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, Darmstadt, Germany
{buescher,karvelas}@seceng.informatik.tu-darmstadt.de
[2] SVS, Universität Hamburg, Hamburg, Germany
demmler@informatik.uni-hamburg.de
[3] Universität Passau, Passau, Germany
stefan.katzenbeisser@uni-passau.de
[4] QPC, Technische Universität Darmstadt, Darmstadt, Germany
{juliane.kraemer,patrick.struck}@tu-darmstadt.de
[5] Department of Computer Science, IIT (BHU) Varanasi, Varanasi, India
deevashwer.student.cse15@iitbhu.ac.in
[6] ENCRYPTO, Technische Universität Darmstadt, 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 [47] can be executed on a real quan-

M. Conti et al. (Eds.): ACNS 2020, LNCS 12146, pp. 461–480, 2020.
https://doi.org/10.1007/978-3-030-57808-4_23

tum 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 [6,36]. 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 [4]. This will give rise to the so-called quantum era [10], 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 [2,17,29,35] 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 [13] and Unruh's quantum lifting [48] 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 [51], 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 [32,49], and several protocol optimizations made GCs practical for many use cases in the last decade. Protocol optimizations such as Garbled Row Reduction [38,42], the free-XOR technique [30], fixed-key garbling [8], the half-gates approach [53], OT extension [5,28], 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 [48]. 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 [40].

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 Sect. 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. [28] is secure against quantum adversaries.
2) We implement our OT protocol in C++ using the Microsoft SEAL homomorphic encryption library [46]. In Sect. 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 Sect. 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 [31] 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. There are several implementations available, that show practical performance for Yao's garbled circuits protocol [16,50,52], that could benefit from incorporating security against quantum adversaries. A full proof of classical security for Yao's garbled circuits protocol was given in [31]. In [14], the free-XOR optimization [30] 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 [7,33]. A formally verified software stack for Yao's garbled circuits was presented in [3]. Known instantiations for post-quantum secure OT protocols are either based on the code-based McEliece crypto system [19] or on the learning with errors (LWE) problem [11]. In [34], 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 the full version of this paper [12].

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 [9]. 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 $\mathcal{A}^{\mathsf{G}} \rightarrow y$ 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 $\mathsf{G}^{\mathcal{A}} \rightarrow y$. 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 the full version of this paper [12]. 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 [39].

2.2 Encryption

A secret key encryption scheme E_S is a pair of efficient algorithms $\texttt{Enc}$ and $\texttt{Dec}$ for encryption and decryption, where $\texttt{Enc}(k, m) \rightarrow c$ and $\texttt{Dec}(k, c) \rightarrow m$ 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* $\mathrm{E}_S = (\texttt{Enc}, \texttt{Dec})$ *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* $\mathcal{A}$*:*

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

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

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

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

procedure E($m_0, m_1$)
    $c \leftarrow_{\$} \texttt{Enc}(k, m_b)$
    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 [27], and were assumed to be very costly in the past. However, in 2003 Ishai et al. [28] 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 [51] 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

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

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 [31]**).** *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* $\Pr[c \in \mathsf{Range}_n(k) \mid \mathcal{A}(1^n) \rightarrow c] \leq \mathrm{negl}(n)$*, 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.* $M(k, c) \rightarrow 1$ *if and only if* $c \in \mathsf{Range}_n(k)$*.*

Theorem 1 (Correctness of Yao's GC Protocol [31]**).** *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. Sect. 3.1) and prove OT extension post-quantum secure (cf. Sect. 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 $m_b = \texttt{Dec}(sk, c_{m_b})$.

We instantiate it using the PQ-secure BFV homomorphic encryption scheme [20] in the implementation provided by Microsoft's SEAL library [46]. 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. the full version of this paper [12, Appendix A.2]), we can choose a plaintext modulus p of more than ℓ bits, such that $p \equiv 1 \bmod x$, i.e., $d = \texttt{ord}_{\mathbb{Z}_x^*}(p) = 1$. 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 [26, 46].

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:

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

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

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

Higher Degree Slots. Alternatively, instead of restricting ourselves to p of order 1, we consider p of higher order $\beta = d = \texttt{ord}_{\mathbb{Z}_x^*}(p) \geq 1$. 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:

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

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

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

The Final Protocol. The final OT protocol $\Pi_{\mathrm{AHE}}^{\mathrm{OT}}$ 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 (≈20 ms in a LAN network, cf. Sect. 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.

