A preliminary version of this paper appears in the proceedings of the *18th International Conference on Applied Cryptography and Network Security (ACNS 2020)*. This is the full version.

Secure Two-Party Computation in a Quantum World

Niklas Büscher[1], Daniel Demmler[2], Nikolaos P. Karvelas[1], Stefan Katzenbeisser[3], Juliane Krämer[4], Deevashwer Rathee[5], Thomas Schneider[6], and Patrick Struck[4]

[1] SecEng, Technische Universität Darmstadt, Germany
{buescher,karvelas}@seceng.informatik.tu-darmstadt.de
[2] SVS, Universität Hamburg, Germany
demmler@informatik.uni-hamburg.de
[3] Universität Passau, Germany
stefan.katzenbeisser@uni-passau.de
[4] CDC, Technische Universität Darmstadt, Germany
{jkraemer,pstruck}@cdc.tu-darmstadt.de
[5] Department of Computer Science, IIT (BHU) Varanasi, India
deevashwer.student.cse15@iitbhu.ac.in
[6] ENCRYPTO, Technische Universität Darmstadt, Germany
schneider@encrypto.cs.tu-darmstadt.de

Abstract. Secure multi-party computation has been extensively studied in the past years and has reached a level that is considered practical for several applications. The techniques developed thus far have been steadily optimized for performance and were shown to be secure in the classical setting, but are not known to be secure against quantum adversaries.
In this work, we start to pave the way for secure two-party computation in a quantum world where the adversary has access to a quantum computer. We show that post-quantum secure two-party computation has comparable efficiency to their classical counterparts. For this, we develop a lattice-based OT protocol which we use to implement a post-quantum secure variant of Yao's famous garbled circuits (GC) protocol (FOCS'82). Along with the OT protocol, we show that the oblivious transfer extension protocol of Ishai et al. (CRYPTO'03), which allows running many OTs using mainly symmetric cryptography, is post-quantum secure. To support these results, we prove that Yao's GC protocol achieves post-quantum security if the underlying building blocks do.

Keywords: Post-quantum security · Yao's GC protocol · Oblivious transfer · Secure two-party computation · Homomorphic encryption

1 Introduction

In light of recent advances in quantum computing, it seems that we are not far from the time that Shor's algorithm [53] can be executed on a real quantum computer. There are several experts that estimate that quantum computers with the required performance and features will be available within the next one or two decades [8, 41]. Recently Google researchers claimed to have achieved quantum-supremacy, i.e., being able to perform a specific type of computation on a quantum computer, that is infeasible on conventional supercomputers [6]. This will give rise to the so-called quantum era [13], in which one of the parties involved in a cryptographic protocol might be able to perform local quantum computation during the protocol run whereas the communication between the parties remains classical. It is therefore necessary to analyse the security of cryptographic protocols against quantum adversaries. Some industrial security review processes already mandate post-quantum security for building blocks that are used in secure systems, which shows that the security threat posed by quantum computers is getting attention even outside of academia. The development of post-quantum secure cryptographic primitives such as [2, 20, 33, 40] in the past years shows the importance that the cryptographic community attributes to the problem. However, more complex cryptographic protocols have not yet been extensively studied, even though Canetti's UC framework [17] and Unruh's quantum lifting [56] provide the necessary theoretical foundations for achieving this task.

One such complex cryptographic protocol is secure two-party computation. In recent years, Yao's general solution for secure computation, the so-called 'Yao's Garbled Circuits' (GC) protocol [59], emerged from a theoretical idea to a powerful and versatile privacy-enhancing technology. Extensive research on the adversarial model, e.g., security against malicious adversaries [36, 57], and several protocol optimizations made GCs practical for many use cases in the last decade. Protocol optimizations such as Garbled Row Reduction [43, 47], the free-XOR technique [34], fixed-key garbling [10], the half-gates approach [60], OT extension [7, 32], and also the use of hardware instructions such as AES-NI or parallelization improved the runtime of the protocol by orders of magnitude.

Despite its maturity and efficiency, e.g., being a constant round protocol using mostly symmetric cryptographic primitives, the security of Yao's GC protocol has only been studied against classical adversaries. Unruh showed that multi-party computation is achievable from commitments in a fully-quantum setting [56]. In their setting quantum computers are ubiquitous, in the post-quantum setting we consider only the adversary has quantum computing power. However, the gap between the highly optimized GC solution used as a privacy-enhancing technology today and this theoretical construction in the fully-quantum case, makes the transition from the classical to the post-quantum case challenging. Therefore, securing Yao's GC protocol against quantum adversaries is of high practical and theoretical interest. A prominent example is the standardization process on post-quantum cryptographic primitives initiated by the NIST [45].

Our Contributions. In this paper, we extend the line of research for secure computation to the post-quantum setting, combining theory and practice. On the practical side, we complement the theoretical results by showing that post-quantum secure two-party computation achieves performance that is close to existing classical implementations. On the theoretical side, we pave the way for post-quantum secure two-party computation by proving security of Yao's GC protocol and OT extension. Our contributions are detailed below.

1) In Section 3, we develop an efficient post-quantum secure OT protocol based on the ring learning with errors (RLWE) problem. The protocol is based on an additively homomorphic encryption scheme. The general method to do this is well-known, but we show how to implement this very efficiently. In particular, we use batching to compute a large number of OTs at the cost of one, while maximizing the packing efficiency and the parallelism we get from homomorphic single instruction multiple data (SIMD) operations. Additionally, we show that OT extension introduced by Ishai et al. [32] is secure against quantum adversaries.

2) We implement our OT protocol in C++ using the Microsoft SEAL homomorphic encryption library [52]. In Section 4 we show that our implementation achieves a throughput of 89k PQ-OTs per second, thus being a promising replacement for existing classical OT protocols. Furthermore, we implement a post-quantum secure version of Yao's GC protocol using our OT implementation and compare its performance with implementations secure in the classical setting. While a performance loss is expected, our results show that it is in fact tolerable. Our implementations are open-source software under the permissive MIT license and are available online at `https://encrypto.de/code/pq-mpc`.

3) In Section 5, we strengthen our practical results by proving that Yao's GC protocol can be hardened to withstand quantum attackers by replacing the underlying components with post-quantum-secure variants. We do so by showing that the classical proof by Lindell and Pinkas [35] also holds in the post-quantum setting. In addition, we give a security proof for double encryption security in the post-quantum setting adapted to the quantum random oracle model (QROM). While these results sound very natural, we stress that they have not been formally proven thus far.

Related Work. There are several works related to Yao's protocol, oblivious transfer and post-quantum security. We give a brief overview of results that are relevant for our work. A full proof of classical security for Yao's garbled circuits protocol was given in [35]. In [18], the free-XOR optimization [34] of Yao's protocol was proven secure under a weaker assumption than the random oracle model. The point-and-permute optimization was introduced and implemented in [9, 38]. A formally verified software stack for Yao's garbled circuits was presented in [4]. Known instantiations for post-quantum secure OT protocols are either based on the code-based McEliece crypto system [22] or on the learning with errors (LWE) problem [15]. In [39], the authors build OT extension from post-quantum secure primitives, but do not prove it post-quantum secure.

2 Preliminaries

Within this section we give the mandatory background regarding notation, encryption schemes, oblivious transfer, and Yao's protocol for our paper. Additional background on the quantum random oracle model and the additively homomorphic encryption scheme is given in Appendix A.

2.1 Notation

We denote the modulus reduction in the symmetric interval $[-q/2, q/2)$ by $[\cdot]_q$, and the modulus reduction of an integer a in the positive interval $[0, q)$ by $a \bmod q$. The set of integers $\{1, \ldots, n\}$ is denoted by $[n]$. We use bold case letters for vectors, e.g., $\boldsymbol{a}$, and identify the i-th entry of a vector $\boldsymbol{a}$ by (a_i). In a secure two-party computation protocol, two parties with corresponding inputs x and y want to compute $\mathcal{F}(x, y)$ for a function $\mathcal{F}$ known by both parties. We use statistical security parameter $\sigma = 40\,\text{bit}$, the symmetric security parameter κ, and the public-key security parameter λ.

In our proofs we use the code-based game playing framework by Bellare and Rogaway [12]. At the start of the game, the initialize procedure is executed and its output is given as the input to the adversary. The output of the game is the output of the finalize procedure which takes as input whatever the adversary outputs. In between, the adversary has oracle access to all other procedures described in the game. For a game G and an adversary $\mathcal{A}$, we write $\mathcal{A}^{\mathsf{G}} \to y$ for the event that the output of $\mathcal{A}$ is y when interacting with G. Likewise, we denote the event that the G outputs y when interacting with $\mathcal{A}$ by $\mathsf{G}^{\mathcal{A}} \to y$. For simplicity, we assume that for any table $f[\,]$ its entries are initialized to $\perp$ at the start of the game. We denote homomorphic addition and subtraction as $\boxplus$ and $\boxminus$, respectively. Homomorphic multiplication with a plaintext is denoted by $\boxdot$. The detailed description of an additively homomorphic encryption scheme is given in Appendix A.2. We assume the reader is familiar with the fundamental concepts of quantum computation like the Dirac notation and measurements. For a more thorough discussion we refer to [44].

2.2 Encryption

A secret key encryption scheme E_S is a pair of efficient algorithms $\mathtt{Enc}$ and $\mathtt{Dec}$ for encryption and decryption, where $\mathtt{Enc}(k, m) \to c$ and $\mathtt{Dec}(k, c) \to m$ for message m, ciphertext c, and key k.

A basic security notion for secret key encryption schemes is *indistinguishability under chosen plaintext attacks* (IND-CPA) which asks an adversary to distinguish between the encryption of two adversarial chosen messages. Below we formally define the corresponding post-quantum security notion, that is, pq-IND-CPA, for secret key encryption schemes in the QROM. Note that the security notion allows for multiple challenges which is an important requirement in the security proof of Yao's protocol.

Definition 1. *Let* $\mathrm{E}_S = (\mathtt{Enc}, \mathtt{Dec})$ *be a secret key encryption scheme and let the game* $\mathsf{pq\text{-}INDCPA}$ *be defined as in Fig. 1. We say that* E_S *is* pq-IND-CPA-secure *if the following term is negligible for any quantum adversary* $\mathcal{A}$*:*

$$\mathbf{Adv}^{\text{pq-ind-cpa}}_{\mathrm{E}_S}(\mathcal{A}) = 2 \Pr\left[\mathsf{INDCPA}^{\mathcal{A}} \to \text{true}\right] - 1\,.$$

```
Game pq-INDCPA
procedure Initialize
  $b \leftarrow_\$ \{0,1\};\ \ k \leftarrow_\$ \mathcal{K}$

procedure $\mathsf{Enc}(m)$
  $c \leftarrow_\$ \mathtt{Enc}(k, m)$
  return $c$

procedure Finalize $(b')$
  return $(b' = b)$

procedure $\mathsf{E}(m_0, m_1)$
  $c \leftarrow_\$ \mathtt{Enc}(k, m_b)$
  return $c$

procedure $\mathsf{O_H}(\sum \alpha_{x,y} |x, y\rangle)$
  return $\sum \alpha_{x,y} |x, y \oplus \mathsf{H}(x)\rangle$
```

Fig. 1. Game to define pq-IND-CPA security for secret key encryption schemes.

2.3 Oblivious Transfer

An oblivious transfer (OT) protocol is a protocol in which a sender transfers one of multiple messages to a receiver, but it remains oblivious as to which message has been transferred. At the same time, the receiver can only select a single message to be retrieved. We focus on 1-out-of-2 OTs, where the sender inputs two ℓ-bit strings m_0, m_1 and the receiver inputs a choice bit $b \in \{0, 1\}$. At the end of the protocol, the receiver obliviously receives only m_b. OT guarantees that the sender learns nothing about the choice bit b, and that the receiver learns nothing about the other message m_{1-b}. OT protocols require public key cryptography as shown in [31], and were assumed to be very costly in the past. However, in 2003 Ishai et al. [32] presented the idea of *OT extension*, which significantly reduces the computational costs of OTs for many interesting applications of MPC by extending a small number of 'real' base OTs to a large number of OTs using only symmetric cryptographic primitives.

2.4 Description of Yao's Protocol

Yao's garbled circuits protocol [59] is a fundamental secure two-party computation protocol. The protocol consists of two cryptographic primitives: a secret key encryption scheme and an OT protocol. It is executed by two parties, the *garbler* $\mathcal{G}$ and the *evaluator* $\mathcal{E}$ with corresponding inputs x and y. At the end of the protocol, both parties want to obtain $\mathcal{F}(x, y)$ for a deterministic function $\mathcal{F}$. At the start of the protocol, both parties agree on a Boolean circuit that evaluates $\mathcal{F}$.

For symmetric security parameter κ, the garbler $\mathcal{G}$ starts by choosing two keys k_i^0 and k_i^1 of length κ bits for each wire w_i in the circuit, which represent the possible values 0 and 1. For a gate g_j, let l, r, and o denote the indices of the left input wire, right input wire, and output wire, respectively. $k_o^{g_j(x,y)}$ denotes the output key for gate j corresponding to the plaintext inputs x and y. Then $\mathcal{G}$ generates the garbled table

$$c_0 \leftarrow \mathtt{Enc}(k_l^0, \mathtt{Enc}(k_r^0, k_o^{g_j(0,0)})) \qquad c_1 \leftarrow \mathtt{Enc}(k_l^0, \mathtt{Enc}(k_r^1, k_o^{g_j(0,1)}))$$
$$c_2 \leftarrow \mathtt{Enc}(k_l^1, \mathtt{Enc}(k_r^0, k_o^{g_j(1,0)})) \qquad c_3 \leftarrow \mathtt{Enc}(k_l^1, \mathtt{Enc}(k_r^1, k_o^{g_j(1,1)}))$$

for each gate g_j in the circuit. Following this, $\mathcal{G}$ sends the garbled tables (permuted using a secret random permutation), called the garbled circuit $G(C)$, along with the keys corresponding to its input x to $\mathcal{E}$. That is, if its input bit on wire w_i is 1 it sends k_i^1, otherwise, it sends k_i^0. Next, $\mathcal{E}$ obliviously receives the keys corresponding to its inputs from $\mathcal{G}$ by executing an OT protocol. For every gate g_j, $\mathcal{E}$ knows two out of the four input keys, which allows to decrypt exactly one entry of the garbled table and yields the corresponding output key. After evaluating the circuit, $\mathcal{E}$ obtains the keys assigned to the labels of the output wires of the circuit. In the final step, $\mathcal{G}$ sends over a mapping from the circuit output keys to the actual bit values and $\mathcal{E}$ shares the result with $\mathcal{G}$.

