In *CRYPTO*, pages 681–700, 2012.

[NO09] Jesper Buus Nielsen and Claudio Orlandi. LEGO for two-party secure computation. In *TCC*, pages 368–386, 2009.

[NP99] Moni Naor and Benny Pinkas. Oblivious transfer with adaptive queries. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 573–590. Springer, 1999.

[NPS99] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *ACM Conference on Electronic Commerce*, pages 129–139, 1999.

[Orl11] C. Orlandi. *Secure Computation in Untrusted Environments*. Department of Computer Science, Aarhus University, 2011.

[PSSW09] Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure two-party computation is practical. In *ASIACRYPT*, pages 250–267, 2009.

[sS11] shelat abhi and Chih-Hao Shen. Two-output secure computation with malicious adversaries. In *EUROCRYPT*, pages 386–405, 2011.

[ST04] Berry Schoenmakers and Pim Tuyls. Practical two-party computation based on the conditional gate. In *ASIACRYPT*, pages 119–136, 2004.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.

A Proof of Theorem 1

To do this proof we first show Lemma 1, which shows that the theorem is true for a malicious B. We then prove Lemma 2 which is needed in order to show security against a malicious A. We finish by proving Lemma 3. From these proofs the theorem follows directly.

Lemma 1. *π_{LEGO} is a secure implementation of $\mathcal{F}_{\mathsf{SFE}}$ against a malicious B^*.*

Proof (Lemma 1).

We present a simulator S that given access to $\mathcal{F}_{\mathsf{SFE}}$ simulates the real world view of the environment Z when Z corrupts B^*.

At the beginning of the simulation Z inputs $(\mathbf{a}, \mathcal{C}_A, k)$ and $(\mathbf{b}, \mathcal{C}_B, k)$ to A and S respectively, and A inputs $(\mathtt{init}, (\mathbf{a}, \mathcal{C}_A), k)$ to $\mathcal{F}_{\mathsf{SFE}}$. The simulator receives $(\ell, \mathcal{C}')$ from $\mathcal{F}_{\mathsf{SFE}}$ and then runs π_{LEGO} completely as an honest A with input $\left(0^\ell, \mathcal{C}'\right)$ except that the simulator fully controls the $\mathcal{F}_{\mathsf{COM}}$ functionality. Thus S reads the input b^* used by B^* as it is given to the $\mathcal{F}_{\mathsf{COM}}$ functionality in step 6 of **Setup**, and S can extract the challenges chosen by B^* in step 5 of **Setup**. To conclude the simulation, if B^* does not cause the protocol to abort, S inputs $(\mathtt{init}, (\mathbf{b}^*, \mathcal{C}_B), k)$ to $\mathcal{F}_{\mathsf{SFE}}$ on behalf of B^* and $\mathcal{F}_{\mathsf{SFE}}$ outputs $\mathcal{C}'(\mathbf{a}, \mathbf{b}^*)$ to A. If B^* causes S to abort, S makes $\mathcal{F}_{\mathsf{SFE}}$ abort.

To argue indistinguishability of the Z's view in the real and ideal world we reduce to the security of garbling scheme $(\mathsf{Yao}, \mathsf{Eval})$. Thus we consider an adversary $\mathcal{A}$ in the $\mathsf{IND\text{-}CIA}$ game as defined in Def. 1 that makes use of Z and B^*. We will construct $\mathcal{A}$ so that, depending on the value of the challenge c chosen by the $\mathsf{IND\text{-}CIA}$ game, $\mathcal{A}$ produces a view for Z that is either indistinguishable from the real world view or the ideal world simulation of S.

$\mathcal{A}$ first runs Z to get the inputs $A = (\mathbf{a}, \mathcal{C}_A)$ for A. When B^* in step 5 and 6 of **Setup** inputs T, $(u_i, v_i)_{i \in T}$, $\mathsf{BucketOf}$ and $\mathbf{b}^*$ $\mathcal{A}$ reads these values.

Now $\mathcal{A}$ uses the $\mathsf{IND\text{-}CIA}$ game to generate the garbled gates to hand to B^*. $\mathcal{A}$ constructs its two strings to $\mathsf{IND\text{-}CIA}$ so that $\mathcal{A}$ receives keys to evaluate the garbled circuit, as specified by T and $\mathsf{BucketOf}$, on inputs $(\mathbf{a}, \mathbf{b}^*)$ for $c = 0$, and on inputs $\left(0^\ell, \mathbf{b}^*\right)$ for $c = 1$. Additionally $\mathcal{A}$ will construct the strings so that for either value of c, $\mathcal{A}$ receives the keys needed to evaluate the gates in the cut-and-choose challenge.

To this end $\mathcal{A}$ evaluates $\mathcal{C}'(\mathbf{a}, \mathbf{b}^*)$ and for each gate $j \in \mathsf{gates}$ records the left and right inputs l^j_0 and r^j_0 respectively. Similarly $\mathcal{A}$ evaluates $\mathcal{C}'\left(0^\ell, \mathbf{b}^*\right)$ and for each gate $j \in \mathsf{gates}$ records the left and right inputs l^j_1 and r^j_1. Then $\mathcal{A}$ outputs the strings $\left(\hat{a}^1_0, \ldots, \hat{a}^\Gamma_0, \hat{b}^1_0, \ldots, \hat{b}^\Gamma_0\right)$ and $\left(\hat{a}^1_1, \ldots, \hat{a}^\Gamma_1, \hat{b}^1_1, \ldots, \hat{b}^\Gamma_1\right)$ where

- For each $i \in T$ $\mathcal{A}$ lets $\left(\hat{a}^i_0, \hat{b}^i_0\right) = \left(\hat{a}^i_1, \hat{b}^i_1\right) = (u_i, v_i)$.
- For each $j \in \mathsf{gates}$ $\mathcal{A}$ lets $\hat{a}^i_0 = l^j_0$ and $\hat{a}^i_1 = l^j_1$ for all $i \in \mathsf{Bucket}(j)$.
- For each $j \in \mathsf{gates}$ $\mathcal{A}$ lets $\hat{b}^i_0 = r^j_0$ and $\hat{b}^i_1 = r^j_1$ for all $i \in \mathsf{Bucket}(j)$.

Given these strings the $\mathsf{IND\text{-}CIA}$ game inputs $\left(gg^i, L^i_{\hat{a}^i_c}, R^i_{\hat{b}^i_c}\right)_{i \in [\Gamma]}$ to $\mathcal{A}$, and $\mathcal{A}$ hands $\left(gg^i\right)_{i \in [\Gamma]}$ to B^*.

$\mathcal{A}$ then computes the output keys $O^i \leftarrow \mathsf{Eval}\left(gg^i, L^i_{\hat{a}^i_c}, R^i_{\hat{b}^i_c}\right)$ for each $i \in [\Gamma]$. Notice that the keys $\left(L^i_{\hat{a}^i_c}, R^i_{\hat{b}^i_c}, O^i\right)_{i \in [\Gamma]}$ are enough for $\mathcal{A}$ to compute the correct value of all the strings she needs to open in **Cut-and-choose** and **Soldering**: For **Cut-and-choose** $\mathcal{A}$ knows $L^i_{u_i} = L^i_{\hat{a}^i_c}$, $R^i_{v_i} = R^i_{\hat{b}^i_c}$ and $O^i = O^i_{u_i \wedge v_i}$. For **Soldering** consider the difference L^{h^d} corresponding to a gate $j \in \mathsf{gates}$ with $h = \mathsf{BucketHead}(j)$. $\mathcal{A}$ can compute L^{h^d} as

$$\begin{aligned}
L^{h^d} &= L^h_{\hat{a}^h_c} \oplus O^{\mathsf{BucketHead}(\mathsf{lp}(j))} \\
&= \left(L^h_0 \oplus \hat{a}^h_c \Delta\right) \oplus \left(O^{\mathsf{BucketHead}(\mathsf{lp}(j))}_0 \oplus \hat{a}^h_c \Delta\right) \\
&= L^h_0 \oplus O^{\mathsf{BucketHead}(\mathsf{lp}(j))}_0
\end{aligned}$$

where the second equality is by the correctness of $(\mathsf{Yao}, \mathsf{Eval})$ and definition of $\hat{a}_c^h$ and $\hat{b}_c^h$. Similarly $\mathcal{A}$ can compute all the other differences needed in **Soldering**.