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..........	
$\mathbf{1} \leftarrow \texttt{PackB}((1)_{i\in[\gamma]})$	$\xleftarrow{\quad pk \quad}$	$(pk, sk) \leftarrow \texttt{KGen}(\mathbb{P})$
	..........OT Phase..........	
$\boldsymbol{m}'_0 \leftarrow \texttt{PackM}(\boldsymbol{m}_0)$ $\boldsymbol{m}'_1 \leftarrow \texttt{PackM}(\boldsymbol{m}_1)$ $c_{\bar{b}} \leftarrow \mathbf{1} \boxminus c_b$ $c_0 \leftarrow \boldsymbol{m}'_0 \boxdot c_{\bar{b}}, \quad c_1 \leftarrow \boldsymbol{m}'_1 \boxdot c_b$	$\xleftarrow{\quad c_b \quad}$	$\boldsymbol{b}' \leftarrow \texttt{PackB}(\boldsymbol{b})$ $c_b \leftarrow \texttt{Enc}(pk, \boldsymbol{b}')$
$c_{m_b} \leftarrow c_0 \boxplus c_1$	$\xrightarrow{\quad c_{m_b} \quad}$	$\boldsymbol{m}'_b \leftarrow \texttt{Dec}(sk, c_{m_b})$ $\boldsymbol{m}_b \leftarrow \texttt{UnpackM}(\boldsymbol{m}'_b)$ **Output:** $\boldsymbol{m}_b = (m_{b_i,i})_{i\in[\gamma]}$

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

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. Details are given in [12].

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. [28] and allows to obtain many 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 Sect. 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 [23] or Private Set Intersection [41,43,44].

In the following theorem, we show that OT extension [28] is post-quantum secure. The proof is given in [12].

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

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 $(\boldsymbol{t}^i, \boldsymbol{r} \oplus \boldsymbol{t}^i)$, $1 \leq i \leq \kappa$
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 $1 \leq j \leq \tau$, 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 $1 \leq j \leq \tau$, R outputs $z_j \leftarrow y_{j,r_j} \oplus \mathsf{H}(j, \boldsymbol{t}_j)$.

Fig. 3. OT extension protocol from [28].

Theorem 3. *The OT extension protocol from [28] 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 [24]. 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 [49,50] 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 Sect. 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 [7,33], 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 [50] of the classical Yao's protocol with state-of-the-art optimizations: free-XOR [30], fixed-key AES-128 garbling [8], and half-gates [53] on $\kappa = 128$-bit wire labels.

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

	PQ	C	EMP [50]
PQ-Secure	✓	✗	✗
OT	PQ-OT (Sect. 3)	OT extension [28]	OT extension [28]
Point& Permute [7,33]	✓	✓	✓
Free-XOR [30]	✗	✗	✓
Half-Gates [53]	✗	✗	✓
Garbling	Variable-Key AES-256	Variable-Key AES-128	Fixed-Key AES-128 [8]

The circuits we benchmarked are described in Table 2.

Table 2. Boolean circuits used to benchmark Yao's protocol in Sect. 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

Table 3. Performance comparison of our PQ-Yao protocol, with a classical unoptimized Yao protocol (C), and the classical optimized EMP version [50] 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

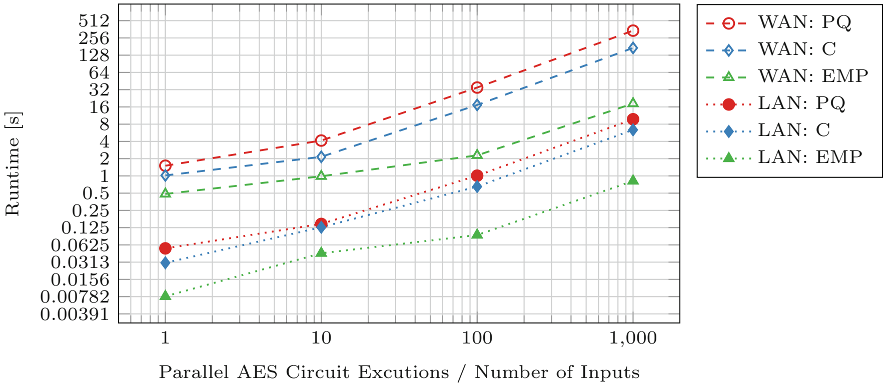

Fig. 4. Comparison of implementations of our PQ-Yao, with the classical, unoptimized Yao protocol (C), and the classical, optimized EMP version in a LAN and WAN network. Evaluation time for parallel executions of an AES circuit.