In the description, it is required that $\mathcal{E}$ can decrypt exactly one entry from the garbled table per gate, which is ensured by the properties elusive and efficiently verifiable range, defined below, followed by the correctness of Yao' GC protocol.

Definition 2 (Elusive and Efficiently Verifiable Range [35]). *Let* E_S *be a secret key encryption scheme with algorithms* $(\texttt{Enc}, \texttt{Dec})$ *and define the range of a key as* $\mathsf{Range}_n(k) = \{\texttt{Enc}(k,m)\}_{m \in \{0,1\}^n}$*.*

1. *We say that* E_S *has an* elusive range*, if for any algorithm* $\mathcal{A}$ *it holds that* $\Pr[c \in \mathsf{Range}_n(k) \,|\, \mathcal{A}(1^n) \to c] \leq \mathrm{negl}(n)$*, probability taken over the keys*
2. *We say that* E_S *has an* efficiently verifiable range*, if there exists a probabilistic polynomial time machine* M *s.t.* $M(k,c) \to 1$ *if and only if* $c \in \mathsf{Range}_n(k)$*.*

Theorem 1 (Correctness of Yao's GC Protocol [35]). *We assume w.l.o.g. that* $x = x_1, \ldots, x_n$ *and* $y = y_1, \ldots, y_n$ *are two* n*-bit inputs for a Boolean circuit* C*. Let* $k_1, \ldots, k_n$ *be the labels of the circuit-input wires corresponding to* x*, and* $k_{n+1}, \ldots, k_{2n}$ *the labels of the circuit-input wires corresponding to* y*. Assume that the encryption scheme used to construct the garbled circuit* $G(C)$ *has an elusive and efficiently verifiable range. Then given* $G(C)$*, and the strings* $k_1^{x_1}, \ldots, k_n^{x_n}, k_{n+1}^{y_1}, \ldots, k_{2n}^{y_n}$*, it is possible to compute* $C(x,y)$*, except with negligible probability.*

3 Post-Quantum Secure Oblivious Transfer

Yao's protocol requires oblivious transfer (OT) for privately transferring the input labels from the garbler to the evaluator. In the following we give a PQ-secure construction for OT from AHE (cf. Section 3.1) and and prove OT extension post-quantum secure (cf. Section 3.2).

3.1 Post-Quantum Secure OT from AHE

We use a natural construction for a 1-out-of-2 OT protocol based on homomorphic encryption, that follows closely the design of the OT protocol from [1, Section 5], and works as follows:

1. The receiver encrypts its choice bit $c_b = \texttt{Enc}(pk, b)$ and sends it to the sender.

2. The sender complements the bit under encryption $c_{\bar{b}} = 1 \boxminus c_b$, computes $c_{m_b} = (m_0 \boxdot c_{\bar{b}}) \boxplus (m_1 \boxdot c_b)$, and sends it back to the receiver.
3. The receiver then decrypts the ciphertext to get $m_b = \mathtt{Dec}(sk, c_{m_b})$.

We instantiate it using the PQ-secure BFV homomorphic encryption scheme [23] in the implementation provided by Microsoft's SEAL library [52]. To substantially improve performance, we adapt this protocol to exploit the single instruction multiple data (SIMD) operations of the AHE scheme. Let the message length in the OT protocol be ℓ bits. In order to achieve maximum parallelism in the homomorphic operations of the AHE scheme (cf. Appendix A.2), we can choose a plaintext modulus p of more than ℓ bits, such that $p \equiv 1 \bmod x$, i.e., $d = \mathtt{ord}_{\mathbb{Z}_x^*}(p) = 1$. This choice of p provides the maximum number of slots (i.e., $n = \varphi(x)$) for a particular x. Then the receiver can encrypt n choice bits at once, and similarly the sender can pack n messages at once into a single plaintext, thereby performing n OTs at the cost of one.

However, for large ℓ such as $\ell = 2\kappa = 256$ bits for keys in PQ-Yao, having a plaintext modulus of more than 256 bits will lead to a very inefficient instantiation of the scheme. We would require a very large ciphertext modulus q to contain the noise, and consequently a very large n to maintain security. Although the number of slots will increase linearly with n, the complexity of the individual operations in the scheme will increase quasi-linearly as well, making the scheme operations very inefficient. Thus, we restrict our choice of p to less than 60 bits, as do the most popular libraries for HE [30, 52].

In order to pack large ℓ-bit messages with a plaintext modulus $p < 2^\ell$, where $\alpha = \lfloor \log_2(p) \rfloor$, we can use one of the following two approaches:

Span Multiple Slots. The first option is to have maximal slots ($n = \varphi(x)$ and $p \equiv 1 \bmod x$), and have the message packed across multiple slots. Given a message $m = (m_1 \parallel \ldots \parallel m_\beta) \in \{0,1\}^\ell$, where each component $m_i \in \{0,1\}^\alpha$, we can pack the message by storing its components in $\beta = \lceil \ell/\alpha \rceil$ different slots. Accordingly, the choice bit for that message is replicated in the corresponding slots. The mapping used is defined as follows:

$$\psi : \begin{cases} \{0,1\}^\ell & \longrightarrow (\{0,1\}^\alpha)^\beta \\ (m_1 \parallel \ldots \parallel m_\beta) & \longmapsto (m_i)_{i \in [\beta]} \end{cases} .$$

Using this approach, we can pack $\gamma = \lfloor n/\beta \rfloor$ messages into a single plaintext. The interface functions PackM, UnpackM, and PackB for this packing method are defined as follows:

$$\begin{aligned} \big(\psi\big(m_{\lfloor (i-1)/\beta \rfloor + 1}\big)_{(i-1) \bmod \beta + 1}\big)_{i \in [n]} &\leftarrow \mathtt{PackM}((m_i)_{i \in [\gamma]}) , \\ \big(\psi^{-1}\big((m_{(i-1)\cdot\beta + j})_{j \in [\beta]}\big)\big)_{i \in [\gamma]} &\leftarrow \mathtt{UnpackM}((m_i)_{i \in [n]}) , \\ (b_{\lfloor (i-1)/\beta \rfloor + 1})_{i \in [n]} &\leftarrow \mathtt{PackB}((b_i)_{i \in [\gamma]}) . \end{aligned}$$

Higher Degree Slots. Alternatively, instead of restricting ourselves to p of order 1, we consider p of higher order $\beta = d = \mathtt{ord}_{\mathbb{Z}_x^*}(p) \geq 1$. As a result, we can embed a polynomial of degree $\beta - 1$ in each slot, and use its higher order coefficients as well to pack a message. Hence, an $\ell = \alpha \cdot \beta$ bit message $m = (m_1 \parallel \ldots \parallel m_\beta)$, where $m_i \in \{0,1\}^\alpha$, can be packed in a single slot with the following mapping:

$$\omega : \begin{cases} \{0,1\}^\ell & \longrightarrow & \mathbb{F}_{p^\beta} \\ (m_1 \parallel \ldots \parallel m_\beta) & \longmapsto & m_1 + \ldots + m_\beta X^{\beta-1} \end{cases} .$$

Consequently, we can pack up to $\gamma = n = \varphi(x)/d$ messages of ℓ bits into a plaintext. The interface functions PackM, UnpackM, and PackB are defined as follows:

$$\begin{aligned} (\omega(m_i))_{i\in[n]} &\leftarrow \mathtt{PackM}((m_i)_{i\in[\gamma]}) , \\ (\omega^{-1}(\mathsf{m}_i))_{i\in[\gamma]} &\leftarrow \mathtt{UnpackM}((\mathsf{m}_i)_{i\in[n]}) , \\ (b_i)_{i\in[n]} &\leftarrow \mathtt{PackB}((b_i)_{i\in[\gamma]}) . \end{aligned}$$

The Final Protocol. The final OT protocol $\Pi^{\mathrm{OT}}_{\mathrm{AHE}}$ is described in Fig. 2. The protocol is divided into two phases, namely the setup phase and the OT phase. The setup phase is cheap ($\approx 20\,\mathrm{ms}$ in a LAN network, cf. Section 4.2) and needs to be performed only once between a set of parties. The OT phase runs on a batch of a maximum of γ inputs at a time. In practice, the OT phase can be iterated over (in parallel) with different batches of inputs to perform arbitrary number of OTs.

The protocol can be instantiated with either of the packing techniques. Note that both the techniques provide equal parallelism, which is $\gamma = \lfloor \varphi(x)/\beta \rfloor$ messages of ℓ bits per plaintext. An advantage of using the 'Span Multiple Slots' technique is that it is more flexible. It allows to double the message length ℓ without changing the scheme parameters by simply halving the batch size γ, and it is trivial to find the parameters for most efficient packing for larger values of ℓ. In the 'High Degree Slots' technique, x has to be chosen such that $\beta = \lceil \ell/\alpha \rceil$ is a divisor of $\varphi(x)$ for the most efficient packing, which makes the parameter selection very restrictive and non-trivial.

For smaller values, i.e., $\ell < \log_2 x$, it is not possible to get maximal slots. In such situations, using higher degree slots might be the better option. Thus, packing the message across multiple slots is more suitable for larger values of ℓ as in the case of Yao, and is the technique we have implemented in our benchmarks.

Theorem 2. *The $\Pi^{\mathrm{OT}}_{\mathrm{AHE}}$ protocol (cf. Fig. 2) securely performs γ OTs of length ℓ in the presence of semi-honest adversaries, providing computational security against a corrupted sender and statistical security against a corrupted receiver.*

The proof follows straightforwardly from the pq-IND-CPA security and the circuit privacy of the AHE scheme (cf. Appendix A.4), and can be found in Appendix B.1. We describe the parameter selection for the scheme in Appendix C.

3.2 Post-Quantum Secure Oblivious Transfer Extension

In this section we show that OT extension works also in the post-quantum setting. This concept has been introduced by Ishai et al. [32] and allows to obtain many

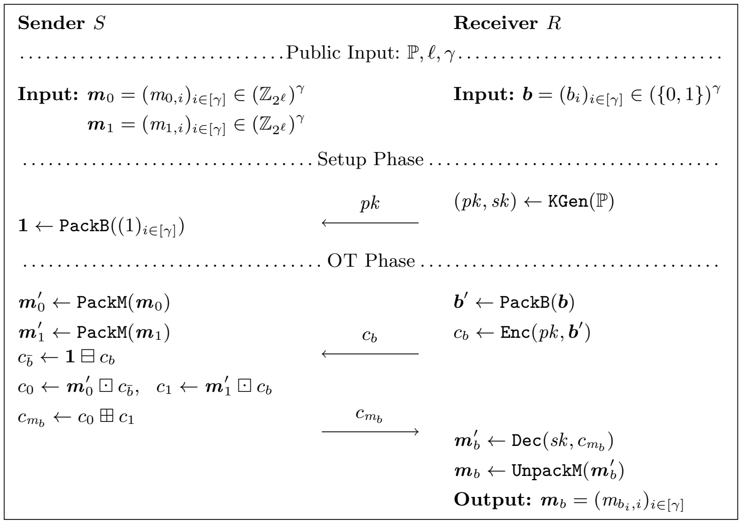
Sender S
Receiver R
Public Input: $\mathbb{P}, \ell, \gamma$
Input: $\boldsymbol{m}_0 = (m_{0,i})_{i\in[\gamma]} \in (\mathbb{Z}_{2^\ell})^\gamma$
$\boldsymbol{m}_1 = (m_{1,i})_{i\in[\gamma]} \in (\mathbb{Z}_{2^\ell})^\gamma$
Input: $\boldsymbol{b} = (b_i)_{i\in[\gamma]} \in (\{0,1\})^\gamma$
Setup Phase
$(pk, sk) \leftarrow \mathtt{KGen}(\mathbb{P})$
pk
$\mathbf{1} \leftarrow \mathtt{PackB}((1)_{i\in[\gamma]})$
OT Phase
$\boldsymbol{m}'_0 \leftarrow \mathtt{PackM}(\boldsymbol{m}_0)$
$\boldsymbol{m}'_1 \leftarrow \mathtt{PackM}(\boldsymbol{m}_1)$
$\boldsymbol{b}' \leftarrow \mathtt{PackB}(\boldsymbol{b})$
$c_b \leftarrow \mathtt{Enc}(pk, \boldsymbol{b}')$
c_b
$c_{\bar{b}} \leftarrow \mathbf{1} \boxminus c_b$
$c_0 \leftarrow \boldsymbol{m}'_0 \boxdot c_{\bar{b}}, \quad c_1 \leftarrow \boldsymbol{m}'_1 \boxdot c_b$
$c_{m_b} \leftarrow c_0 \boxplus c_1$
c_{m_b}
$\boldsymbol{m}'_b \leftarrow \mathtt{Dec}(sk, c_{m_b})$
$\boldsymbol{m}_b \leftarrow \mathtt{UnpackM}(\boldsymbol{m}'_b)$
Output: $\boldsymbol{m}_b = (m_{b_i,i})_{i\in[\gamma]}$

Fig. 2. Ring-LWE based OT protocol $\Pi_{\mathrm{AHE}}^{\mathrm{OT}}$.