Thus $\mathcal{A}$ can simulate **Cut-and-choose** and **Soldering** as it fully controls the $\mathcal{F}_{\mathsf{COM}}$ functionality.

Similarly $\mathcal{A}$ can simulate **Input** by simply sending the input keys given by the IND-CIA game to B^* on behalf of the $\mathcal{F}_{\mathsf{COM}}$ functionality.

In **Evaluation** $\mathcal{A}$ receives $\hat{O}^j$ for each $j \in \mathsf{outputGates}$ from B^*. Now since $\mathcal{A}$ has all the same keys for the garbled circuit as B^*, she can compute the keys O^j that an honest B would have sent. Thus A aborts if $\hat{O}^j \neq O^j$. Otherwise, $\mathcal{A}$ outputs $\mathcal{C}'(\mathbf{a}, \mathbf{b}^*)$ to Z on behalf of A. $\mathcal{A}$ then outputs whatever Z outputs.

Denote by F the event that B^* outputs a $\hat{O}^j \neq O^j$ so that $\hat{O}^j \oplus O^j = \varDelta$, and assume F does not occur. Then by definition of the IND-CIA game and the way $\mathcal{A}$ constructs the strings $\left(\hat{a}_0^1, \ldots, \hat{a}_0^\Gamma, \hat{b}_0^1, \ldots, \hat{b}_0^\Gamma\right)$ and $\left(\hat{a}_1^1, \ldots, \hat{a}_1^\Gamma, \hat{b}_1^1, \ldots, \hat{b}_1^\Gamma\right)$ it is easy to verify that when $c = 0$ the view produced by $\mathcal{A}$ towards Z is perfectly indistinguishable to the real world view, while for $c = 1$ the view is perfectly indistinguishable from the ideal world simulation of S. Thus any advantage of Z in distinguishing the real from ideal world directly translates into advantage of $\mathcal{A}$ in the IND-CIA game. Thus assuming F does not occur and $(\mathsf{Yao}, \mathsf{Eval})$ is secure the ideal and real worlds are indistinguishable.

Now assume F does occur. In this case $\mathcal{A}$ aborts the protocol whereas S and honest A does not. To handle this we argue that F only occurs with negligible probability. In fact, if F occurs and B^* outputs a $\hat{O}^j \neq O^j$ then $\mathcal{A}$ can use $\varDelta' = \hat{O}^j \oplus O^j$ to get all keys for some garbled gate and therefore distinguish. By assumption that F does occur this allows $\mathcal{A}$ to win the IND-CIA game with noticeable probability. □

Before we show security for corrupted A, we show in Lemma 2 that if the protocol does not abort in **Cut-and-choose** then for the remainder of the protocol there will be more than $\lfloor \rho/2 \rfloor$ fault free garbled gates in each of the buckets with overwhelming probability. We note that in [Orl11] a similar lemma is proved, however, there they did not have the wildcard commitments to worry about.

Lemma 2. *Consider the protocol* π_{LEGO} *with honest* B. *Let* k *be the statistical security parameter and* $\rho = O(k/\log(s))$ *and assume* A *in* ***Garbling*** *prepares* f *faulty gates and* v *gates with faulty wiring. Let*

- E_1 *be the event that the protocol does not abort in* ***Cut-and-choose****.*
- E_2 *be the event that, before* ***Soldering****, there exists a* $j \in \mathsf{gates}$ *so that* $(gg^i)_{i \in \mathsf{Bucket}(j)}$ *does not have more than* $\lfloor \rho/2 \rfloor$ *fault free gates.*

Then $\Pr(E_1 \wedge E_2)$ *is at most*

$$s \left(\frac{k+f}{s\rho}\right)^{\lceil \rho/2 \rceil} 2^{\rho - (f/6+1)} \leq 2^{-k}$$

Proof (Lemma 2).

Let $F, V \subset [\Gamma]$ be the sets of indices of all garbled gates that are faulty or have faulty wiring respectively and let $G = F \cup V$. Note that by the definition of faulty gates and faulty wiring $F \cap V = \emptyset$. Let $b = |G| = f + v$, $r_i = |\mathsf{Bucket}(i) \cup G|$ and $R = |\{i \in [s] | r_i > \rho/2\}$ (Here we assume wlog ρ to be odd).

Then for each garbled gate in $\{gg^i\}_{i \in F}$ we have that $i \in T$ with probability 1/2, and that if $i \in T$ then gg^i will be detected as being faulty with probability at least 1/4. I.e., we have

$$\Pr(E_1) \leq \left(\frac{7}{8}\right)^f = \left(\frac{8}{7}\right)^{-f} \leq 2^{-f/6} \ .$$

To bound $\Pr(E_2) \leq \Pr(R > 0)$, first notice that $\Pr(r_i > \rho/2)$ is the same for any $i \in [s]$. Thus by union bound we have

$$\Pr(R > 0) \leq s \cdot \Pr(r_1 > \rho/2) \ . \tag{1}$$

The event $r_1 > \rho/2$ we can describe in terms of the following experiment: from a collection of ρs balls were b balls are red and $\rho s - b$ balls are green pick at random ρ balls. The probability that we pick q red balls in this way is exactly $\Pr(r_1 = q)$. I.e., r_1 follows a hyper geometric distribution and we have

$$\Pr(r_1 = q) = \binom{b}{q}\binom{s\rho - b}{\rho - q}\binom{s\rho}{\rho}^{-1} .$$

Note that we can assume $b \geq q$ or this probability is clearly 0. Writing out the binomial coefficients we get

$$\Pr(r_1 = q) = \prod_{i=0}^{q-1} \frac{b-i}{q-i} \prod_{i=0}^{\rho-q-1} \frac{s\rho - b - i}{\rho - q - i} \prod_{i=0}^{\rho-1} \frac{\rho - i}{s\rho - i} . \tag{2}$$

Assume $\rho \geq q > \rho/2$ and focus on the last two products of (2). Then we have

$$\begin{aligned}\prod_{i=0}^{\rho-q-1} \frac{s\rho - b - i}{\rho - q - i} \prod_{i=0}^{\rho-1} \frac{\rho - i}{s\rho - i} &= \prod_{i=0}^{\rho-q-1} \frac{s\rho - b - i}{\rho - q - i} \prod_{i=0}^{\rho-q-1} \frac{\rho - q - i}{s\rho - q - i} \prod_{i=0}^{q-1} \frac{\rho - i}{s\rho - i} \\ &\leq \prod_{i=0}^{q-1} \frac{\rho - i}{s\rho - i} ,\end{aligned}$$

where the inequality follows from $b \geq q$. If we plug this into (2) we get

$$\begin{aligned}\Pr(r_1 = q) &\leq \prod_{i=0}^{q-1} \frac{b-i}{q-i} \prod_{i=0}^{q-1} \frac{\rho - i}{s\rho - i} \\ &= \prod_{i=0}^{q-1} \frac{\rho-i}{q-i} \prod_{i=0}^{q-1} \frac{b - i}{s\rho - i} \\ &= \binom{\rho}{q} \prod_{i=0}^{q-1} \frac{b - i}{s\rho - i} \\ &\leq \binom{\rho}{q} \left(\frac{b}{s\rho}\right)^q ,\end{aligned}$$