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 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 [25], since their code is not publicly available. Based on experimental results in [25], 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 [50] 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 Sect. 3 using the Microsoft SEAL library [46]. We use the implementation from the EMP toolkit [50] 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. Sect. 3).
- NP: classical Naor-Pinkas (NP)-OT [37] on 128-bit inputs, from EMP.
- OTe: classical semi-honest OT extension of [28] on 128-bit inputs, from the implementation in EMP. It uses NP-OT [37] 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. Sect. 3) with the classical Naor-Pinkas (NP)-OT [37] 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. [22] with Kyber-1024 (AVX2 optimized 90s variant) [45] 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 ≈ 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 Sect. 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. Sect. 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 [21]. 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 [31]. 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 [18] and Slae [15].[1]

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

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 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 is given in the full version of this paper [12].

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*[2]*, and the encryption scheme has an elusive and efficiently verifiable range. Then the protocol described in Sect. 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 [31] 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 [31] 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

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

the (classical) one given in [31]. First, we allow for multiple challenge queries from the adversary while [31] allow merely one. Second, the two known keys are honestly generated by the challenger and then handed over to the adversary. In [31], 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* $\mathsf{pq2enc}$ *be defined as in Fig. 5. Then for any quantum adversary* $\mathcal{A}$ *its advantage against the double encryption security is defined as:*

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

We say that E_S *is* pq-2Enc-*secure if* $\mathbf{Adv}_{\mathrm{E}_S}^{\mathsf{pq2enc}}(\mathcal{A})$ *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 $\mathsf{O_H}(\sum \alpha_{x,y} |x, y\rangle)$
    return $\sum \alpha_{x,y} |x, y \oplus \mathsf{H}(x)\rangle$

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

Fig. 5. Game $\mathsf{pq2enc}$ 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 the full version of this paper [12].

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:*

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

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. Aiello, B., Ishai, Y., Reingold, O.: Priced oblivious transfer: how to sell digital goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_8
2. Alkim, E., Alkim, E., et al.: Revisiting TESLA in the quantum random Oracle model. In: Lange, T., Takagi, T. (eds.) PQCrypto 2017. LNCS, vol. 10346, pp. 143–162. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59879-6_9
3. Almeida, J.B., et al.: A fast and verified software stack for secure function evaluation. In: ACM CCS 2017, pp. 1989–2006. ACM Press (2017)
4. Arute, F., et al.: Quantum supremacy using a programmable superconducting processor. Nature **574**(7779), 505–510 (2019)
5. Asharov, G., Lindell, Y., Schneider, T., Zohner, M.: More efficient oblivious transfer extensions. J. Cryptol. **30**(3), 805–858 (2017)
6. Bauer, B., Wecker, D., Millis, A.J., Hastings, M.B., Troyer, M.: Hybrid quantum-classical approach to correlated materials (2015)
7. Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols (extended abstract). In: 22nd ACM STOC, pp. 503–513. ACM Press (1990)
8. Bellare, M., Hoang, V.T., Keelveedhi, S., Rogaway, P.: Efficient garbling from a fixed-key blockcipher. In: 2013 IEEE Symposium on Security and Privacy, pp. 478–492. IEEE Computer Society Press (2013)
9. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_25
10. Bernstein, D.J., Buchmann, J., Dahmen, E.: Post-Quantum Cryptography. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-540-88702-7
11. Brakerski, Z., Döttling, N.: Two-message statistically sender-private OT from LWE. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018. LNCS, vol. 11240, pp. 370–390. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03810-6_14
12. Büscher, N., et al.: Secure two-party computation in a quantum world. Cryptology ePrint Archive, Report 2020/441 (2020). https://eprint.iacr.org/2020/411
13. Canetti, R.: 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)
14. Choi, S.G., Katz, J., Kumaresan, R., Zhou, H.-S.: On the security of the "Free-XOR" technique. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 39–53. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28914-9_3
15. Degabriele, J.P., Janson, C., Struck, P.: Sponges resist leakage: the case of authenticated encryption. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part II. LNCS, vol. 11922, pp. 209–240. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34621-8_8
16. Demmler, D., Schneider, T., Zohner, M.: ABY - a framework for efficient mixed-protocol secure two-party computation. In: NDSS 2015. The Internet Society (2015)
17. Ding, J., Schmidt, D.: Rainbow, a new multivariable polynomial signature scheme. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 164–175. Springer, Heidelberg (2005). https://doi.org/10.1007/11496137_12

18. Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F., Unterluggauer, T.: ISAP - towards side-channel secure authenticated encryption. IACR Trans. Symm. Cryptol. **2017**(1), 80–105 (2017)
19. Dowsley, R., van de Graaf, J., Müller-Quade, J., Nascimento, A.C.A.: Oblivious transfer based on the McEliece assumptions. In: Safavi-Naini, R. (ed.) ICITS 2008. LNCS, vol. 5155, pp. 107–117. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85093-9_11
20. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144 (2012). http://eprint.iacr.org/2012/144. 2012
21. Gagliardoni, T.: Quantum security of cryptographic primitives. Darmstadt University of Technology, Germany (2017)
22. Gertner, Y., Kannan, S., Malkin, T., Reingold, O., Viswanathan, M.: The relationship between public key encryption and oblivious transfer. In: 41st FOCS, pp. 325–335. IEEE Computer Society Press (2000)
23. Goldreich, O., Micali, S., Wigderson, A.: 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)
24. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: 28th ACM STOC, pp. 212–219. ACM Press (1996)
25. Gueron, S., Lindell, Y., Nof, A., Pinkas, B.: Fast garbling of circuits under standard assumptions. J. Cryptol. **31**(3), 798–844 (2018)
26. Halevi, S., Shoup, V.: HElib-an implementation of homomorphic encryption. Cryptology ePrint Archive, Report 2014/039. http://eprint.iacr.org/2014/039
27. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: 21st Annual ACM Symposium on Theory of Computing, 14–17 May 1989, Seattle, Washigton, USA, pp. 44–61 (1989)
28. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 145–161. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_9
29. Jao, D., De Feo, L.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 19–34. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25405-5_2
30. Kolesnikov, V., Schneider, T.: Improved garbled circuit: free XOR gates and applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008. LNCS, vol. 5126, pp. 486–498. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70583-3_40
31. Lindell, Y., Pinkas, B.: A proof of security of Yao's protocol for two-party computation. J. Cryptol. **22**(2), 161–188 (2009)
32. Lindell, Y., Pinkas, B.: An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 52–78. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72540-4_4
33. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay - secure two-party computation system. In: USENIX Security 2004, pp. 287–302. USENIX Association (2004)
34. Masny, D., Rindal, P.: Endemic oblivious transfer. In: ACM CCS 2019, pp. 309–326. ACM Press (2019)
35. McEliece, R.J.: A public-key cryptosystem based on algebraic coding theory. DSN Progress Report (1978)
36. Mosca, M.: Cybersecurity in an era with quantum computers: will we be ready? IEEE Secur. Priv. **16**(5), 38–41 (2018)

37. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: 12th SODA, pp. 448–457. ACM-SIAM (2001)
38. Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: ACM Conference on Electronic Commerce, pp. 129–139 (1999)
39. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press (2011)
40. NIST: PQ Cryptography Standardization Process (2017)
41. Pinkas, B., Schneider, T., Segev, G., Zohner, M.: Phasing: private set intersection using permutation-based hashing. In: USENIX Security 2015, pp. 515–530. USENIX Association (2015)
42. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure two-party computation is practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 250–267. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_15
43. Pinkas, B., Schneider, T., Zohner, M.: Faster private set intersection based on OT extension. In: USENIX Security 2014, pp. 797–812. USENIX Association (2014)
44. Pinkas, B., Schneider, T., Zohner, M.: Scalable private set intersection based on OT extension. ACM TOPS **21**(2), 7:1–7:35 (2018)
45. Schwabe, R., et al.: CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology (2019). https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions
46. "Microsoft SEAL (release 3.3)". Microsoft Research, Redmond, WA (2019). https://github.com/Microsoft/SEAL
47. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: FOCS (1994)
48. Unruh, D.: Universally composable quantum multi-party computation. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 486–505. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_25
49. Wang, X.: A New Paradigm for Practical Maliciously Secure Multi-Party Computation. University of Maryland, College Park (2018)
50. Wang, X., Malozemoff, A.J., Katz, J.: EMP-toolkit: efficient multiparty computation toolkit (2016). https://github.com/emp-toolkit
51. Yao, A.C.-C.: Protocols for secure computations (extended abstract). In: 23rd FOCS, pp. 160–164. IEEE Computer Society Press (1982)
52. Zahur, S., Evans, D.: Obliv-C: a language for extensible data-oblivious computation. Cryptology ePrint Archive, Report 2015/1153 (2015). http://eprint.iacr.org/2015/1153
53. Zahur, S., Rosulek, M., Evans, D.: Two halves make a whole. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 220–250. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_8