OTs using only a few actual OTs as base OTs and fast symmetric cryptographic operations for each OT. As Yao's GC protocol requires an OT for every bit of the evaluator's input, OT extension can be used to improve performance of Yao's GC protocol with many evaluator inputs. OT extension makes use of random oracles. As described in Section 2, this entails that the post-quantum security proof has to be conducted in the QROM instead of the ROM.

Our result is of interest even beyond Yao's protocol for other applications that use many OTs and could be proven to be post-quantum secure in future work, e.g., the GMW protocol [27] or Private Set Intersection [46, 48, 49].

Input of S: τ pairs $(x_{i,0}, x_{i,1})$ of l-bit strings, $1 \le i \le \tau$

Input of R: τ selection bits $\boldsymbol{r} = (r_1, \dots, r_\tau)$

Common Input: a security parameter κ

Oracle: a random oracle $\mathsf{H}\colon [\tau] \times \{0,1\}^\kappa \to \{0,1\}^l$

Cryptographic Primitive: An ideal OT primitive

1. S initializes a random vector $\boldsymbol{s} \leftarrow_\$ \{0,1\}^\kappa$ and R a random matrix $\mathbf{T} \leftarrow_\$ \{0,1\}^{\tau\times\kappa}$
2. The parties invoke the OT primitive, where S acts as the receiver with input $\boldsymbol{s}$ and R acts as the sender with input $(\boldsymbol{t}^i, \boldsymbol{r} \oplus \boldsymbol{t}^i)$, $1 \le i \le \kappa$
3. Let $\mathbf{Q}$ denote the matrix of values received by S. Note that $\boldsymbol{q}_j = (r_j \boldsymbol{s}) \oplus \boldsymbol{t}_j$. For $1 \le j \le \tau$, S sends $(y_{j,0}, y_{j,1})$ where $y_{j,0} \leftarrow x_{j,0} \oplus \mathsf{H}(j, \boldsymbol{q}_j)$ and $y_{j,1} \leftarrow x_{j,1} \oplus \mathsf{H}(j, \boldsymbol{q}_j \oplus \boldsymbol{s})$.
4. For $1 \le j \le \tau$, R outputs $z_j \leftarrow y_{j,r_j} \oplus \mathsf{H}(j, \boldsymbol{t}_j)$.

Fig. 3. OT extension protocol from [32].

In the following theorem, we show that OT extension [32] is post-quantum secure. The full proof is given in Appendix B.2.

Theorem 3. *The OT extension protocol from [32] shown in Fig. 3 is post-quantum secure against malicious sender and semi-honest receiver in the quantum random oracle model.*

To instantiate post-quantum secure OT extension, it is sufficient to double the security parameter by doubling the output length of the hash function, using SHA-512 instead of SHA-256. This corresponds to the speed-up achieved by Grover's algorithm [28]. Hence, for PQ-security of OT extension the security parameter κ is set to 256 instead of 128 in the classical setting. This is in line with the recommendations provided at `https://keylength.com`.

4 Implementation and Performance Evaluation

In this section we describe our concrete instantiation and implementation of the PQ-secure protocols that we described in the previous sections. We benchmarked all implementations on two identical machines using an Intel Core i9-7960X CPU with 2.80 GHz and 128 GiB RAM. We compare the performance in a (simulated) WAN network (100 Mbit/s, 100 ms round trip time) and a LAN network (10 Gbit/s, 0.2 ms round trip time). All benchmarks run with a single thread. We instantiate all primitives to achieve the equivalent of 128-bit classical security.

4.1 Post-Quantum Yao Implementation and Performance

We used the code of the EMP toolkit [57, 58] as foundation for our implementation and comparison. We compare 3 variants of Yao's protocol in order to assess the impact of post-quantum security on the concrete efficiency (cf. Table 1 for an overview):

1. PQ: a post-quantum version of Yao's protocol with $2\kappa = 256$ bit wire labels. For obliviously transferring the evaluator's input labels, we use our PQ-OT protocol from Section 3. Garbling is done using the wire labels as keys for AES-256 as follows:

$$\begin{aligned}\mathsf{table}[e] &= \mathtt{Enc}(k_l, \mathtt{Enc}(k_r, k_o)) \\ &= k_o \oplus (\mathtt{Enc}^{\text{AES-256}}(k_l, T \parallel 0 \parallel 0) \parallel \mathtt{Enc}^{\text{AES-256}}(k_l, T \parallel 0 \parallel 1)) \\ &\quad \oplus (\mathtt{Enc}^{\text{AES-256}}(k_r, T \parallel 1 \parallel 0) \parallel \mathtt{Enc}^{\text{AES-256}}(k_r, T \parallel 1 \parallel 1)),\end{aligned}$$

where k_o is the output label of gate with ID j, k_l is its left input label, k_r its right input label, and $T = j \parallel e$ is the tweak. We use the point-and-permute optimization [9, 38], which reduces the number of decryptions per gate to a single one by appending a random signal bit to every label. This approach merely prevents decryption of the wrong entries in the garbled table. Since the signal bits are chosen at random, it has clearly no effect on the security of the scheme itself, which makes it a suitable optimization also in the post-quantum setting.

2. C: an implementation of the classical Yao's protocol with the same instantiations as PQ, but using $\kappa = 128$-bit wire labels and AES-128. Specifically, garbling is done as follows in this implementation:

$$\begin{aligned}\mathsf{table}[e] &= \mathtt{Enc}(k_l, \mathtt{Enc}(k_r, k_o)) \\ &= k_o \oplus \mathtt{Enc}^{\text{AES-128}}(k_l, T \parallel 0) \oplus \mathtt{Enc}^{\text{AES-128}}(k_r, T \parallel \ 1)\,.\end{aligned}$$

3. EMP: the original EMP implementation [58] of the classical Yao's protocol with state-of-the-art optimizations: free-XOR [34], fixed-key AES-128 garbling [10], and half-gates [60] on $\kappa = 128$-bit wire labels.

Table 1. Overview of our implementations and the used parameters and optimizations.

	PQ	C	EMP [58]
PQ-Secure	✓	✗	✗
OT	PQ-OT (Section 3)	OT extension [32]	OT extension [32]
Point&Permute [9, 38]	✓	✓	✓
Free-XOR [34]	✗	✗	✓
Half-Gates [60]	✗	✗	✓
Garbling	Variable-Key AES-256	Variable-Key AES-128	Fixed-Key AES-128 [10]

The circuits we benchmarked are described in Table 2.

Table 2. Boolean Circuits used to benchmark Yao's protocol in Section 4.

Circuit	Description	Garbler Inputs	Evaluator Inputs	ANDs	XORs	NOTs
aes	AES-128	128	128	6800	25124	1692
add	32-bit Adder	32	32	127	61	187
mult	32x32-bit Multiplier	32	32	5926	1069	5379

The benchmark results are given in Table 3 for a LAN connection and in Table 4 for a WAN connection. As the implementation of the EMP toolkit uses pipelining and interleaves circuit garbling and evaluation, we only report the time until the circuit evaluation finishes, which includes the circuit garbling. We note that this time is marginally larger than the sole garbling time, i.e., the garbling time makes up almost all of the reported total evaluation time.

The runtime of PQ-Yao is on average 1.5× and 2× greater than the runtime of classical unoptimized Yao in the LAN and the WAN setting, respectively. The performance difference gets more prominent in the WAN setting, because PQ-Yao requires twice as much communication as the classical unoptimized version due to the doubled length of the wire labels. Nevertheless, even the 2× slowdown is reasonable for achieving PQ security. The difference in the runtime and communication for the input sharing phase stems from the cost of the PQ-OT. For a batch of 1,000 parallel 32-bit multiplications, our PQ-Yao implementation

Table 3. Performance comparison of our PQ-Yao protocol, with a classical unoptimized Yao protocol (C), and the classical optimized EMP version [58] in a LAN network.

		Input Sharing						Garbling & Evaluation					
		Runtime [s]			Comm. [MiB]			Runtime [s]			Comm. [MiB]		
Circ.	Batch	PQ	C	EMP	PQ	C	EMP	PQ	C	EMP	PQ	C	EMP
aes	1	0.05	0.03	0.02	0.6	0.3	0.3	0.05	0.03	0.01	3.9	1.9	0.2
aes	10	0.06	0.02	0.02	1.4	0.3	0.3	0.15	0.13	0.04	39.0	19.5	2.1
aes	100	0.22	0.04	0.03	10.0	0.9	0.5	1.01	0.65	0.09	389.7	194.8	20.8
aes	1,000	1.67	0.13	0.10	97.9	7.9	4.0	9.75	6.36	0.82	3,897.0	1,948.5	207.5
add	1	0.05	0.03	0.02	0.6	0.3	0.3	0.00	0.00	0.00	0.0	0.0	0.0
add	10	0.05	0.02	0.02	0.6	0.3	0.3	0.01	0.01	0.00	0.2	0.1	0.0
add	100	0.10	0.03	0.03	3.0	0.4	0.3	0.04	0.03	0.01	2.3	1.1	0.4
add	1,000	0.62	0.07	0.05	24.9	2.0	1.0	0.11	0.07	0.05	22.9	11.5	3.9
mult	1	0.05	0.02	0.02	0.6	0.3	0.3	0.03	0.02	0.01	0.9	0.4	0.2
mult	10	0.05	0.03	0.02	0.6	0.3	0.3	0.07	0.05	0.04	8.5	4.3	1.8
mult	100	0.10	0.02	0.03	3.0	0.4	0.3	0.26	0.17	0.08	85.4	42.7	18.1
mult	1,000	0.44	0.06	0.04	24.9	2.0	1.0	2.19	1.48	0.38	853.9	426.9	180.8

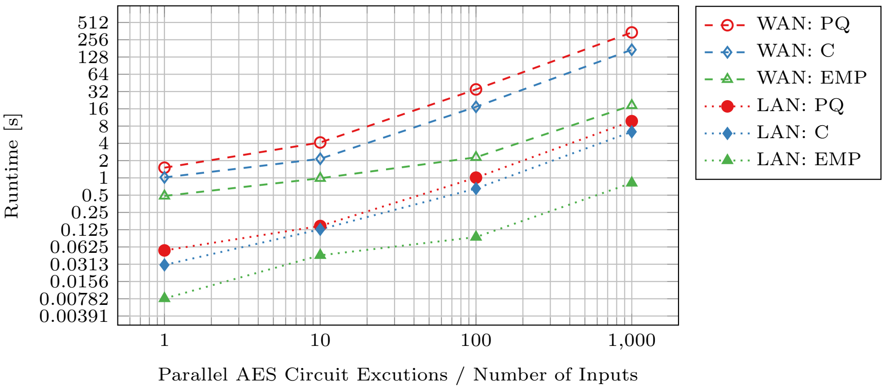

Fig. 4. Comparison of implementations of our PQ-Yao, with the classical, unoptimized Yao protocol (C), and the classical, optimized EMP version in a LAN and WAN network. Evaluation time for parallel executions of an AES circuit.

performs 2.7M (88k) gates/s, while a classical unoptimized Yao version achieves 4.8M (179k) gates/s; the fully optimized classical implementation can perform 16.8M (404k) gates/s in the LAN (WAN) setting. This accounts only for AND and XOR gates, since NOT gates can be evaluated for free in all three versions.

In Fig. 4, we plot the evaluation time (including garbling time) of parallel AES circuits evaluated with the three versions of Yao's protocol for different batch sizes and show that it scales linearly.

We could not evaluate the concrete performance of the implementation of [29], since their code is not publicly available. Based on experimental results in [29], we expect the performance to be similar to that of the optimized, classical implementation using all state-of-the-art optimizations (EMP).

Table 4. Performance comparison of our PQ-Yao protocol, with a classical unoptimized Yao protocol (C), and the classical optimized EMP version [58] in a WAN network.

		Input Sharing						Garbling & Evaluation					
		Runtime [s]			Comm. [MiB]			Runtime [s]			Comm. [MiB]		
Circ.	Batch	PQ	C	EMP	PQ	C	EMP	PQ	C	EMP	PQ	C	EMP
aes	1	1.40	0.81	0.81	0.6	0.3	0.3	1.51	1.02	0.48	3.9	1.9	0.2
aes	10	1.73	0.92	0.90	1.4	0.3	0.3	4.14	2.15	0.99	39.0	19.5	2.1
aes	100	2.83	1.22	1.12	10.0	0.9	0.5	34.85	17.33	2.28	389.7	194.8	20.8
aes	1,000	13.05	2.57	2.04	97.9	7.9	4.0	342.91	171.25	18.32	3,897.0	1,948.5	207.5
add	1	1.03	0.71	0.61	0.6	0.3	0.3	0.20	0.11	0.10	0.0	0.0	0.0
add	10	1.22	0.72	0.61	0.6	0.3	0.3	0.90	0.50	0.21	0.2	0.1	0.0
add	100	2.44	1.10	0.80	3.0	0.4	0.3	1.87	0.90	0.31	2.3	1.1	0.4
add	1,000	4.07	1.51	1.20	24.9	2.0	1.0	2.79	1.50	0.63	22.9	11.5	3.9
mult	1	1.02	0.71	0.61	0.6	0.3	0.3	0.68	0.52	0.41	0.9	0.4	0.2
mult	10	1.02	0.71	0.61	0.6	0.3	0.3	1.67	1.10	0.80	8.5	4.3	1.8
mult	100	2.27	1.10	0.80	3.0	0.4	0.3	8.13	4.12	2.12	85.4	42.7	18.1
mult	1,000	4.03	1.51	1.20	24.9	2.0	1.0	75.68	37.60	16.14	853.9	426.9	180.8

4.2 Post-Quantum OT Implementation and Performance

We implement our PQ-OT protocol from Section 3 using the Microsoft SEAL library [52]. We use the implementation from the EMP toolkit [58] for the classical OTs. In our experiments, we compare the following three 1-out-of-2 OT protocols:

- PQ: our implementation of PQ-OT on 256-bit inputs (cf. Section 3).
- NP: classical Naor-Pinkas (NP)-OT [42] on 128-bit inputs, from EMP.
- OTe: classical semi-honest OT extension of [32] on 128-bit inputs, from the implementation in EMP. It uses NP-OT [42] to perform the base OTs.