Plugging this into (1) we have

$$\begin{aligned}\Pr(R > 0) &\leq s \Pr(r_1 > \rho/2) \\ &\leq s \sum_{q=\lceil \rho/2 \rceil}^{\rho} \binom{\rho}{q} \left(\frac{b}{s\rho}\right)^q \\ &\leq s \left(\frac{b}{s\rho}\right)^{\lceil \rho/2 \rceil} \sum_{q=\lceil \rho/2 \rceil}^{\rho} \binom{\rho}{q} \\ &= s \left(\frac{b}{s\rho}\right)^{\lceil \rho/2 \rceil} 2^{\rho-1} .\end{aligned}$$

Now since $v \leq k$ by definition of the $\mathcal{F}_{\mathsf{COM}}$ functionality, we have as stated in the lemma

$$\Pr(E_1 \wedge E_2) \leq s\left(\frac{f+k}{s\rho}\right)^{\lceil \rho/2 \rceil} 2^{\rho-1}2^{-f/6}\ .$$

It is easy to verify (simply by finding the derivative) that this is maximized for $f = 6\lceil \rho/2 \rceil / \ln(2) - k$. Thus we have

$$\begin{aligned}\Pr(E_1 \wedge E_2) &\leq s\left(\frac{6\lceil \rho/2 \rceil/\ln(2)}{s\rho}\right)^{\lceil \rho/2 \rceil} 2^{\rho-1-\lceil \rho/2 \rceil/\ln(2)+k/6} \\ &\leq s\left(5/s\right)^{\lceil \rho/2 \rceil} 2^{\rho-2\rho/3+k/6} \\ &= 2^{-\log(s)(\rho-1)/2+\log(5)(\rho+1)/2+\rho/3+k/6} \\ &\leq 2^{-\log(s)\rho/3+7\rho/3+2+k/6} \\ &\leq 2^{\rho(7-\log(s))/3+2+k/6},\end{aligned}$$

where the third inequality is true since $(\rho-1)/2 \geq \rho/3$ which follows from the assumption on ρ. From $\rho = O(k/\log(s))$ it follows that $\Pr(E_1 \wedge E_2) \leq 2^{-k}$. □

Lemma 3. *π_{LEGO} s a secure implementation of $\mathcal{F}_{\mathsf{SFE}}$ against a malicious* A.

Proof (Lemma 3). We present a simulator S that given access to $\mathcal{F}_{\mathsf{SFE}}$ simulates the real world view of the environment Z when Z corrupts A^*.

At the beginning of the simulation Z inputs $(\texttt{init}, (\mathbf{b}, \mathcal{C}_B), k)$ to B who inputs it to $\mathcal{F}_{\mathsf{SFE}}$. So S receives $\mathcal{C}_B = (\ell, \mathcal{C}')$ from $\mathcal{F}_{\mathsf{SFE}}$ and then runs π_{LEGO} as an honest B with input $B = \left(0^\ell, \mathcal{C}_B\right)$ towards A^*.

In step 2 of **Input** S cannot immediately simulate the $\mathcal{F}_{\mathsf{COM}}$ functionality as it does not know the input $\mathbf{b}$ of the honest B. I.e., it does not know if the $\mathcal{F}_{\mathsf{COM}}$ would output `Alice cheats` making the real world protocol abort. However, S can check for each $j \in \mathsf{Binputs}$ if there is a value of $b_j \in \{0,1\}$ that *would* make $\mathcal{F}_{\mathsf{WCOM}}$ output `Alice cheats`. For each j where such a value of b_j exists denote that value $\hat{b}_j$ (if more than one such value exists S can safely abort the protocol) and let ζ be set of all such j's. Then S sets $\beta_j = 1 - b'_j$ and inputs $B' = (j, \beta_j)_{j \in \zeta}$ to $\mathcal{F}_{\mathsf{SFE}}$. Note, that this perfectly simulates the selective failure attack that A^* might do.

Since S fully controls $\mathcal{F}_{\mathsf{COM}}$ through out the protocol, S can extract any information A^* inputs to $\mathcal{F}_{\mathsf{COM}}$. This includes all committed keys, the indicies of wildcard commitments W and A^*'s input $\mathbf{a}^*$.

Also, S can extract the value $\varDelta$ from the commitment done in step 2 **Setup**. The only problem is that if the commitment to $\varDelta$ is a wildcard commitment, S cannot be sure that A^* will use this value of $\varDelta$ later in the protocol. However, even if the commitment to $\varDelta$ is a wildcard commitment it will become fixed as soon as it is opened XOR a non-wildcard commitment. This happens in **Cut-and-choose** if for some $i \in T$ $u_i = 1$ and $\mathsf{id}(L^i) \notin W$ (or $v_i = 1$ and $\mathsf{id}(R^i) \notin W$ respectively). Say that this does occurs for some $i \in T$ and A^* opens the XOR of the two commitments to K, then since S can extract the value L^i it can also compute $\varDelta = L^i \oplus K$ which is the value that the commitment to $\varDelta$ is now fixed to open to by $\mathcal{F}_{\mathsf{COM}}$.

The probability that $\varDelta$ does not become fixed in this way for some $i \in T$ is negligible. To see this consider that there is at most κ wildcard commitments but $|T| = \rho s$, thus the since the challenges u_i and v_i are sampled uniformly at random, the probability that $\varDelta$ is not fixed is at most $2^{\kappa - s\rho}$. By assumption that $s \geq \kappa^4$ this probability is negligible. I.e., regardless of whether or not A^* commits to $\varDelta$ using a wildcard commitment, S learns $\varDelta$ with overwhelming probability.

If A^* did not make the protocol abort, S inputs $\mathbf{a}^*$ to $\mathcal{F}_{\mathsf{SFE}}$ and learns the output $\mathbf{z}$.

By Lemma 2 and the discussion above we have that, with overwhelming probability, all buckets have a majority of fault free shifted gates that have the same input and output keys. S can now, for all $j \in \mathsf{outputGates}$, compute keys corresponding to the actual outputs, i.e., $O^j_{z_j} = O^j_0 \oplus z_j \varDelta$, where O^j_0 is the output zero-key of the fault free gates in $\mathsf{Bucket}(j)$ (which where extracted earlier).

Now we argue that the view of A^* is statistically indistinguishable (in the ($\mathcal{F}_{\mathsf{COM}}$)-hybrid model) when playing with a real B and when playing with the simulator S. This is by construction: Note that in the protocol the only information that travels from B to A^* is

1. in step 1 of **Cut-and-choose**, when B opens the commitment Com and the simulator does this as an honest B would do.
2. B's reaction to the selective failure attack A^* may use in step 2 of **Input**, but as we have argued above, this is simulated perfectly.
3. In step 3 of **Evaluate**, when B reveals the output keys to A.

So as long as the simulator S sends the correct output keys to A^*, the views are identical.

It follows from the construction that the simulator S will send the wrong keys only if output of a real execution of the protocol and the output of the ideal functionality are different.

Those outputs are identical if all buckets have a majority of fault free gates (this is by construction), and by Lemma 2 this happens with overwhelming probability.

□

B Proof of Theorem 2

In this section we formally prove the security of our homomorphic commitment scheme.