We provide performance results for running batches of N OTs in Table 5.

It is evident from the benchmarks that computation is the bottleneck for NP-OT, while communication is the bottleneck for both PQ-OT and OT extension. The network setting affects PQ-OT significantly, but not as much as it affects OT extension, since OT extension is computationally very efficient.

Table 5. 1-out-of-2 OT measured in a LAN and WAN network, comparing our PQ-OT on 256-bit inputs (cf. Section 3) with the classical Naor-Pinkas (NP)-OT [42] and classical OT extension (OTe) implementation on 128-bit inputs from the EMP toolkit.

	Setup Phase						Online Phase								
	Runtime [s]				Comm. [KiB]		Runtime [s]						Comm. [KiB]		
	LAN		WAN				LAN			WAN					
#OTs	PQ	OTe	PQ	OTe	PQ	OTe	PQ	NP	OTe	PQ	NP	OTe	PQ	NP	OTe
2^0	0.03	0.04	0.5	0.15	256	21.3	0.04	0.03	0.01	0.7	0.2	0.4	384	0	256
2^2	0.02	0.03	0.5	0.15	256	21.3	0.04	0.03	0.01	0.7	0.2	0.4	384	1	256
2^4	0.02	0.03	0.5	0.14	256	21.3	0.04	0.03	0.01	0.7	0.2	0.4	384	3	257
2^6	0.02	0.04	0.5	0.15	256	21.3	0.04	0.03	0.01	0.7	0.2	0.4	384	11	258
2^8	0.02	0.03	0.5	0.14	256	21.3	0.04	0.05	0.01	0.7	0.4	0.4	384	43	264
2^{10}	0.02	0.03	0.5	0.15	256	21.3	0.05	0.12	0.01	1.2	0.7	0.5	768	170	288
2^{12}	0.03	0.04	0.5	0.15	256	21.3	0.10	0.29	0.02	2.0	2.0	0.7	3,073	680	384
2^{14}	0.02	0.03	0.5	0.15	256	21.3	0.26	1.23	0.03	2.4	3.3	0.9	12,293	2,720	768
2^{16}	0.02	0.03	0.5	0.15	256	21.3	0.87	5.55	0.07	5.0	6.4	1.3	49,173	10,880	3,072
2^{18}	0.02	0.03	0.5	0.15	256	21.3	3.07	22.85	0.12	17.7	22.6	2.8	196,690	43,520	12,288
2^{20}	0.02	0.03	0.5	0.14	256	21.3	11.77	91.38	0.18	68.6	91.3	5.3	786,760	174,080	49,152

Comparison with PK-based OT. PQ-OT provides better performance than NP-OT for most practical cases ($N \geq 2^8$) in the LAN setting. It reaches a maximum throughput of $\approx$ 89k OT/s for $N = 2^{20}$, while NP-OT only reaches a maximum of $\approx$ 14k OT/s for $N = 2^{12}$. In the WAN setting, PQ-OT outperforms NP-OT for $N \geq 2^{12}$ OTs. We also compared PQ-OT with an instantiation of the OT construction by Gertner et al. [26] with Kyber-1024 (AVX2 optimized 90s variant) [51] and found it to be less efficient than our scheme, achieving a maximum throughput of 50k OT/s, even though Kyber is already among the fastest PKE schemes in the NIST standardization process. Therefore, we do not expect this situation to change significantly with other instantiations. Even for smaller number of OTs, the performance between the two is comparable in the WAN setting, even though with PQ-OT we achieve PQ security and are dealing with inputs that are twice as long. For $N = 2^8$ in the WAN setting, the throughput of NP-OT is 640 OT/s, while the throughput of PQ-OT is 365 OT/s. While NP-OT does not have a setup phase, PQ-OT requires to share a public key in the setup phase. It is negligible in the LAN setting and dominated by the communication in the WAN setting. It is relatively expensive for a small number of OTs, but only needs to be run once with a particular party, independently of the inputs. Thus, PQ-OT is a suitable candidate to replace NP-OT as the protocol for base OT in the post-quantum setting at $\approx 4.5\times$ the communication cost of NP-OT for large batch sizes. On the one hand, we show that our implementation of PQ-OT achieves similar performance compared to NP-OT for a small number of OTs, which is common for Yao's protocol with a moderate number of client input bits. On the other hand, our implementation clearly outperforms classical NP-OT for larger batches, especially in fast networks.

Comparison with OT extension. OT extension outperforms the two public-key based OT protocols, in both computation and communication, for practical number of OTs, reaching a maximum throughput of $\approx$ 5.7M (199k) OT/s in the LAN (WAN) setting. The runtime and communication not growing linearly for $N \leq 2^{14}$ OTs is an artefact of the EMP implementation of OT extension. While there is approximately one order of magnitude difference between classical OT extension and our PQ-OT, there is room for significant improvement by implementing post-quantum secure OT extension, as described in Section 3.2, which we leave as future work.

5 Post-Quantum Security of Yao's Garbled Circuits

In this section, we prove that Yao's garbled circuits protocol (cf. Section 2.4) achieves post-quantum security if each of the underlying building blocks is replaced with a post-quantum secure variant. As this seems intuitive, we stress that a simple switch to post-quantum secure building blocks is not always sufficient [24]. An example for this is the Fiat-Shamir transformation. Simply constructing a signature scheme based on a quantum hard problem is not sufficient, due to the switch from the ROM to the QROM. For the signature scheme qTESLA [2], for instance, the post-quantum security has been proven directly.

The classical security of Yao's protocol is due to Lindell and Pinkas [35]. They showed that a secure OT protocol and a secret key encryption scheme which is secure under double encryption (a security notion they introduced) are sufficient to prove Yao's protocol secure against semi-honest adversaries. Concerning the security under double encryption, they show that, classically, IND-CPA security implies security under double encryption. We show that both proofs can be lifted against quantum adversaries. Regarding the proof for the protocol, this is relatively straightforward, by arguing about the different steps from the classical proof. As for the security under double encryption, we directly prove the post-quantum security since the classical proof is merely sketched. Furthermore, we conduct the proof in the QROM whereas the classical proof sketch does not consider random oracles. This is relevant when one wants to use encryption scheme where the proof is naturally in the QROM, like sponge-based constructions. Examples for this are the encryption schemes deployed in Isap [21] and Slae [19].[7]

Protocol Security. In this section, we prove that Yao's protocol is post-quantum secure against semi-honest quantum adversaries. In this setting, the adversary can perform local quantum computations and tries to obtain additional information while genuinely running the protocol.

The restriction to local quantum computations is due to the post-quantum setting, in which only the adversary has quantum power while all other parties, in this case the protocol partner, remain classical. By restricting the adversary to be semi-honest, we ensure that it does not deviate from the protocol specification. This models a typical scenario of an adversary which tries to obtain additional

[7] Note, however, that both schemes have yet to be proven post-quantum secure.

information without being noticed by the other party. One can think of a computer virus affecting one of the protocol participants, which tries to be unnoticed.

The theorem below states the post-quantum security of Yao's GC protocol given that both the OT and the encryption scheme are post-quantum secure. The proof appears in Appendix B.3.

Theorem 4 (Post-Quantum Security of Yao's GC Protocol). *Let $\mathcal{F}$ be a deterministic function. Suppose that the oblivious transfer protocol is post-quantum secure against semi-honest adversaries, the encryption scheme is* pq-2Enc-*secure*[8], *and the encryption scheme has an elusive and efficiently verifiable range. Then the protocol described in Section 2.4 securely computes $\mathcal{F}$ in the presence of semi-honest quantum adversaries.*

Double Encryption Security. To securely instantiate Yao's protocol, an encryption scheme which is secure under double encryption is required. In the classical setting, Lindell and Pinkas [35] provide a short sketch that the standard security notion for encryption schemes (IND-CPA) implies security under double encryption. In this section, we show that the same argument holds in the post-quantum setting, i.e., pq-IND-CPA security implies post-quantum security under double encryption (pq-2Enc). Furthermore, we extend the result to the QROM. This allows to cover a wider class of encryption schemes compared to the proof sketch from [35] which does not consider random oracles.

We start by introducing the post-quantum variant of the double encryption security game in the QROM (cf. Fig. 5). Similar to the $\mathsf{pq\text{-}INDCPA}$ game (cf. Fig. 1), the adversary has to distinguish between the encryption of messages of its choice. The main difference is that there are four secret keys involved in the game, from which two are given to the adversary. As challenge messages, the adversary provides three pairs of messages. For each pair, one message is encrypted twice using two different keys from which at least one is unknown to the adversary. The adversary wins the game if it can distinguish which messages have been encrypted. The adversary is granted access to two *learning* oracles which encrypt messages under a combination of a key given by the adversary and one of the unknown keys. There are two differences between our notion and the (classical) one given in [35]. First, we allow for multiple challenge queries from the adversary while [35] allow merely one. Second, the two known keys are honestly generated by the challenger and then handed over to the adversary. In [35], the adversary chooses these keys by itself. Since these keys correspond to the keys that the garbler generates honestly and obliviously sends to the evaluator, this change in the security notion models the actual scenario very well. In fact, the proof of Yao's protocol only requires the adversary to know two of the keys but not being able to generate them at will.

Definition 3 (Post-Quantum Security under Double Encryption). *Let* $\mathrm{E}_S = (\mathtt{Enc}, \mathtt{Dec})$ *be a secret key encryption scheme and let the game* $\mathtt{pq2enc}$ *be*

[8] We formally define post-quantum security under double encryption (pq-2Enc security) in Definition 3.

defined as in Fig. 5. Then for any quantum adversary $\mathcal{A}$ its advantage against the double encryption security is defined as:

$$\mathbf{Adv}^{\mathsf{pq2enc}}_{\mathrm{E}_S}(\mathcal{A}) = 2\Pr\left[\mathsf{pq2enc}^{\mathcal{A}} \to \mathrm{true}\right] - 1\,.$$

We say that E_S *is* pq-2Enc-*secure if* $\mathbf{Adv}^{\mathsf{pq2enc}}_{\mathrm{E}_S}(\mathcal{A})$ *is negligible.*

```
Game pq2enc
procedure Initialize
    $b \leftarrow_\$ \{0,1\}$
    $k_0, k_1, k_0', k_1' \leftarrow_\$ \mathcal{K}$
return $k_0, k_1$

procedure $\overline{\mathsf{Enc}_0}(k, m)$
    $c \leftarrow \mathtt{Enc}(k_0', \mathtt{Enc}(k, m))$
    return $c$
```

```
procedure $\overline{\mathsf{Enc}_1}(k, m)$
    $c \leftarrow \mathtt{Enc}(k, \mathtt{Enc}(k_1', m))$
    return $c$

procedure Finalize $(b')$
    return $(b' = b)$

procedure $\mathsf{O}_{\mathsf{H}}(\sum \alpha_{x,y} |x, y\rangle)$
    return $\sum \alpha_{x,y} |x, y \oplus \mathsf{H}(x)\rangle$
```

```
procedure $\overline{\mathsf{E}}(m_0, m_1)$
    parse $m_0$ as $x_0 \parallel y_0 \parallel z_0$
    parse $m_1$ as $x_1 \parallel y_1 \parallel z_1$
    $c_1 \leftarrow \mathtt{Enc}(k_0, \mathtt{Enc}(k_1', x_b))$
    $c_2 \leftarrow \mathtt{Enc}(k_0', \mathtt{Enc}(k_1, y_b))$
    $c_3 \leftarrow \mathtt{Enc}(k_0', \mathtt{Enc}(k_1', z_b))$
    $c \leftarrow (c_1, c_2, c_3)$
    return $c$
```

Fig. 5. Game pq2enc to define post-quantum security under double encryption.

The theorem below states that pq-IND-CPA security implies pq-2Enc security. The proof is given in Appendix B.4.

Theorem 5. *Let* $\mathrm{E}_S = (\mathtt{Enc}, \mathtt{Dec})$ *be a secret key encryption scheme. Then for any quantum adversary* $\mathcal{A}$ *against the post-quantum security under double encryption security of* E_S*, there exists a quantum adversary* $\overline{\mathcal{A}}$ *against the* pq-IND-CPA *security of* E_S *such that:*

$$\mathbf{Adv}^{\mathsf{pq2enc}}_{\mathrm{E}_S}(\mathcal{A}) \leq 3\,\mathbf{Adv}^{\mathrm{pq\text{-}ind\text{-}cpa}}_{\mathrm{E}_S}(\overline{\mathcal{A}})\,.$$

Acknowledgements

This work was co-funded by the Deutsche Forschungsgemeinschaft (DFG) — SFB 1119 CROSSING/236615297 and GRK 2050 Privacy & Trust/251805230, by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within ATHENE, and by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 850990 PSOTI).

References

1. W. Aiello, Y. Ishai, O. Reingold: Priced Oblivious Transfer: How to Sell Digital Goods. In: EUROCRYPT 2001. LNCS, pp. 119–135. Springer (2001)
2. E. Alkim, N. Bindel, J.A. Buchmann, Ö. Dagdelen, E. Eaton, G. Gutoski, J. Krämer, F. Pawlega: Revisiting TESLA in the Quantum Random Oracle Model. In: Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, pp. 143–162. Springer (2017)

3. E. Alkim, L. Ducas, T. Pöppelmann, P. Schwabe: Post-quantum Key Exchange - A New Hope. In: USENIX Security 2016, pp. 327–343. USENIX Association (2016)
4. J.B. Almeida, M. Barbosa, G. Barthe, F. Dupressoir, B. Grégoire, V. Laporte, V. Pereira: A Fast and Verified Software Stack for Secure Function Evaluation. In: ACM CCS 2017, pp. 1989–2006. ACM Press (2017)
5. A. Ambainis, M. Hamburg, D. Unruh: Quantum Security Proofs Using Semi-classical Oracles. In: CRYPTO 2019, pp. 269–295 (2019)
6. F. Arute, K. Arya, R. Babbush, D. Bacon, J.C. Bardin, R. Barends, R. Biswas, S. Boixo, F.G. Brandao, D.A. Buell: Quantum supremacy using a programmable superconducting processor. Nature 574(7779), 505–510 (2019)
7. G. Asharov, Y. Lindell, T. Schneider, M. Zohner: More Efficient Oblivious Transfer Extensions. Journal of Cryptology 30(3), 805–858 (2017)
8. B. Bauer, D. Wecker, A.J. Millis, M.B. Hastings, M. Troyer: Hybrid quantum-classical approach to correlated materials. (2015)
9. D. Beaver, S. Micali, P. Rogaway: The Round Complexity of Secure Protocols (Extended Abstract). In: 22nd ACM STOC, pp. 503–513. ACM Press (1990)
10. M. Bellare, V.T. Hoang, S. Keelveedhi, P. Rogaway: Efficient Garbling from a Fixed-Key Blockcipher. In: 2013 IEEE Symposium on Security and Privacy, pp. 478–492. IEEE Computer Society Press (2013)
11. M. Bellare, P. Rogaway: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: ACM CCS 93, pp. 62–73. ACM Press (1993)
12. M. Bellare, P. Rogaway: The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In: EUROCRYPT 2006. LNCS, pp. 409–426. Springer (2006)
13. D.J. Bernstein, J. Buchmann, E. Dahmen: Post-Quantum Cryptography. Springer (2009)
14. D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner, M. Zhandry: Random Oracles in a Quantum World. In: ASIACRYPT 2011. LNCS, pp. 41–69. Springer (2011)
15. Z. Brakerski, N. Döttling: Two-Message Statistically Sender-Private OT from LWE. In: TCC 2018, Part II. LNCS, pp. 370–390. Springer (2018)
16. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, G. Maxwell: Bulletproofs: Short Proofs for Confidential Transactions and More. In: 2018 IEEE Symposium on Security and Privacy, pp. 315–334. IEEE Computer Society Press (2018)
17. R. Canetti: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: 42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA, pp. 136–145. IEEE Computer Society (2001)
18. S.G. Choi, J. Katz, R. Kumaresan, H.-S. Zhou: On the Security of the "Free-XOR" Technique. In: TCC 2012. LNCS, pp. 39–53. Springer (2012)
19. J.P. Degabriele, C. Janson, P. Struck: Sponges Resist Leakage: The Case of Authenticated Encryption. In: ASIACRYPT 2019, Part II. LNCS, pp. 209–240. Springer (2019)
20. J. Ding, D. Schmidt: Rainbow, a New Multivariable Polynomial Signature Scheme. In: ACNS 05. LNCS, pp. 164–175. Springer (2005)
21. C. Dobraunig, M. Eichlseder, S. Mangard, F. Mendel, T. Unterluggauer: ISAP – Towards Side-Channel Secure Authenticated Encryption. IACR Trans. Symm. Cryptol. 2017(1), 80–105 (2017)

22. R. Dowsley, J. van de Graaf, J. Müller-Quade, A.C.A. Nascimento: Oblivious Transfer Based on the McEliece Assumptions. In: ICITS 08. LNCS, pp. 107–117. Springer (2008)
23. J. Fan, F. Vercauteren: *Somewhat Practical Fully Homomorphic Encryption*, Cryptology ePrint Archive, Report 2012/144 (2012). http://eprint.iacr.org/2012/144. 2012.
24. T. Gagliardoni: Quantum Security of Cryptographic Primitives. Darmstadt University of Technology, Germany (2017).
25. C. Gentry: Fully homomorphic encryption using ideal lattices. In: 41st ACM STOC, pp. 169–178. ACM Press (2009)
26. Y. Gertner, S. Kannan, T. Malkin, O. Reingold, M. Viswanathan: The Relationship between Public Key Encryption and Oblivious Transfer. In: 41st FOCS, pp. 325–335. IEEE Computer Society Press (2000)
27. O. Goldreich, S. Micali, A. Wigderson: How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In: 19th ACM STOC, pp. 218–229. ACM Press (1987)
28. L.K. Grover: A Fast Quantum Mechanical Algorithm for Database Search. In: 28th ACM STOC, pp. 212–219. ACM Press (1996)
29. S. Gueron, Y. Lindell, A. Nof, B. Pinkas: Fast Garbling of Circuits Under Standard Assumptions. Journal of Cryptology 31(3), 798–844 (2018)
30. S. Halevi, V. Shoup: HElib-An Implementation of homomorphic encryption. Cryptology ePrint Archive, Report 2014/039 (2014)
31. R. Impagliazzo, S. Rudich: Limits on the Provable Consequences of One-Way Permutations. In: 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA, pp. 44–61 (1989)
32. Y. Ishai, J. Kilian, K. Nissim, E. Petrank: Extending Oblivious Transfers Efficiently. In: CRYPTO 2003. LNCS, pp. 145–161. Springer (2003)
33. D. Jao, L. De Feo: Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies. In: Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, pp. 19–34. Springer (2011)
34. V. Kolesnikov, T. Schneider: Improved Garbled Circuit: Free XOR Gates and Applications. In: ICALP'08, pp. 486–498 (2008)
35. Y. Lindell, B. Pinkas: A Proof of Security of Yao's Protocol for Two-Party Computation. Journal of Cryptology 22(2), 161–188 (2009)
36. Y. Lindell, B. Pinkas: An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries. In: EUROCRYPT'07, pp. 52–78 (2007)
37. V. Lyubashevsky, C. Peikert, O. Regev: On Ideal Lattices and Learning with Errors over Rings. In: EUROCRYPT 2010. LNCS, pp. 1–23. Springer (2010)
38. D. Malkhi, N. Nisan, B. Pinkas, Y. Sella: Fairplay - Secure Two-Party Computation System. In: USENIX Security 2004, pp. 287–302. USENIX Association (2004)
39. D. Masny, P. Rindal: Endemic Oblivious Transfer. In: ACM CCS 2019, pp. 309–326. ACM Press (2019)
40. R.J. McEliece: A Public-Key Cryptosystem Based On Algebraic Coding Theory. DSN Progress Report (1978)
41. M. Mosca: Cybersecurity in an Era with Quantum Computers: Will We Be Ready? IEEE Security & Privacy 16(5), 38–41 (2018)
42. M. Naor, B. Pinkas: Efficient Oblivious Transfer Protocols. In: 12th SODA, pp. 448–457. ACM-SIAM (2001)
43. M. Naor, B. Pinkas, R. Sumner: Privacy preserving auctions and mechanism design. In: ACM Conference on Electronic Commerce, pp. 129–139 (1999)

44. M.A. Nielsen, I.L. Chuang: Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press (2011)
45. NIST: *PQ Cryptography Standardization Process*, (2017). 2017.
46. B. Pinkas, T. Schneider, G. Segev, M. Zohner: Phasing: Private Set Intersection Using Permutation-based Hashing. In: USENIX Security 2015, pp. 515–530. USENIX Association (2015)
47. B. Pinkas, T. Schneider, N.P. Smart, S.C. Williams: Secure Two-Party Computation Is Practical. In: ASIACRYPT'09, pp. 250–267 (2009)
48. B. Pinkas, T. Schneider, M. Zohner: Faster Private Set Intersection Based on OT Extension. In: USENIX Security 2014, pp. 797–812. USENIX Association (2014)
49. B. Pinkas, T. Schneider, M. Zohner: Scalable Private Set Intersection Based on OT Extension. ACM TOPS 21(2), 7:1–7:35 (2018)
50. T. Poppelmann, E. Alkim, R. Avanzi, J. Bos, L. Ducas, A. d.l. Piedra, P. Schwabe, D. Stebila: NewHope. Tech. rep., available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions. National Institute of Standards and Technology (2017)
51. P. Schwabe, R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J.M. Schanck, G. Seiler, D. Stehlé: CRYSTALS-KYBER. Tech. rep., available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions. National Institute of Standards and Technology (2019)
52. *Microsoft SEAL (release 3.3)*, https://github.com/Microsoft/SEAL (2019). Microsoft Research, Redmond, WA. 2019.
53. P.W. Shor: Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In: FOCS (1994)
54. N. Smart, F. Vercauteren: *Fully Homomorphic SIMD Operations*, Cryptology ePrint Archive, Report 2011/133 (2011). http://eprint.iacr.org/2011/133. 2011.
55. D. Unruh: Revocable Quantum Timed-Release Encryption. J. ACM 62(6), 49:1–49:76 (2015)
56. D. Unruh: Universally Composable Quantum Multi-party Computation. In: EUROCRYPT 2010. LNCS, pp. 486–505. Springer (2010)
57. X. Wang: A New Paradigm for Practical Maliciously Secure Multi-Party Computation. University of Maryland (College Park, Md.) (2018).
58. X. Wang, A.J. Malozemoff, J. Katz: *EMP-toolkit: Efficient MultiParty computation toolkit*, https://github.com/emp-toolkit (2016). 2016.
59. A.C.-C. Yao: Protocols for Secure Computations (Extended Abstract). In: 23rd FOCS, pp. 160–164. IEEE Computer Society Press (1982)
60. S. Zahur, M. Rosulek, D. Evans: Two Halves Make a Whole - Reducing Data Transfer in Garbled Circuits Using Half Gates. In: EUROCRYPT 2015, Part II. LNCS, pp. 220–250. Springer (2015)
61. M. Zhandry: How to Record Quantum Queries, and Applications to Quantum Indifferentiability. In: CRYPTO 2019, Part II. LNCS, pp. 239–268. Springer (2019)

A Additional Preliminary Material

A.1 Quantum Random Oracle Model

The quantum random oracle model (QROM) introduced by Boneh et al. [14] is the adaptation of the random oracle model (ROM) by Bellare and Rogaway [11] for the quantum world. In the ROM, every party, including the adversary, is granted access to a random oracle H. In the QROM, on the other hand, parties with quantum computing power have access to a quantum random oracle $|\mathsf{H}\rangle$, which upon being queried on $|x, y\rangle$ returns $|x, y \oplus \mathsf{H}(x)\rangle$. Boneh et al. [14] point out that the ROM is inappropriate when considering quantum adversaries, hence post-quantum security proofs should always be conducted in the QROM.

Below we state the one-way to hiding (O2H) Lemma by Unruh [55], albeit using the recent reformulation by Ambainis et al. [5] adapted to our case. The lemma gives a bound on the advantage that an adversary can distinguish between two random oracles, when given superposition access to these.

Lemma 1 (One-way to hiding (O2H) **[5]).** *Let* $\mathsf{G}, \mathsf{H} \colon \mathcal{X} \to \mathcal{Y}$ *be random functions, let* z *be a random value, and let* $\mathcal{S} \subset \mathcal{X}$ *be a random set such that* $\forall x \notin \mathcal{S}$, $\mathsf{G}(x) = \mathsf{H}(x)$. $(\mathsf{G}, \mathsf{H}, \mathcal{S}, z)$ *may have arbitrary joint distribution. Furthermore, let* $\mathcal{A}_q^{|\mathsf{H}\rangle}$ *be a quantum oracle algorithm which queries* $|\mathsf{H}\rangle$ *at most* q *times. Let* Ev *be an arbitrary classical event. Define an oracle algorithm* $\mathcal{B}_q^{|\mathsf{H}\rangle}$ *as follows: Pick* $i \leftarrow_{\$} [q]$. *Run* $\mathcal{A}_q^{|\mathsf{H}\rangle}(z)$ *until just before its* i*-th round of queries to* $|\mathsf{H}\rangle$. *Measure the query in the computational basis, and output the measurement outcome. Let*

$$\begin{aligned} P_{left} &\coloneqq \Pr[\mathsf{Ev} \colon \mathcal{A}_q^{|\mathsf{H}\rangle}(z)], \\ P_{right} &\coloneqq \Pr[\mathsf{Ev} \colon \mathcal{A}_q^{|\mathsf{G}\rangle}(z)], \\ P_{guess} &\coloneqq \Pr[x \in \mathcal{S} \colon \mathcal{B}_q^{|\mathsf{H}\rangle}(z) \to x] \,. \end{aligned}$$

Then it holds that

$$|P_{left} - P_{right}| \leq 2q\sqrt{P_{guess}} \,.$$

A.2 RLWE-based Additively Homomorphic Encryption

In this section, we describe an abstract Additively Homomorphic Encryption (AHE) scheme based on the Ring-LWE problem [37]. The concrete details of the scheme are given in Appendix A.3, and the pq-IND-CPA security and Circuit privacy provided by the scheme are discussed in Appendix A.4.

The plaintext space parameters are x and p, where x is a positive integer and p is a prime plaintext modulus. The plaintext space $\mathcal{P}$ is defined as $(\mathbb{F}_{p^d})^n$, where d is the order of p in $\mathbb{Z}_x^*$ and $n = \varphi(x)/d$. The elements of $\mathcal{P}$ are vectors of size n, where each entry corresponds to a plaintext slot and the arithmetic, denoted by $+$ and $\cdot$, is performed entry-wise on the vectors.