Proof. We first describe how to simulate the setup phase against a corrupt A^*. The simulator simulates the ideal functionality $\binom{n}{u}$-$\mathcal{F}_{\mathrm{OT}}(2\mu)$ to the adversarial A^* and records the outputs $(R_i)_{i\in[n]}$ and $(I,(R_i)_{i\in I})$. The simulator lets $R \in \{0,1\}^{n\times 2\mu}$ be the matrix with R_i as the i'th row and lets $T_j \in \{0,1\}^n$ be the j'th column of R. Then the simulator runs the rest of the setup as B would have done. What remains is to input to the ideal functionality $ID = \bar{C}$ along with a set W specifying which commitments should be wildcard commitments.

We describe how W is computed. When A^* sends a commitment (U_j, j), compute $S_j = U_j \oplus T_j$, let $N_j = \mathrm{ncw}(S_j)$ and let $e_j = N_j \oplus S_j$, where ncw gives the nearest codeword, i.e., e_j is the smallest error of S_j relative to the error correcting code. Let $E \in \{0,1\}^{n\times 2\mu}$ be the matrix with e_i as the i'th column[3].

In Lemma 4 and 5 we show that, if A^* passes the cut-and-choose test, with overwhelming probability there exists sets $F \in [2\mu]$ and $D \in [n]$ with $|D|, |F| \leq k$, so that for all $E_{i,j} = 0$ if and only if $(i,j) \in D \times F$. In other words all commitments with in $[2\mu]\setminus F$ has all of their errors isolated to indices in D. In Lemma 6 we show that, if such sets exist then the simulator can efficiently compute sets D and F with the same properties except $|F| \leq nk + k$. Thus the simulator will compute these sets and input $(ID, W) = \left(\bar{C}, F \cap \bar{C}\right)$.

We now describe how to simulate a commitment. Record the received value (y_j, j). Let $x_j = \mathrm{dec}(N_j)$, for the value N_j computed when the j'th commitment was received in the setup phase as described above. Let $m_j = y_j \oplus x_j$, and input $(\texttt{commit}, j, m_j)$ to the ideal functionality. Again the simulation is clearly perfect.

We describe how to simulate an opening. Let $c_j = (U_j, j)$ be the commitments sent earlier and let y_j be the correction value sent at commit time. Let $o_J = (x_J, r_J, J)$ be the opening. Iff

$$w_I(\mathrm{enc}(x_J; r_J)) = w_I\left(\bigoplus_{j\in J} U_j\right) \oplus w_I\left(\bigoplus_{j\in J} T_j\right) ,$$

input $(\texttt{open}, J)$ to the ideal functionality—when J was not previously opened and $W \cap J \neq \emptyset$ follow this input by $(\texttt{corrupt-open}, J, m_J)$, where $m_J = x_J \oplus y_J$ and $y_J = \bigoplus_{j\in J} y_j$. If $w_I(\mathrm{enc}(x_J; r_J)) \neq w_I\left(\bigoplus_{j\in J} U_j\right) \oplus w_I\left(\bigoplus_{j\in J} T_j\right)$ the simulator rejects the opening and terminates the protocol.

It is clear that this simulation is perfect until one of the following two events occur:

[3] Note that the simulator cannot necessarily compute $N_j = \mathrm{ncw}(S_j)$ efficiently when there are more than $d/2$ errors in S_j (i.e., when $\mathrm{hw}(e_j) > d/2$). However, from $k < d/2$, it follows that $N_j = \mathrm{ncw}(S_j)$ is correct when $\mathrm{dec}(S_j) \neq \perp$. So, if the simulator lets $e_j = 1^n$ when $\mathrm{dec}(S_j) = \perp$, it will only add too many 1 to a column e_j where there would be more than k non-zero entries anyway. It can be seen that this maintains the correctness of the below analysis, but for notational convenience we simply assume that ncw works for all inputs.

1. During the simulation of an opening, the simulator inputs $(\texttt{open}, J)$ to the ideal functionality and $J \cap W = \emptyset$ and it is not the case that $m_J = \bigoplus_{j \in J} m_j$ for the m_j stored in the ideal functionality and the m_J computed by the simulator.
2. During the simulation of an opening, the simulator inputs $(\texttt{open}, J)$ to the ideal functionality followed by $(\texttt{corrupt-open}, J, m_J)$ and the equation $m_J = \bigoplus_{j \in J} m_j$ is not consistent with the equations stored in $\mathcal{F}_{\mathsf{WCOM}}$.

We prove that each of these events occur with negligible probability.

We start with the first case. For $j \in J$, let m_j be the values computed by the simulator, i.e.,

$$S_j = U_j \oplus T_j \ , \quad N_j = \mathrm{ncw}(S_j) \ , \quad e_j = N_j \oplus S_j \ , \quad x_j = \mathrm{dec}(N_j) \ , \quad m_j = y_j \oplus x_j \ .$$

For the first event to happen we have that A^* opened the commitment to some m_J for which

$$m_J \neq \bigoplus_{j \in J} m_j \ ,$$

since

$$\bigoplus_{j \in J} m_j = y_J \oplus \bigoplus_{j \in J} x_j \ .$$

Since the opening was of the form $o_J = (x_J, r_J, J)$ with

$$w_I(\mathrm{enc}(x_J; r_J)) = w_I \left(\bigoplus_{j \in J} U_j \right) \oplus w_I \left(\bigoplus_{j \in J} T_j \right) \tag{3}$$

and the output of the opening is defined to be

$$m_J = y_J \oplus x_J \ ,$$

we get that

$$x_J \neq \bigoplus_{j \in J} x_j \ , \tag{4}$$

which we will take as the basis for our contradiction. Note for later use how the correction values y_j cancel out and $m_J \neq \bigoplus_{j \in J} m_j$ became $x_J \neq \bigoplus_{j \in J} x_j$. We later use this to simplify the proof of the more involved second event.

From $S_j = U_j \oplus T_j$ and (3) we get

$$w_I(\mathrm{enc}(x_J; r_J)) = w_I \left(\bigoplus_{j \in J} S_j \right) \ .$$

Using the argument from the discussion in Section 4, it follows from the above equation that except with negligible probability

$$\mathrm{dec} \left(\bigoplus_{j \in J} S_j \right) = x_J \ . \tag{5}$$

Namely, if $\mathrm{dec} \left(\bigoplus_{j \in J} S_j \right) \neq x_J$, then $\bigoplus_{j \in J} S_j$ and $\mathrm{enc}(x_J; r_J)$ would have Hamming distance at least $d/2$ and then $w_I(\mathrm{enc}(x_J; r_J)) = w_I \left(\bigoplus_{j \in J} S_j \right)$ occurs with negligible probability.

It follows from the way we pick $W = F \cap ID$ (as described above) and from $j \notin W$ for $j \in J$, that $\mathrm{hw}(e_j) \leq k$ for all $j \in J$. I.e., e_j only has non-zero entries with indices in D and $|D| \leq k$. Furthermore, from this it also follows that for all $j \in J$ the at most k errors in each e_j are sitting in the same k positions (those indexed by D). I.e.,

$$\mathrm{hw}\left(\bigoplus_{j \in J} e_j\right) \leq k \ .$$

From $S_j = N_j \oplus e_j$ we get that

$$\bigoplus_{j \in J} S_j = \left(\bigoplus_{j \in J} N_j\right) \oplus \left(\bigoplus_{j \in J} e_j\right) \ ,$$

i.e.,

$$\mathrm{ham}\left(\bigoplus_{j \in J} S_j, \bigoplus_{j \in J} N_j\right) \leq k \ ,$$

so

$$\mathrm{ncw}\left(\bigoplus_{j \in J} S_j\right) = \bigoplus_{j \in J} N_j \ .$$

Since

$$\mathrm{dec}\left(\bigoplus_{j \in J} N_j\right) = \bigoplus_{j \in J} \mathrm{dec}(N_j) = \bigoplus_{j \in J} x_j \ ,$$

we get that

$$\mathrm{dec}\left(\bigoplus_{j \in J} S_j\right) = \bigoplus_{j \in J} x_j \ . \tag{6}$$

Equations (4), (5) and (6) are in contradiction. Thus the first event occurs with at most negligible probability.

We then handle the second event. For simplicity we will assume that all $y_j = 0^t$ such that $m_j = x_j$ and $m_J = x_J$. This is without loss of generality following the comment made above about the y_j-values canceling out.

Note that all equations stored in $\mathcal{F}_{\mathsf{WCOM}}$ are of the form

$$\bigoplus_{j \in J'} \mathtt{X}_j = x_{J'} \ , \tag{7}$$

for some constant value $x_{J'}$ and some $J' \subset ID$. For any J' we can rewrite (7) stored in $\mathcal{F}_{\mathsf{WCOM}}$ as

$$\bigoplus_{j \in J' \cap W} \mathtt{X}_j = \left(\bigoplus_{j \in J' \setminus W} \mathtt{X}_j\right) \oplus x_{J'} \ .$$

If we let

$$\xi_{J'} = \left(\bigoplus_{j \in J' \setminus W} x_j\right) \oplus x_{J'} \ ,$$

we can then rewrite (7) as an equation

$$\bigoplus_{j \in J' \cap W} \mathtt{X}_j = \xi_{J'} \ ,$$

where $\xi_{J'}$ is fixed by the stored (x_j, j) for $j \notin W$ and the constants $x_{J'}$.

By assumption the system $\{\bigoplus_{j \in J' \cap W} \mathtt{X}_j = \xi_{J'}\}_{J' \in \mathcal{J}}$ is inconsistent. It is an easy piece of linear algebra to see that this means that there exists $\mathcal{J}_1, \mathcal{J}_2 \subseteq \mathcal{J}$ and $V \subset W$ such that

$$\bigoplus_{j\in V}\mathtt{X}_j = \bigoplus_{J'\in\mathcal{J}_1}\left(\bigoplus_{j\in J'\cap W}\mathtt{X}_j\right)$$

$$\bigoplus_{j\in V}\mathtt{X}_j = \bigoplus_{J'\in\mathcal{J}_2}\left(\bigoplus_{j\in J'\cap W}\mathtt{X}_j\right)$$

$$\bigoplus_{J'\in\mathcal{J}_1}\xi_{J'} \neq \bigoplus_{J'\in\mathcal{J}_2}\xi_{J'}\ .$$

(Simply apply Gaussian elimination to solve the system. This must fail, which will yield a system as above.)

From the last equation we get that

$$\bigoplus_{J'\in\mathcal{J}_1}\left(\left(\bigoplus_{j\in J'\setminus W}x_j\right)\oplus x_{J'}\right) \neq \bigoplus_{J'\in\mathcal{J}_2}\left(\left(\bigoplus_{j\in J'\setminus W}x_j\right)\oplus x_{J'}\right)\ ,$$

which implies that

$$\bigoplus_{J'\in\mathcal{J}_1\cup\mathcal{J}_2}\left(\bigoplus_{j\in J'\setminus W}x_j\right) \neq \bigoplus_{J'\in\mathcal{J}_1\cup\mathcal{J}_2}x_{J'}\ . \tag{8}$$

Note that the existence of the equation

$$\bigoplus_{j\in J'\cap W}\mathtt{X}_j = \left(\bigoplus_{j\in J'\setminus W}x_j\right)\oplus x_{J'}\ ,$$

stored in $\mathcal{F}_{\mathsf{WCOM}}$, implies that A^* sent a codeword $S_{J'} = \mathrm{enc}(x_{J'}; r_{J'})$ such that

$$w_I(S_{J'}) = w_I\left(\bigoplus_{j\in J'}S_j\right)\ ,$$

which means that

$$w_I\left(\bigoplus_{j\in J'\cap W}S_j\right) = w_I\left(\bigoplus_{j\in J'\setminus W}S_j\right)\oplus w_I(S_{J'})\ .$$

Combining, we get that

$$w_I\left(\bigoplus_{j\in V}S_j\right) = \bigoplus_{J'\in\mathcal{J}_1}w_I\left(\bigoplus_{j\in J'\setminus W}S_j\right)\oplus w_I(S_{J'})$$

$$w_I\left(\bigoplus_{j\in V}S_j\right) = \bigoplus_{J'\in\mathcal{J}_2}w_I\left(\bigoplus_{j\in J'\setminus W}S_j\right)\oplus w_I(S_{J'})\ .$$

Using transitivity of $=$ and that $w_I(S_{J'}) = w_I(\mathrm{enc}(x_{J'}; r_{J'}))$ we conclude that

$$\bigoplus_{J'\in\mathcal{J}_1}\left(w_I\left(\bigoplus_{j\in J'\setminus W}S_j\right)\oplus w_I(\mathrm{enc}(x_J; r_{J'}))\right) =$$

$$\bigoplus_{J'\in\mathcal{J}_2}\left(w_I\left(\bigoplus_{j\in J'\setminus W}S_j\right)\oplus w_I(\mathrm{enc}(x_{J'}; r_{J'}))\right)\ ,$$

which, implies that

$$w_I\left(\bigoplus_{J'\in\mathcal{J}_1\cup\mathcal{J}_2}\left(\bigoplus_{j\in J'\setminus W} S_j\right)\right) = w_I\left(\bigoplus_{J'\in\mathcal{J}_1\cup\mathcal{J}_2}\mathrm{enc}(x_{J'};r_{J'})\right)\ . \tag{9}$$

By construction of W, the string $\bigoplus_{J'\in\mathcal{J}_1\cup\mathcal{J}_2}\left(\bigoplus_{j\in J'\setminus W} S_j\right)$ has distance at most k to a codeword encoding $\bigoplus_{J'\in\mathcal{J}_1\cup\mathcal{J}_2}\left(\bigoplus_{j\in J'\setminus W} x_j\right)$, namely, they differ in at most the k positions in the set D, as we sum only over $j \notin W$. By linearity, the string $\bigoplus_{J'\in\mathcal{J}_1\cup\mathcal{J}_2}\mathrm{enc}(x_{J'};r_{J'})$ is a codeword, encoding $\bigoplus_{J'\in\mathcal{J}_1\cup\mathcal{J}_2} x_{J'}$. By (9) and I being chosen at random and independent of the view of A^*, we have that

$$\mathrm{ham}\left(\bigoplus_{J'\in\mathcal{J}_1\cup\mathcal{J}_2}\left(\bigoplus_{j\in J'\setminus W} S_j\right), \bigoplus_{J'\in\mathcal{J}_1\cup\mathcal{J}_2}\mathrm{enc}(x_{J'};r_{J'})\right) \le k$$

except with negligible probability. It then follows from $k < d/2$ that with overwhelming probability the string $\bigoplus_{J'\in\mathcal{J}_1\cup\mathcal{J}_2}\mathrm{enc}(x_{J'};r_{J'})$ is the only codeword within distance k of $\bigoplus_{J'\in\mathcal{J}_1\cup\mathcal{J}_2}\left(\bigoplus_{j\in J'\setminus W} S_j\right)$. Hence $\bigoplus_{J'\in\mathcal{J}_1\cup\mathcal{J}_2}\mathrm{enc}(x_{J'};r_{J'})$ must be the codeword encoding $\bigoplus_{J'\in\mathcal{J}_1\cup\mathcal{J}_2}\left(\bigoplus_{j\in J'\setminus W} x_j\right)$. From this it follows that $\bigoplus_{J'\in\mathcal{J}_1\cup\mathcal{J}_2}\mathrm{enc}(x_{J'};r_{J'})$ is an encoding of both $\bigoplus_{J'\in\mathcal{J}_1\cup\mathcal{J}_2} x_{J'}$ and $\bigoplus_{J'\in\mathcal{J}_1\cup\mathcal{J}_2}\left(\bigoplus_{j\in J'\setminus W} x_j\right)$. This contradicts (8), and thus the second event can only occur with negligible probability. This concludes the proof. □

Now all that remains is to show that the simulator can indeed pick a set W as described in the proof. I.e., that we can pick sets $F \subset [2\mu]$ and $D \subset [n]$, where $|D| \le k$ and $|F| \le nk + k$, so that all commitments in $[2\mu] \setminus F$ have their errors isolated to indices in D.