The scheme is defined using the following algorithms:

Parameter Generation. $\mathbb{P} \cup \{\perp\} \leftarrow \texttt{PGen}(1^\lambda, C_{\text{AHE}}, x, p)$: For security parameter λ, the class of permitted arithmetic circuits C_{AHE}, and the plaintext space parameters x and p, it outputs context $\mathbb{P}$ if parameters exist that allow evaluation of circuits in C_{AHE} while maintaining a security of at least λ bits, and $\perp$ else.

Key Generation. $(pk, sk) \leftarrow_{\$} \texttt{KGen}(\mathbb{P})$: This randomized algorithm takes a context $\mathbb{P}$ as input and outputs a public/secret key-pair (pk, sk).

Encryption. $c \leftarrow\!\!\$\ \texttt{Enc}(pk, \boldsymbol{m})$: This randomized algorithm outputs a ciphertext $c \in \mathcal{C}$, given a public key pk and a message $\boldsymbol{m} \in \mathcal{P}$. $\texttt{Enc}(pk, \cdot)$ is a homomorphism from $(\mathcal{P}, +)$ to $(\mathcal{C}, \boxplus)$, and this also implies scalar multiplication, i.e., $\boxdot : \mathcal{P} \times \mathcal{C} \to \mathcal{C}$. Homomorphic subtraction, denoted by $\boxminus$, can be perfomed as: $c_1 \boxminus c_2 = c_1 \boxplus (\boldsymbol{-1} \boxdot c_2)$, where $\boldsymbol{-1} \in \mathcal{P}$ is a vector with -1 in each slot. We remark that $\boxplus$ and $\boxminus$ also operate on pairs of plaintext and ciphertext.

Decryption. $\boldsymbol{m} \cup \{\perp\} \leftarrow \texttt{Dec}(sk, c)$: Given are a secret key sk and a ciphertext $c = \widehat{f}(\pi_1, \ldots, \pi_{I(f)}) \in \mathcal{C}$, where $\widehat{f}$ is the circuit induced by replacing the $+$, $-$, and $\cdot$ operators in f with $\boxplus$, $\boxminus$ and $\boxdot$ respectively, $I(f)$ is the number of inputs to the circuit f, and $\pi_i = \texttt{Enc}(pk, \boldsymbol{m}_i) \in \mathcal{C}$ or $\pi_i = \boldsymbol{m}_i \in \mathcal{P}$. If pk corresponds to sk and $f \in C_{\mathrm{AHE}}$, $\texttt{Dec}$ outputs $\boldsymbol{m} = f(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_{I(f)})$, else it outputs $\perp$.

A.3 The concrete AHE Scheme

In this section, we describe a subset of the BFV scheme [23], which we are referring to as the AHE scheme. The scheme is defined over the polynomial ring $R = \mathbb{Z}[X]/\Phi_x$, where Φ_x is the x-th cyclotomic polynomial. In addition to x, there are two other parameters p and q, which determine the plaintext space $\mathcal{P}$ and the ciphertext space $\mathcal{C}$ respectively. Each ciphertext in this scheme has a noise component associated with it, which grows as we perform arithmetic operations (homomorphically) on a ciphertext. For decryption to work, the noise has to be smaller than a certain threshold determined by the scheme parameters. Therefore, the AHE scheme can only evaluate a limited class of arithmetic circuits, which we denote by C_{AHE}.

Plaintext Space. The AHE scheme natively operates on $R_p = \mathbb{Z}_p[X]/\Phi_x$, where the plaintext modulus p is a prime and R_p consists of polynomials of degree $\varphi(x) - 1$ with coefficients in $\mathbb{Z}_p$. Let d be the order of p in $\mathbb{Z}_x^*$. Under modulo prime p, Φ_x factors into $n = \varphi(x)/d$ irreducible polynomials each of degree d such that $\Phi_x = \prod_{i=1}^{n} \mathsf{F}_i \pmod{p}$. Each factor F_i of Φ_x corresponds to the finite field $\mathbb{Z}_p[X]/\mathsf{F}_i \simeq \mathbb{F}_{p^d}$, and the following isomorphism holds:

$$R_p \simeq \mathbb{Z}_p[X]/\mathsf{F}_1 \times \ldots \times \mathbb{Z}_p[X]/\mathsf{F}_n \simeq \mathbb{F}_{p^d} \times \ldots \times \mathbb{F}_{p^d}.$$

Let $\mathcal{P}$ denote the plaintext space defined by the direct product of n finite fields $(\mathbb{F}_{p^d})^n$. We refer to each independent field $\mathbb{F}_{p^d}$ as a plaintext slot. The elements of $\mathcal{P}$ can be thought of as vectors of size n, where each entry corresponds to a plaintext slot and the arithmetic is performed entry-wise on the vectors. The authors in [54] exploited the isomorphism between $\mathcal{P}$ and R_p to construct isomorphic mappings $\texttt{Encode} : (\mathbb{F}_{p^d})^n \to R_p$ and $\texttt{Decode} : R_p \to (\mathbb{F}_{p^d})^n$. Thus, given polynomial encodings $\mathsf{a} = \texttt{Encode}\big((a_i)_{i\in[n]}\big)$ and $\mathsf{b} = \texttt{Encode}\big((b_i)_{i\in[n]}\big)$, we have:

$$\texttt{Decode}(\mathsf{a} + \mathsf{b} \bmod (p, \Phi_x)) = (a_i + b_i \bmod (p, \mathsf{F}_i))_{i\in[n]}\,,$$
$$\texttt{Decode}(\mathsf{a}\, \cdot\, \mathsf{b} \bmod (p, \Phi_x)) = (a_i\, \cdot\, b_i \bmod (p, \mathsf{F}_i))_{i\in[n]}\,,$$

where $\mathsf{a} \bmod (p, \mathsf{F})$ denotes $(\mathsf{a} \bmod \mathsf{F}) \bmod p$. Therefore, even though the scheme natively operates on polynomials in R_p, we can use the `Encode` and `Decode` mappings to operate on vectors in $\mathcal{P}$. This technique allows us to exploit the SIMD operations [54], and is called batching.

Ciphertext Space. The ciphertext space is defined as $\mathcal{C} = (R_q)^2$, where $R_q = \mathbb{Z}_q[X]/\Phi_x$. The ciphertext modulus q is taken as a product of k distinct primes q_i (also called the modulus chain), where each $q_i \equiv 1 \bmod x$.

Description of the Scheme. The scheme is defined as the tuple $\mathrm{AHE} = (\mathtt{PGen}, \mathtt{KGen}, \mathtt{Enc}, \mathtt{Dec})$ of algorithms, which are described as follows:

Parameter Generation. $\mathbb{P} \cup \{\perp\} \leftarrow \mathtt{PGen}(1^\lambda, C_{\mathrm{AHE}}, x, p)$: On the basis of x and p, this function first constructs the `Encode` and `Decode` mappings. It then chooses an appropriate error distribution $\chi = \chi(\lambda)$. Finally, it finds a large enough q to allow homomorphic evaluation of the circuits in C_{AHE} and also provide circuit privacy (cf. Section A.4), while maintaining the upper bound on q, which is a function of x and λ required to maintain security. If such a q does not exist, the function outputs $\perp$, else it outputs the context $\mathbb{P} = (x, p, q, \chi, \mathtt{Encode}, \mathtt{Decode})$. For concise representation, we omit the `Encode` and `Decode` mappings from the context $\mathbb{P}$ for the rest of the paper.

Key Generation. $(pk, sk) \leftarrow_{\$} \mathtt{KGen}(\mathbb{P})$: Given the context $\mathbb{P}$ as input, `KGen` samples $\mathsf{a} \xleftarrow{R} R_q$ and $\mathsf{s}, \mathsf{e} \leftarrow \chi$. It then sets $pk = (\mathsf{a}, \mathsf{b} = [\mathsf{a} \cdot \mathsf{s} + \mathsf{e}]_q)$ and $sk = \mathsf{s}$, and outputs the key-pair (pk, sk). We assume that both the keys implicitly contain the context $\mathbb{P}$.

Encryption. $c \leftarrow_{\$} \mathtt{Enc}(pk, \boldsymbol{m})$: Given the public key $pk = (\mathsf{a}, \mathsf{b}) \in (R_q)^2$ and a message $\boldsymbol{m} \in (\mathbb{F}_{p^d})^n$ as inputs, `Enc` first maps $\boldsymbol{m}$ to an element $m = \mathtt{Encode}(\boldsymbol{m}) \in R_p$. Let $\mu : R_p \to R_q$ be a function defined as $\mu(m) = \lfloor q/p \rfloor \cdot m$. `Enc` then samples $\mathsf{u}, \mathsf{e}', \mathsf{e}'' \leftarrow \chi$, and outputs the ciphertext $c = (\mathsf{c}_0 = [\mathsf{a} \cdot \mathsf{u} + \mathsf{e}']_q, \mathsf{c}_1 = [\mathsf{b} \cdot \mathsf{u} + \mathsf{e}'' + \mu(m)]_q) \in (R_q)^2$.

Decryption. $\boldsymbol{m} \cup \{\perp\} \leftarrow \mathtt{Dec}(sk, c)$: Given the secret key $sk = \mathsf{s} \in R_q$ and a ciphertext $c = (\mathsf{c}_0, \mathsf{c}_1) \in (R_q)^2$, `Dec` computes $\mathsf{t} = \frac{p}{q}[\mathsf{c}_1 - \mathsf{c}_0 \cdot \mathsf{s}]_q = m + \mathsf{v} + p \cdot \mathsf{k}$, where $\mathsf{v} \in R \otimes \mathbb{Q}$ is the polynomial with smallest inifinity norm for some $\mathsf{k} \in R$. `Dec` outputs $\perp$ if $\|\mathsf{v}\|_\infty \geq 1/2$, else it computes $m = [\lfloor \mathsf{t} \rceil]_q$ and outputs $\boldsymbol{m} = \mathtt{Decode}(m) \in (\mathbb{F}_{p^d})^n$. `PGen` ensures that q is large enough to keep the infinity norm of the noise polynomial v smaller than $1/2$ for the circuits in C_{AHE}.

A.4 Security of the AHE scheme

The security of the AHE scheme relies on the Decision-RLWE problem [37], which is conjectured to be hard even in the presence of quantum adversaries. Below, we first define the decisional ring learning with error (DRLWE) problem followed by the theorem that states the pq-IND-CPA security of the scheme. Finally, we discuss circuit privacy, which ensures privacy of the private inputs of the evaluator.

Definition 4 (Decisional Ring Learning With Errors Problem). *Let Φ_x be the x-th cyclotomic polynomial and $q \geq 2$ be an integer, that define $R = \mathbb{Z}[X]/\Phi_x$ and $R_q = R/qR$, and let χ be an error distribution over R. For $\mathsf{s} \leftarrow_{\$} R_q$, the decision-RLWE problem $\mathrm{DRLWE}_{M,x,q,\chi}$ is to distinguish M samples either all of the form $(\mathsf{a}, \mathsf{b} = [\mathsf{a} \cdot \mathsf{s} + \mathsf{e}]_q)$, where $\mathsf{a} \leftarrow_{\$} R_q$ and $\mathsf{e} \leftarrow \chi$, or of the form (a, b), where $\mathsf{a}, \mathsf{b} \leftarrow_{\$} (R_q)^2$. The advantage of an adversary $\mathcal{A}$ in solving the $\mathrm{DRLWE}_{M,x,q,\chi}$ problem is defined as $\mathbf{Adv}^{\mathrm{DRLWE}}_{M,m,q,\chi}(\mathcal{A})$.*

Theorem 6 (IND-CPA security of the AHE **scheme).** *Let x and q be integers, and χ be an error distribution on R_q. For any quantum adversary $\mathcal{A}$ against the* pq-IND-CPA *security of the* AHE *scheme, there exist quantum adversaries $\mathcal{R}_1$ and $\mathcal{R}_2$ against the* DRLWE *problem such that:*

$$\mathbf{Adv}^{\text{IND-CPA}}_{\mathrm{AHE}(x,q,\chi)}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{DRLWE}}_{1,x,q,\chi}(\mathcal{R}_1) + \mathbf{Adv}^{\mathrm{DRLWE}}_{2,x,q,\chi}(\mathcal{B}_2).$$

The proof of the theorem follows the standard security proof for LWE-based encryption schemes using two game hops which are bound by the DRLWE advantage, which is why we omit the formal proof.

Circuit Privacy. There are situations where the evaluator introduces secret inputs to the circuit. In such cases, the decryptor can compute the noise component of the output ciphertext and learn information about the secret inputs of the evaluator. To get around this problem, we use the "noise-flooding" technique introduced in [25]. This involves adding an encryption of zero (to randomize the ciphertext) with noise super-polynomially larger (specifically, σ bits larger for statistical security parameter σ) than the noise in the output ciphertext, to hide any statistical differences created by the private inputs of the evaluator.

B Postponed Proofs

B.1 Proof of Theorem 2

Correctness is immediate from the homomorphic properties of the AHE scheme, and thus, we proceed directly to the construction of simulators for corrupted sender and corrupted receiver.

Corrupted Sender. We construct a simulator $\mathcal{S}_{\mathrm{S}}$ that outputs a view computationally indistinguishable from the view of of the sender in $\Pi^{\mathrm{OT}}_{\mathrm{AHE}}$. The view of the sender in the protocol is $(\boldsymbol{m}_0, \boldsymbol{m}_1, \rho, pk, c_b)$, where ρ is the random tape of the sender. The simulator $\mathcal{S}_{\mathrm{S}}$ has access to the input $(\boldsymbol{m}_0, \boldsymbol{m}_1)$ of the sender in the protocol.