To this end let a set of *independent errors*, be a set $\{(r_i, c_i)\}_{i=1}^{Q}$ for which $r_i \in [n]$, $c_i \in [2\mu]$, $|\{r_i\}_{i=1}^{Q}| = |\{c_i\}_{i=1}^{Q}| = Q$ and $E_{r_i,c_i} = 1$ for $i = [Q]$ (recall that E is the matrix with the error vectors of all commitments as its columns). Let the *independent errors value* $\nu(E)$ to be the size of the largest set of independent errors.

Lemma 4. *The probability that A^* is not caught cheating during the setup is $\le \alpha^{\nu(E)}$ for a positive constant $\alpha < 1$ independent of ν. In particular, if $\nu(E) = \Theta(k)$, then A^* is caught except with overwhelming probability.*

Proof. To see this, let $\{(r_i, c_i)\}_{i=1}^{\nu}(E)$ be a maximal set of independent errors. Note that if $j \in C$ for some $j = c_i \in \{c_i\}_{i=1}^{\nu}(E)$ for the challenge C sent by B, i.e., A^* is asked to open commitment number $j = c_i$, she must send an opening (x'_j, r'_j, j) with $\mathrm{enc}\left(x'_j; r'_j\right) = N_j$. Assume namely that she does not: Then $\mathrm{ham}\left(\mathrm{enc}(x'_j; r'_j), S_j\right) \ge d/2$, and from the above it also follows that $w_I(U_j) \oplus w_I(T_j) = w_I(U_j \oplus T_j) = w_I(S_j)$, so $w_I(S_j) = w_I(\mathrm{enc}(x'_j; r'_j))$. Since I is independent of the view of A^*, the probability that $w_I(A) = w_I(B)$, when $\mathrm{ham}(A, B) \ge d/2$ is at most $\left(\frac{n-d/2}{n}\right)^u$, which is negligible. If A^* does use an opening $\left(x'_j, r'_j, j\right)$ with $\mathrm{enc}\left(x'_j; r'_j\right) = N_j$, then $e_j = \mathrm{enc}\left(x'_j; r'_j\right) \oplus S_j$ has a 1 in position r_i, so A^* is caught with probability at least $\frac{u}{n}$. The probability that $j = c_i$ is chosen for opening, i.e., $j \in C$, is at least $\frac{1}{2}$. This means the probability A^* is not caught is at most $\alpha = \frac{1}{2}\left(1 - \frac{n-u}{n}\right)$. This holds independently for the $\nu(E)$ errors (r_i, c_i) as they sit in different rows and columns by the definition of ν. □

By Lemma 4 we can henceforth without loss of generality assume that the independent error value of E is less than k.

We then let a pair of *error isolating* sets be two sets $D \subset [n]$ and $F \subset [2\mu]$ such that if $i \in [n] \setminus D$ and $j \in [2\mu] \setminus F$, then $E_{i,j} = 0$. I.e., sets so that all the ones in E are isolated to at most $|D|$ rows and $|F|$ columns. We let the *error isolation value* $\mu(E)$ be the smallest integer Q such that there exists error isolating sets D and F with $|D|, |F| \le Q$. We show that the error isolation value is less than or equal the independent error value.

Lemma 5. *$\mu(E) \leq \nu(E)$.*

Proof. If E contains only zeros, then $\mu(E) = 0 = \nu(E)$. Otherwise, we prove the lemma by constructing a set of independent errors G from E, so that $\mu(E) \leq |G|$: pick (r_1, c_1) to be in G such that $E_{r_1,c_1} = 1$. Then greedily pick values (r_i, c_i) such that $E_{r_i,c_i} = 1$ and such that r_i is different from all $r_{j<i}$ and c_i is different from all $c_{j<i}$, and put these in G. Do this till it is not longer possible to pick a new pair (r_i, c_i). Then by definition of ν we have $|G| \leq \nu(E)$ (since G is a set of independent errors). Furthermore, note that all errors are now isolated to either rows indexed by r_i or columns indexed by c_i for some pair $(r_i, c_i) \in G$. Thus by definition of μ we have $\mu(E) \leq |G|$. □

Finally we show that an pair of almost optimal error isolating sets can be found efficient by simulator.

Lemma 6. *When $\mu(E) \leq k$, the simulator can efficiently compute D and F with $|D| \leq k$ and $|F| \leq nk + k$ such that if $i \in [n] \setminus D$ and $j \in [2\mu] \setminus F$, then $E_{i,j} = 0$. I.e., all the errors in the codewords are isolated to the k rows of D and $nk + k$ columns of F*[4].

Proof. To construct F and D we proceed as follows: First add the index of any column in E with Hamming weight more than k to F and add the index of any row with Hamming weight more than k to D. There are at most k such columns and k such rows, or we could not have $\mu(E) \leq k$. Now all remaining rows have at most Hamming weight k. There are at most n remaining rows. This gives a total of nk non-zero entries in the remaining rows. Add the index of any columns with one of these non-zero entries to F. This gives $|F| \leq nk + k$. □

C Proof of complete construction

We here formally prove the theorems and lemmas needed to make our commitment scheme usable in the MiniLEGO protocol in Fig. 7.

C.1 Oblivious opening

Proof (Theorem 3). We focus on the simulator when simulating the **Oblivious-Opening** command, since for all other commands we can use the simulator from the proof of Theorem 2. Furthermore, the case of corrupted B is trivial so we will focus on corrupted A.

On output $(\texttt{OT-choose}, \mathit{otid})$ from $\mathcal{F}_{\mathsf{COM}}$ the simulator observes the messages (O'_0, O'_1) as A inputs them to $\mathcal{F}_{\mathrm{OT}}$. Later when A sends OTP keys M_0 and M_1 the simulator computes both openings $o'_0 = O'_0 \oplus M_0$ and $o'_1 = O'_1 \oplus M_1$. For each $i \in \{0, 1\}$ the simulator parses o'_i as $o'_i = (x_{J_i}, r_{J_i}, J_i)$ and inputs $(\texttt{OT-open}, \mathit{otid}, J_0, J_1)$ to $\mathcal{F}_{\mathsf{COM}}$. So far the simulation is perfect.

Note that from this point on the behavior of B in the real world protocol is the same as when he receives o'_b as the opening in a regular opening. Since there is no other communication from B to A, A will only learn whether or not B accepts or rejects the opening of o'_b. Apart from handling the input $(\texttt{guess}, g)$, essentially the same goes for the ideal functionality $\mathcal{F}_{\mathsf{COM}}$. Thus, if the simulator ignores o'_{1-b} and runs the simulator from the proof of Theorem 2 when opening o'_b, the simulation will go through. The only problem is that the simulator does not know the value of b. To solve this the simulator will run the simulator from the proof of Theorem 2 on both o'_0 and o'_1, and if necessary use $\mathcal{F}_{\mathsf{COM}}$ to try and guess b. There are three cases:

1. The simulator of Theorem 2 rejects both openings. In this case the simulator of the Oblivious-Opening can safely reject and terminate, since o'_b must be a rejecting opening.
   The simulation is clearly perfect.