At the start $\mathcal{S}_{\mathrm{S}}$ runs `KGen` to obtain a key pair (pk, sk). It chooses a random tape ρ, a random message $\boldsymbol{m}$, computes the ciphertext $c \leftarrow \texttt{Enc}(pk, \boldsymbol{m})$, and outputs $(\boldsymbol{m}_0, \boldsymbol{m}_1, \rho, pk, c)$.

The view between the real view and the simulated view are computationally indistinguishable, which follows directly from the pq-IND-CPA security of the encryption scheme (cf. Theorem 6).

Corrupted Receiver. We construct a simulator $\mathcal{S}_{\mathrm{R}}$ that outputs a view statistically indistinguishable from the view of the receiver in $\varPi^{\mathrm{OT}}_{\mathrm{AHE}}$. The view of the receiver in the protocol is $(\boldsymbol{b}, \rho, c_{m_b})$, where ρ is the random tape of the receiver. The input to $\mathcal{S}_{\mathrm{R}}$ are the input $\boldsymbol{b}$ and output $\boldsymbol{m}$ of the receiver in the protocol.

The simulator $\mathcal{S}_{\mathrm{R}}(\boldsymbol{b}, \boldsymbol{m})$ proceeds as follows: It chooses a random tape ρ, runs $\mathtt{KGen}$ to obtain a key pair (pk, sk), and computes the ciphertext $c \leftarrow \mathtt{Enc}(pk, \boldsymbol{m})$. It then floods c with noise sampled from the same distribution as the extra noise used to drown the noise component of the output ciphertext in the real execution (cf. Appendix A.4). Finally, the simulator outputs $(\boldsymbol{b}, \rho, c)$.

It remains to show that c is distributed statistically close to the ciphertext c_{m_b} in the real execution ($\boldsymbol{b}$ and ρ are obviously distributed identically as in the real execution). The noise components in c and c_{m_b} are statistically indistinguishable. This is owed to the noise flooding technique (cf. Appendix A.4) used in the real execution by the Sender on c_{m_b}, and in the ideal execution by the simulator on c. Since c and c_{m_b} are encryptions of the same message, their statistical indistinguishability follows directly from the statistical indistinguishability of their noise components. This implies statistical indistinguishability of the views in the real and ideal execution.

B.2 Proof of Theorem 3

To prove the theorem, we show that there exists a simulator which can simulate the view of the corrupted party, which is either the sender or the receiver. The simulator has access to an ideal functionality which securely implements the OT extension protocol. Then the security follows if the adversary is unable to differentiate a real execution from a simulated one.

Recall that the communication between the parties remains classical in the post-quantum setting, even though the corrupted party has access to a quantum computer. This entails that the real view of the malicious (quantum) parties in the protocol remains the same, which allows to use the same simulators given in the classical proof. Now the main challenge lies in proving that the additional quantum computing power does not help in distinguishing between the real and simulated view. The additional quantum computing power is modelled by granting the adversary access to a quantum random oracle.

The proof is divided into two cases. In the first, the sender is assumed to be malicious, in the second, the receiver is assumed to be semi-honest. We show that translating the first case to the QROM is not an obstacle. For the second case, the situation is much more involved. The reason is that the classical proof keeps a list of the adversary's random oracle queries. To translate this into the QROM, we have to work around the no-cloning theorem which prevents simply copying the superposition queries by the adversary. In fact, this was widely believed to be an insurmountable barrier to lift such a proof. A workaround might be to use the *compressed oracles* technique, recently developed by Zhandry [61], which allows to keep a list of superposition queries, to some extent at least. However, we show that keeping such a list is not required for the proof. Instead we reduce the problem of distinguishing between different random oracles to distinguishing between the real and simulated view. This enables us to use the O2H lemma (cf. Lemma 1 on p. 21), resulting in a much simpler post-quantum security proof.

Malicious Sender. The simulator for a malicious sender S proceeds as follows. It runs S on a uniformly random input ρ. The input of S to the OT primitive in step 2 (cf. Fig. 3) is denoted by $\boldsymbol{s}^*$. The simulator sends the columns of a randomly generated matrix $\mathbf{Q}$ as the output of OT to S. The simulator invokes the trusted party with inputs $x^*_{j,0} \leftarrow y^*_{j,0} \oplus \mathsf{H}(j, \boldsymbol{q}_j)$ and $x^*_{j,1} \leftarrow y^*_{j,1} \oplus \mathsf{H}(j, \boldsymbol{q}_j \oplus \boldsymbol{s}^*)$, where $y^*_{j,0}$ and $y^*_{j,1}$ are the messages sent by S in step 3. Finally, the simulator outputs whatever S outputs.

The crucial observation made in [32] is that this simulator is perfect even for an arbitrary number of random oracle calls. That is, a distinguisher which queries the whole random oracle (or receives a full description of it) can still not distinguish between the real and the ideal process. Intuitively, the reason is that the view of the sender is just a set of random values obtained from the OT protocol in step 2. In the QROM, the adversary can query the random oracle on several inputs in superposition per query. However, it can by no means obtain more information than a classical adversary which queries the random oracle on all possible inputs. Hence, we conclude that the protocol is perfectly post-quantum secure with respect to a malicious sender.

Semi-honest Receiver. The simulator for a semi-honest receiver R proceeds as follows. It invokes the trusted party on input $\boldsymbol{r}$ to obtain outputs $z_1, \ldots, z_\tau$. It simulates the protocol between R and S, where it substitutes the known inputs of S, that is x_{j,r_j}, with z_j and the unknown inputs, that is $x_{j,1-r_j}$, with 0^l. Finally, it outputs the entire view of R.

The difference between the real execution and the simulation is that in the latter, the simulator sets half of the inputs of S to 0^l instead of random values. To detect the simulation, R has to obtain one of these values. By construction, however, these inputs are masked by XORing them with $\mathsf{H}(j, \boldsymbol{t}_j \oplus \boldsymbol{s})$ before sending them to R, for $\boldsymbol{s}$ unknown to R. To unmask such a value the distinguisher has to query the random oracle on $(j, \boldsymbol{t}_j \oplus \boldsymbol{s})$, which is called an *offending query* [32]. A bound on the probability of making an offending query is therefore also a bound on detecting the simulation. In the ROM this is straightforward, since the random oracle can be queried on exactly one input per query. In the QROM, on the other hand, the situation is more complicated. The main issue is to determine whether the adversary has made an offending query. Consider that the adversary makes an equal superposition over the domain of the random oracle. It is not obvious whether this query is offending or not.

To circumvent this issue, we do the following Let $\mathcal{S}$ be the set of offending queries and define, based on the random oracle H, the random oracle $\mathsf{H}_{\mathcal{S}\to\perp}$, where

$$\mathsf{H}_{\mathcal{S}\to\perp}(a) = \begin{cases} \perp & \text{, if } a \in \mathcal{S}, \\ \mathsf{H}(a) & \text{, else} . \end{cases}$$

Assume that the adversary is given access to $\mathsf{H}_{\mathcal{S}\to\perp}$ instead of H. Then it is not able to detect the simulation since all offending queries result in $\perp$, which prevents it from unmasking the unknown inputs. The very same holds in the QROM, where the adversary has access to $|\mathsf{H}_{\mathcal{S}\to\perp}\rangle$ instead of $|\mathsf{H}\rangle$.

It remains to bound the probability of the adversary to distinguish whether it is given access to $|\mathsf{H}\rangle$ or $|\mathsf{H}_{\mathcal{S}\to\perp}\rangle$. This is where we make use of the O2H lemma Note that the O2H lemma (cf. Lemma 1 on p. 21) allows to analyse the behaviour of the adversary when interacting with either $|\mathsf{H}\rangle$ or $|\mathsf{H}_{\mathcal{S}\to\perp}\rangle$ even when we can only simulate the oracle $|\mathsf{H}\rangle$. For the remaining part, let q denote the number of queries that $\mathcal{A}$ makes to its quantum random oracle ($|\mathsf{H}\rangle$ or $|\mathsf{H}_{\mathcal{S}\to\perp}\rangle$). By applying the O2H lemma, using that $\mathcal{A}$ has no information about $\boldsymbol{s}$, $|\mathcal{S}| = \tau$, and $|\mathsf{Dom}(\mathsf{H})| = \tau 2^{\kappa}$ we obtain

$$\begin{aligned}\Pr[\mathcal{A} \text{ detects simulation}] &\leq |\Pr[\mathcal{A}^{|\mathsf{H}\rangle} \to 1] - \Pr[\mathcal{A}^{|\mathsf{H}_{\mathcal{S}\to\perp}\rangle} \to 1]| \\ &\leq 2q\sqrt{\Pr[x \in \mathcal{S} \,|\, \mathcal{B}^{|\mathsf{H}\rangle} \to x]} \leq 2q\sqrt{\frac{\tau}{\tau 2^{\kappa}}} = 2q\sqrt{2^{-\kappa}}\,.\end{aligned}$$

This does not conclude the proof for a semi-honest receiver yet. The reason is that the receiver also invokes the random oracle during the protocol (step 4). At this point we make use of the fact that R is semi-honest, that is, it invokes the random oracle (classically) on input $(j, \boldsymbol{t}_j)$ for $j \in [\tau]$. These queries are offending if and only if $\boldsymbol{s} = \mathbf{0}$. Since $\boldsymbol{s}$ is sampled honestly from $\{0,1\}^{\kappa}$ by S this happens with probability $2^{-\kappa}$. Combined with the bound above, we conclude that a semi-honest (quantum) receiver, making q queries to the quantum random oracle, can differentiate between the real execution and the simulated one with probability at most $2^{-\kappa} + 2q\sqrt{2^{-\kappa}}$. Combining the result with the result for the first case concludes the proof.

B.3 Proof of Theorem 4

We show that the classical proof by Lindell and Pinkas [35] carries over to the post-quantum setting. The proof is divided into two cases, one for a semi-honest garbler and one for a semi-honest evaluator. For each of these, the classical proof constructs a simulator which simulates the view of the corrupted party solely based on its input and output. Then the proof is based on a hybrid argument showing that the simulated view and the real view are computationally indistinguishable. Note that the real view remains the same when switching to the post-quantum setting, i.e., a quantum adversary. This is due to the fact that the adversary runs the protocol with an honest classical party. The mere difference is that the adversary can perform local quantum computation in order to break the security of the protocol.

Garbler $\mathcal{G}$ corrupted. The view of the garbler $\mathcal{G}$ consists of its view from the OTs and the message (the circuit output) it obtains from the evaluator $\mathcal{E}$ at the end of the protocol.

Simulator $\mathcal{S}_{\mathcal{G}}$ for a corrupted garbler receives as input $(x, f(x,y))$ and outputs $(x, r, \mathcal{S}_{\mathcal{G}}^{OT}(k_{n+1}^0, k_{n+1}^1), \dots, \mathcal{S}_{\mathcal{G}}^{OT}(k_{2n}^0, k_{2n}^1), f(x,y))$, where r is a random tape that $\mathcal{S}_{\mathcal{G}}$ uses to generate the garbled circuit and $\mathcal{S}_{\mathcal{G}}^{OT}(k_{n+i}^0, k_{n+i}^1)$ is the simulator for the i-th OT execution, which exists by the post-quantum security of the OT protocol.

To prove that the simulated view is computationally indistinguishable from the real view, the classical proof uses a sequence of hybrid distributions $H_0, \dots, H_n$. In hybrid H_i, the first i OT executions are real while the remaining $n-i$ OT executions are simulated. Lindell and Pinkas show that a distinguisher $\mathcal{D}$ for any two consecutive hybrids H_{i-1} and H_i can be transformed into an adversary $\mathcal{A}$ that distinguishes between the real view of the OT protocol and the simulated one. In our case, $\mathcal{D}$ is quantum which entails that $\mathcal{A}$ is quantum. Given that the OT protocol is post-quantum secure, we conclude that the hybrids H_0 and H_n are indistinguishable as every consecutive hybrid H_{i-1} and H_i are indistinguishable. Note that H_n is not yet the real view of the protocol, since the simulator uses $f(x,y)$ to simulate the message that the evaluator sends to the garbler at the end of the protocol. The indistinguishability between the real view and H_n follows from the correctness of the protocol (cf. Theorem 1), regardless of the post-quantum setting.

Based on this, we conclude that the real view is indistinguishable from the output of $\mathcal{S}_{\mathcal{G}}$.

Evaluator $\mathcal{E}$ corrupted. The view of the evaluator $\mathcal{E}$ consists of a garbled circuit and n keys, which it receives from the garbler, as well as its view from the OTs.

The simulator $\mathcal{S}_{\mathcal{E}}$ for a corrupted evaluator receives as input $(y, \mathcal{F}(x,y))$ and outputs $(y, \widetilde{G}(C), k_1, \dots, k_n, \mathcal{S}_{\mathcal{E}}^{OT}(y_1, k_{n+1}), \dots, \mathcal{S}_{\mathcal{E}}^{OT}(y_n, k_{2n}))$, where $\widetilde{G}(C)$ is the 'fake' garbled circuit which evaluates to $\mathcal{F}(x,y)$ irrespectively of the used keys, $k_1, \dots, k_n$ are randomly sampled keys, and $\mathcal{S}_{\mathcal{E}}^{OT}(y_i, k_{n+i})$ is the simulator for the i-th OT execution, which exists by the post-quantum security of the OT protocol.

The same argument as above yields that the output of $\mathcal{S}_{\mathcal{E}}$ is computationally indistinguishable from H_0, where the OT executions are replaced with the real view, since the OT protocol is post-quantum secure. The difference between the real view of $\mathcal{E}$ and H_0 is that the latter contains the 'fake' garbled circuit $\widetilde{G}(C)$. To show these are computationally indistinguishable, another sequence of hybrids $H_0, \dots, H_{|C|}$ is introduced, where the first i gates[9] correspond to the 'fake' garbled circuit $\widetilde{G}(C)$, while the remaining $|C|-i$ gates correspond to the real garbled circuit $G(C)$. Lindell and Pinkas show that any distinguisher $\mathcal{D}$ between hybrids H_{i-1} and H_i can be transformed into an adversary $\mathcal{A}$ against the security of the encryption scheme. Given that the encryption scheme is pq-2Enc-secure, this yields that such a distinguisher does not exist.

Combined with the result above, this concludes the proof.

[9] Using a topological ordering for the gates in the circuit.

B.4 Proof of Theorem 5

Intuitively, the game pq2enc (cf. Fig. 5) looks very much akin to pq-INDCPA, except that more keys are involved and each challenge query consists of several message pairs. However, simply reducing the pq-IND-CPA advantage of E_S to the winning game pq2enc does not work. The issue is that every message part is encrypted using a different combination of keys. Consider the following reduction which uses the game pq-INDCPA to simulate encryptions under key k_1'. For every challenge $(x_0, y_0, z_0), (x_1, y_1, z_1)$, the adversary forwards (x_0, x_1) and (z_0, z_1) to its challenge oracle from the pq-INDCPA game and encrypts the result using k_0 and k_0' which it samples by itself. In order to simulate the correct challenge oracle for the adversary, it has to encrypt y_b under keys k_1 and k_0'. While the reduction knows both keys, it is unaware of the message it has to encrypt, as this would trivially allow to win the pq-INDCPA game.

To circumvent this issue, we introduce another game G (cf. Fig. 6). In this game, the second ciphertext part c_2 always contains the encryption of y_0, irrespectively of the secret bit b. By removing the issue described above, this allows to transform any adversary against G into an adversary against pq-INDCPA. It remains to bound the game advantage between pq2enc and G, which, in turn, can also be bound by the pq-IND-CPA security of E_S. Let $\mathcal{A}$ be a quantum adversary against pq2enc. A simple reformulation yields

```
Game G
procedure Initialize
  $b \leftarrow_\$ \{0,1\}$
  $k_0, k_1, k_0', k_1' \leftarrow_\$ \mathcal{K}$
return $k_0, k_1$

procedure $\overline{\mathsf{Enc}_0}(k, m)$
  $c \leftarrow \mathtt{Enc}(k_0', \mathtt{Enc}(k, m))$
  return $c$

procedure $\overline{\mathsf{Enc}_1}(k, m)$
  $c \leftarrow \mathtt{Enc}(k, \mathtt{Enc}(k_1', m))$
  return $c$

procedure Finalize $(b')$
  return $(b' = b)$

procedure $\mathsf{O_H}(\sum \alpha_{x,y} |x, y\rangle)$
  return $\sum \alpha_{x,y} |x, y \oplus \mathsf{H}(x)\rangle$

procedure $\overline{\mathsf{E}}(m_0, m_1)$
  parse $m_0$ as $x_0 \parallel y_0 \parallel z_0$
  parse $m_1$ as $x_1 \parallel y_1 \parallel z_1$
  $c_1 \leftarrow \mathtt{Enc}(k_0, \mathtt{Enc}(k_1', x_b))$
  $c_2 \leftarrow \mathtt{Enc}(k_0', \mathtt{Enc}(k_1, y_0))$
  $c_3 \leftarrow \mathtt{Enc}(k_0', \mathtt{Enc}(k_1', z_b))$
  $c \leftarrow (c_1, c_2, c_3)$
  return $c$
```

Fig. 6. Game G used in the proof of Theorem 5.

$$\mathbf{Adv}^{\mathsf{pq2enc}}(\mathcal{A}) = 2\,\mathbf{Adv}\left(\mathsf{pq2enc}^{\mathcal{A}}, \mathsf{G}^{\mathcal{A}}\right) + \mathbf{Adv}^{\mathsf{G}}(\mathcal{A})\,.$$

We first bound the game advantage between pq2enc and G. A simple reformulation yields

$$2\,\mathbf{Adv}\left(\mathsf{pq2enc}^{\mathcal{A}}, \mathsf{G}^{\mathcal{A}}\right) = \Pr[\mathcal{A}^{\mathsf{pq2enc}} \to 1 \,|\, b = 1] - \Pr[\mathcal{A}^{\mathsf{G}} \to 1 \,|\, b = 1]\,.$$

Now we construct a quantum reduction $\mathcal{R}_1$ playing the pq-INDCPA game using $\mathcal{A}$ as a subroutine. It samples three keys k_0, k_1, and k_1', and sends the former two to the adversary $\mathcal{A}$. For every query (k, m) that $\mathcal{A}$ makes to $\overline{\mathsf{Enc}_1}$, $\mathcal{R}_1$ answers with the ciphertext obtained by first encrypting m using key k_1' and then encrypting the result using key k. For queries (k, m) to $\overline{\mathsf{Enc}_0}$, $\mathcal{R}_1$ invokes its own oracle Enc on $\mathtt{Enc}(k, m)$ and forwards the response to $\mathcal{A}$. For challenge queries (m_0, m_1), $\mathcal{R}_1$ first parses m_0 as $x_0 \parallel y_0 \parallel z_0$ and m_1 as $x_1 \parallel y_1 \parallel z_1$. The ciphertext c_1 is computed locally by encrypting x_1 using keys k_1' and k_0. For the ciphertext c_3, $\mathcal{R}_1$ encrypts z_1 using k_1', queries its own oracle Enc on the result, and sets c_3 to the response. As for c_2, $\mathcal{R}_1$ encrypts both y_0 and y_1 using k_1, invokes its own challenge oracle E on the two ciphertexts, and assigns the response to c_2. For each of the above queries, $\mathcal{R}_1$ invokes its own quantum random oracle whenever the local simulation of the encryption algorithm requires it. Queries $\sum \alpha_{x,y} |x, y\rangle$ to the quantum random oracle by $\mathcal{A}$ are answered by forwarding the query to the quantum random oracle from the $\mathsf{pq\text{-}INDCPA}$ game and sending the response $\sum \alpha_{x,y} |x, y \oplus \mathsf{H}(x)\rangle$ back to $\mathcal{A}$. When $\mathcal{A}$ outputs a bit b', $\mathcal{R}_1$ simply forwards b' as its own output.

It is easy to see that $\mathcal{R}_1$ perfectly simulates $\mathsf{pq2enc}$ and G conditioned on $b = 1$ and $b = 0$, respectively, where b is the secret bit from the $\mathsf{pq\text{-}INDCPA}$ game. This yields

$$\begin{aligned} 2\,\mathbf{Adv}\left(\mathsf{pq2enc}^{\mathcal{A}}, \mathsf{G}^{\mathcal{A}}\right) &= \Pr[\mathcal{A}^{\mathsf{pq2enc}} \to 1 \mid b = 1] - \Pr[\mathcal{A}^{\mathsf{G}} \to 1 \mid b = 1] \\ &\leq \Pr[\mathcal{R}_1^{\mathsf{pq\text{-}INDCPA}} \to 1 \mid b = 1] - \Pr[\mathcal{R}_1^{\mathsf{pq\text{-}INDCPA}} \to 1 \mid b = 0] \\ &= \mathbf{Adv}^{\mathsf{pq\text{-}INDCPA}}(\mathcal{R}_1)\,. \end{aligned}$$

Next, we show that any quantum adversary $\mathcal{A}$ against G can be transformed into an adversary $\mathcal{R}_2$ against $\mathsf{pq\text{-}INDCPA}$. The idea is as follows: $\mathcal{R}_2$ samples all keys but k_1' by itself. Encryptions using key k_1' are simulated using its own oracles Enc and E while all other encryptions are performed locally.

At the start of the game, $\mathcal{R}_2$ chooses three keys k_0, k_1, and k_0' and sends k_0 and k_1 to $\mathcal{A}$. Queries (k, m) to $\overline{\mathsf{Enc}_0}$ can be simulated using only the quantum random oracle, as $\mathcal{R}_2$ knows both k and k_0'. Queries (k, m) to $\overline{\mathsf{Enc}_1}$ require to first invoke the oracle Enc on m before encrypting the response using key k, again querying the quantum random oracle as required by the encryption algorithm. For the challenge queries (m_0, m_1), $\mathcal{R}_2$ first parses m_0 and m_1 as $x_0 \parallel y_0 \parallel z_0$ and $x_1 \parallel y_1 \parallel z_1$, respectively. Recall that in G the value y_1 is never encrypted. Hence, the second ciphertext is computed locally, only invoking the quantum random oracle, by encrypting y_0 using the keys k_0' and k_1. As for the other challenge messages (x_0, x_1, z_0, z_1), $\mathcal{R}_2$ invokes its own challenge oracle E twice, namely on (x_0, x_1) and (z_0, z_1). The first response is encrypted using key k_0, the second using key k_0'. Finally, the ciphertexts are sent back to $\mathcal{A}$. Just as in the first part, queries $\sum \alpha_{x,y} |x, y\rangle$ that $\mathcal{A}$ makes to its quantum random oracle are forwarded to the quantum random oracle from the $\mathsf{pq\text{-}INDCPA}$ game, as is the response $\sum \alpha_{x,y} |x, y \oplus \mathsf{H}(x)\rangle$. Whenever $\mathcal{A}$ outputs a bit b', $\mathcal{R}_2$ simply outputs the same bit.

It is easy to see that $\mathcal{R}_2$ perfectly simulates G for $\mathcal{A}$, where the secret key from the $\mathsf{pq\text{-}INDCPA}$ games corresponds to the key k_1' in G. By forwarding the output of $\mathcal{A}$, we end up with

$$\mathbf{Adv}^{\mathsf{G}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{pq\text{-}INDCPA}}(\mathcal{R}_2)\,.$$

The statement of Theorem 5 follows by collecting the bounds above and defining $\overline{\mathcal{A}}$ to be the adversary from $\mathcal{R}_1$ and $\mathcal{R}_2$ with the higher pq-IND-CPA advantage.

Remark. By removing the quantum random oracle from the games `pq2enc` and G and restricting $\mathcal{A}$ to be classical, we obtain a classical security proof for the security under double encryption, yielding essentially the detailed security proof based on the proof sketch of [35].

C Parameter Selection for PQ-secure OT.

The parameters of the AHE scheme used in our experiments are as follows: $p = 65537, q = 1099510054913 \cdot 1152921504606584833, x = 16384$, and $s = 3.2$ (χ is a discrete Gaussian distribution with standard deviation s). Consequently, we get $\alpha = 16$, and $\gamma = 512$ for $\ell = 256$. We use the conservative analysis (from the point of view of the key owner) of the hardness of the decision Ring-LWE (DRLWE) problem from NewHope [3] to estimate the post-quantum security for our AHE scheme. The security of their PKE scheme has an identical reduction to the DRLWE problem as our AHE scheme. Thus, their analysis applies directly to our case as well. Using the script (`scripts/PQsecurity.py`) provided by the authors of NewHope [50], the post-quantum security for these parameters comes out to be 258 bits. This is greater than the post-quantum security of NewHope1024 (233 bits), which is being considered for category V security in the NIST post-quantum cryptography standardization process [45]. According to NIST, category V security guarantees a greater effort in breaking the scheme than the effort required for key search on AES-256. Thus, the addition of post-quantum OT does not lower the security of the scheme, and the security of AES-256 is still the lower bound on security of the post-quantum Yao protocol.

Our choice of parameters has been optimized for using PQ-OT as a base-OT in OT extension [32], which we prove to be post-quantum secure (cf. Section 3.2). Specifically, exactly half of the plaintext slots are utilised for performing $2\kappa = 256$ OTs on 256-bit messages, which is the requirement for the base-OT.

The justification for concrete parameters is as follows: Microsoft's SEAL [52], on which our implementation of AHE is based, restricts x to be a power of 2 for efficiency purposes. Accordingly, we choose $p = 65537$ as our plaintext modulus, which is the smallest prime that satisfies $p \equiv 1 \bmod x$ for practical values of x. Consequently, we required a q of 100 bits to allow evaluation of the circuit employed in $\Pi^{\mathrm{OT}}_{\mathrm{AHE}}$, and ensure circuit privacy with statistical security of $\sigma = 40$ bits. For these parameters, the smallest value of x that provides more than 128-bit PQ security is 16384. Choosing $x = 8192$ would lead to perfect utilization of slots for performing 256 OTs on 256-bit messages, but it only provides PQ security of 111 bits according to the rather pessimistic analysis in [3]. We believe that a less conservative analysis will allow $x = 8192$, and expect this change to double the efficiency of the scheme in performing the base-OTs.

We discuss possible security against malicious adversaries in Appendix D.

D PQ-OT with Malicious Receivers

The protocol that we describe in Section 3 is only secure against semi-honest adversaries. In practice it is desirable to achieve security against malicious adversaries. While we did not implement this stronger form of security in our instantiation, we want to provide an overview of possibilities that achieve security against active adversaries. An obvious attack on our 1-out-of-2 OT that could be performed by a malicious receiver is setting the choice bit b to a value that is neither 0 nor 1. Thereby the receiver would receive a superposition of both sender messages. To mitigate this issue, we propose adding a range proof, e.g., using [16] with which the verifier can obliviously show that his choice bit is indeed either 0 or 1. Along with a range proof, we would also require a Zero-Knowledge Proof of Knowledge (ZKPoK), which will ensure that the noise in a ciphertext is less than a known bound.

When running multiple OTs at the same time, choice bit replication would need an additional proof of equality of all replicated choice bits. In that case higher degree slots are beneficial, as this additional proof is not needed.

An alternative construction that offers security against malicious receivers and can be instantiated with post-quantum secure primitives is provided in [39]. Their solution circumvents the above-described problem by not explicity encrypting the coice bit, but instead encoding it by chosing a message at the desired index according to the protocol specification.