[4] Given enough time the simulator could get $|F|$ down to k, but we conjecture that it is NP hard to get get $|F|$ down to $O(k)$ due to the similarity to the bipartite clique problem.

2. The simulator of Theorem 2 accepts both openings. In this case o'_b must be an accepting opening. The simulator of the Oblivious-Opening can safely input $(\texttt{guess}, \perp)$ followed by $(\texttt{corrupt-open}, J_0, J_1, m_{J_0}, m_{J_1})$ to $\mathcal{F}_{\mathsf{COM}}$, were m_{J_0} and m_{J_1} are computed as by the simulator of Theorem 2.
   The simulation is indistinguishable from the real world by the same argument as in the proof of Theorem 2.
3. For some $g \in \{0,1\}$ the simulator of Theorem 2 rejects on opening o'_{1-g} but not o'_g. In this case the simulator of the Oblivious-Opening inputs $(\texttt{guess}, g)$ to $\mathcal{F}_{\mathsf{COM}}$. If $\mathcal{F}_{\mathsf{COM}}$ terminates the simulator does the same. Otherwise, the simulator learns that $b = g$, and proceeds to input $(\texttt{corrupt-open}, J_0, J_1, m_{J_0}, m_{J_1})$ where m_{J_b} is computed as by the simulator of Theorem 2 and $m_{J_{1-b}} = 0^t$.
   If $\mathcal{F}_{\mathsf{COM}}$ terminates on input $(\texttt{guess}, g)$ the simulation is perfect since o'_b is a rejecting opening. Otherwise, the simulation is indistinguishable from the real world by the same argument as in the proof of Theorem 2.

So since in all cases the simulation is indistinguishable from the from the real world protocol, this concludes our proof.

□

C.2 Less wildcards

Proof (Theorem 4).

In the setup phase the simulator behaves as an honest B while fully controlling the $\mathcal{F}_{\mathsf{WCOM}}$ functionality. Doing so the simulator can read of the set of wildcard commitments input by corrupt A W' when initializing $\mathcal{F}_{\mathsf{WCOM}}$ with $(\texttt{init}, ID', W')$ for $|W'| \leq nk+k$ and $|ID'| = 2\mu'$. For all inputs $(\texttt{commit}, i, x_i)$ made by A during the setup the simulator records (x_i, i). For all inputs $(\texttt{open}, \{i, \pi(i)\})$ followed by $(\texttt{corrupt-open}, \{i, \pi(i)\}, z_i)$ made by A the simulator records (z_i, i).

After the setup phase the simulator computes the set $W = \{i \in ID \cap W' | \pi(i) \in W'\}$, i.e., W is the set of wildcard commitments in ID that were paired with another wildcard commitments. To initialize the $\mathcal{F}_{\mathsf{COM}}$ functionality, $(\texttt{init}, ID, W)$ with $|ID| = \mu'$ and $|W| \leq k$, the simulator then inputs ID as the set of commitment identifiers and W as the set of wildcard commitments. This makes $\mathcal{F}_{\mathsf{COM}}$ output ID to B and simulation is clearly perfect.

From this point on the simulator is going to simulate the $\mathcal{F}_{\mathsf{WCOM}}$ functionality by essentially just forwarding messages between A and $\mathcal{F}_{\mathsf{COM}}$. Note though that if A gives an input to $\mathcal{F}_{\mathsf{WCOM}}$ with an id $j \in ID' \setminus ID$ then this input cannot simply be forwarded to $\mathcal{F}_{\mathsf{COM}}$ because that functionality only knows about id's in ID. This is not a problem though because for any such input A could make, $\mathcal{F}_{\mathsf{WCOM}}$ outputs j to B and honest B terminates. So, in case of such an input the simulator just terminates the protocol. In the following we will assume that A only makes inputs to $\mathcal{F}_{\mathsf{WCOM}}$ with id's in ID.

To simulate the j'th commitment the simulator records the value (y_j, j) sent by A. If $j \in W' \setminus W$ the simulator lets $m_j = (x_{\pi(j)} \oplus z_j) \oplus y_j$. Otherwise, the simulator lets $m_j = x_j \oplus y_j$. Then the simulator inputs $(\texttt{commit}, j, m_j)$ to the $\mathcal{F}_{\mathsf{COM}}$, and the functionality outputs $(\texttt{commit}, j)$ to B.

Note that for $j \in ID \setminus W'$ the $\mathcal{F}_{\mathsf{COM}}$ functionality used in the ideal world stores the equation $\mathtt{X}_j = x_j \oplus y_j$ while the $\mathcal{F}_{\mathsf{WCOM}}$ functionality used in the real world would have stored $\mathtt{X}_j = x_j$. Furthermore, for $j \in W' \setminus W$ $\mathcal{F}_{\mathsf{COM}}$ stores $\mathtt{X}_j = x_{\pi(j)} \oplus z_j \oplus y_j$ while $\mathcal{F}_{\mathsf{WCOM}}$ stores $\mathtt{X}_j \oplus \mathtt{X}_{\pi(j)} = z_j$ and $\mathtt{X}_{\pi(j)} = x_{\pi(j)}$. Thus, ignoring the correction values, the restrictions on how to open commitment $j \in ID$ is equivalent in both worlds. I.e., the XOR of commitments in any set $J \subset ID$ can, in the real world, be opened to x_J using $\mathcal{F}_{\mathsf{WCOM}}$ if and only if the XOR of commitments in J can be opened to $m_J = x_J \oplus \left(\bigoplus_{j \in J} y_j\right)$ in the ideal world using $\mathcal{F}_{\mathsf{COM}}$.

On input $(\texttt{open}, J)$ from A the simulator forwards the input to $\mathcal{F}_{\mathsf{COM}}$. On input $(\texttt{corrupt-open}, J, x_J)$ the simulator inputs $\left(\texttt{corrupt-open}, J, m_J = x_J \oplus (\bigoplus_{j \in J} y_j)\right)$ to $\mathcal{F}_{\mathsf{COM}}$. By the discussion above the simulation is perfect.

To simulate the Oblivious-Openings the simulator forwards messages between A and the $\mathcal{F}_{\mathsf{COM}}$ in essentially the same way as for the regular opening. I.e., all messages are forwarded verbatim except if A inputs the message $(\texttt{corrupt-open}, J_0, J_1, x_{J_0}, x_{J_1})$. In that case the simulator inputs to $\mathcal{F}_{\mathsf{COM}}$ the command $\left(\texttt{corrupt-open}, J_0, J_1, x_{J_0} \oplus (\bigoplus_{j \in J_0} y_j), x_{J_1} \oplus (\bigoplus_{j \in J_1} y_j)\right)$. Again the simulation is perfect.

The only thing left to argue is that with overwhelming probability $|W| \leq k$. Consider any of μ' pairs of commitments opened in the setup phase. The probability that both commitments in this pair are wildcard commitments less than $\left(\frac{nk+k}{2\mu'}\right)^2$. As there are μ' pairs, by union bound we have

$$\begin{aligned}
\Pr(|W| \geq k) &= \binom{k}{\mu'} \left(\frac{nk+k}{2\mu'}\right)^{2k} \\
&\leq {\mu'}^{k} \left(\frac{nk+k}{2\mu'}\right)^{2k} \\
&\leq (nk+k)^{2k} nk + k^{-2k} 2^{-2k} = 2^{-2k} < 2^{-k} \ .
\end{aligned}$$

where the last inequality is by $\mu' > (nk+k)^2$. □

C.3 Or-Opening

Proof (Theorem 5). From Theorem 4 we have that the protocol implements the **Setup**, **Commit**, **Open** and **Oblivious-Opening** commands of $\mathcal{F}_{\mathsf{COM}}$ with $|ID| = \mu + 6k\omega$ and $|W| \leq k$ in the $(\mathcal{F}_{\mathsf{WCOM}})$-hybrid model when initializing $\mathcal{F}_{\mathsf{WCOM}}$ with $|ID'| = \mu' = 2(\mu + 6k\omega)$ and $|W'| \leq nk + k$. So in the rest of this proof we will simply assume that we have access to these commands of $\mathcal{F}_{\mathsf{COM}}$ with $|ID| = \mu + 6k\omega$ and $|W| \leq k$. So for these commands the simulator simply simulates all interaction between A and $\mathcal{F}_{\mathsf{WCOM}}$ by interacting correspondingly with the $\mathcal{F}_{\mathsf{COM}}$ functionality and recording all messages. Clearly this simulation is perfect, so for the remainder of proof we will concentrate on the **Or-Opening** command.

For the Or-Opening the simulator acts as the honest B. Note that since the simulator has recorded all messages between A and $\mathcal{F}_{\mathsf{WCOM}}$ it can simulate the $\mathcal{F}_{\mathsf{WCOM}}$ functionality perfectly. This means that when the simulator rejects the opening the simulation has been perfect.[5]

Assuming that the opening was not rejected, the simulator then needs to interact with $\mathcal{F}_{\mathsf{COM}}$ so that the functionality outputs $(\mathtt{OR\text{-}open}, J_0, J_1, m)$ to B with overwhelming probability. I.e., the simulator must find an a' so that either $J_{a'} \cap W = \emptyset$ and $\bigoplus_{j \in J_{a'}} m_j = m$ or $J_{a'} \cap W \neq \emptyset$ and the equations stored in the $\mathcal{F}_{\mathsf{COM}}$ functionality allows to corruptly open the XOR of commitments $J_{a'}$ to m.

To show that the simulator can do this we first show that, given the opening was not rejected, there exists an $i' \in [3k]$ and a $j^{i'}_{a'} \in \{j^1_0, j^1_1, \ldots, j^{3k}_0, j^{3k}_1\} = D_d$ with the properties that

1. $m^{i'}_{a^{i'}} = m$, where $m^{i'}_{a'}$ is the message committed to as commitment $j^{i'}_{a'}$.
2. The XOR of commitments in $J_{a' \oplus p^{i'}}$ can be opened to $m^{i'}_{a'}$ (honestly or corruptly).

Notice that such a $j^{i'}_{a'}$ implies that $J_{a' \oplus p^{i'}}$ can be opened to m.

We would like to prove this by showing that for each $i \in [3k]$ if there is not a $j^i_{a'}$ with both properties then the opening is rejected with probability $\frac{1}{2}$. Thus, assuming the opening is not rejected the probability that there is no $i \in [3k]$ with a $j^i_{a'}$ with both properties is at most 2^{-3k}. However, the wildcard commitments get in the way of such a straight forward proof.

Instead we assume that the set $H \subset [3k]$ where for all $i \in H$ neither j^i_0 nor j^i_1 is a wildcard commitment, i.e.,

$$H = \left\{ i \in [3k] | j^i_0, j^i_1 \notin W \right\} \ ,$$

has size least k. Below we show that since honest B chooses the set $D_d \subset ID$ uniformly at random, this is the case with overwhelming probability.

We then have that for each $i \in H$ the opening will reject with probability at least $\frac{1}{2}$ if there is not a $j^i_{a'}$ with property 1. Namely, if $c^i = 1$ A must open a commitment $j^i_{a'}$ to m and by $i \in H$ we have that $j^i_{a'} \notin W$ can only be opened to $m^i_{a'}$.

[5] Note that to simulate the $\mathcal{F}_{\mathsf{WCOM}}$ functionality the simulator needs to know all the equations stored inside the functionality. This is why we cannot do Or-Openings after the first Oblivious-Opening: After an Oblivious-Opening the equations stored in $\mathcal{F}_{\mathsf{WCOM}}$ depend on B's secret bit b which would be unknown to the simulator.

On the other hand for each $i \in H$ the opening will also reject with probability at least $\frac{1}{2}$ if there is not a $j^i_{a'}$ with property 2. Namely, if $c^i = 0$ A must be able to open the XOR of commitments in $J_{a' \oplus p^i} \cup \{j^i_{a'}\}$ to 0, but since $j^i_{a'} \notin W$ this is equivalent to being able to open the XOR of commitments in $J_{a' \oplus p^i}$ to $m^i_{a'}$.

Thus with probability at least $1 - 2^{-k}$ there exists an $i \in H$ and a $j^i_{a'}$ with both the above properties. Note that since the simulator knows all messages sent to $\mathcal{F}_{\mathsf{WCOM}}$ by A it can easily find such a $j^i_{a'}$.

The simulator then inputs $(\texttt{OR-open}, J_0, J_1, a')$ to the $\mathcal{F}_{\mathsf{COM}}$ functionality, and if $J_{a'} \cap W \neq \emptyset$ proceeds to input $(\texttt{corrupt-open}, J_{a'}, m)$. The $\mathcal{F}_{\mathsf{COM}}$ functionality then outputs (J_0, J_1, m) to B and the simulation is indistinguishable to the real world protocol.

To conclude the proof we must show that with overwhelming probability $|H| \geq k$. To see this we use that if $|D_d \cap W| \leq s$ then $|H| \geq 3k - 2s$. Thus we will show that $|D_d \cap W| \leq k$ with overwhelming probability.

Consider some $j \in D_d$ and assume

$$\left(\bigcup_{i \in [\omega]} D_i \setminus \{j\} \right) \cap W = \emptyset \ ,$$

then the probability that $j \in W$ is $k/((\mu + 6k\omega) - 6k\omega + 1) = k/(\mu + 1)$. Therefore, for each $j \in \left\{j^1_0, j^1_1, \ldots, j^{3k}_0, j^{3k}_1\right\}$ we have that

$$\Pr(j \in W) \leq \frac{k}{\mu + 1} \ .$$

Now, let $Y_1, \ldots, Y_{6k}$ be random independent variables where each $Y_i = 1$ with probability $k/(\mu + 1)$ and $Y_i = 0$ otherwise, and let $S = \sum_{i \in [6k]} Y_i$. We then have that

$$1 - \Pr(|H| \geq k) \leq \Pr(|D_d \cap W| > k|) \leq \Pr(S > k|) \ .$$

The random variable S has expected value $\mathrm{E}[S] = 6k\frac{k}{\mu+1} \leq 1$ by assumption on μ. Thus by Hoeffdings inequality we have that

$$\Pr(S > k) \leq e^{-2\frac{2k^2}{6k}} = e^{-\frac{2}{3}k} \ ,$$

which means that

$$\Pr(|H| \geq k) \geq 1 - e^{-\frac{2}{3}k} \ .$$

I.e., we have shown that H has size at least k with overwhelming probability. □

D List of Variable Names

Table 2. Overview of the different variables used along with their semantic meaning.

k	Statistical security parameter
t	Computational security parameter
s	Amount of AND gates in a circuit
ℓ	Amount of input and output bits to the circuit
n	Size of commitments in bits, $\Theta(k)$
u	Size of message to commit to in bits
ϕ	The randomness used in the XOR-homomorphic commitments
d	The minimum distance of the secret sharing error correcting code (ssecc)
w	Defined such that $d = 2w + 1$ and $w < n/2$
μ	Variable expressing the amount of commitments in a given protocol
κ	Variable expressing amount of wildcard commitments in a given protocol
ω	Amount of Or-Openings
ρ	Replication factor. Defined as $\rho = O(k/\log(s))$
Γ	Gates in the replicated circuit. Defined as $2\rho s$
Γ'	Keys in the replicated circuit. Defined as $3\Gamma + 1$