On The Orthogonal Vector Problem and The Feasibility of Unconditionally Secure Leakage Resilient Computation

Ivan Damgård, Frédéric Dupuis, and Jesper Buus Nielsen[⋆]

Dept. of Computer Science, Aarhus University

Abstract. We consider unconditionally secure leakage resilient two-party computation, where security means that the leakage obtained by an adversary can be simulated using a similar amount of leakage from the private inputs or outputs. A related problem is known as circuit compilation, where there is only one device doing a computation on public input and output. Here the goal is to ensure that the adversary learns only the input/output behaviour of the computation, even given leakage from the internal state of the device. We study these problems in an enhanced version of the "only computation leaks" model, where the adversary is additionally allowed a bounded amount of *global* leakage from the state of the entity under attack. In this model, we show the first unconditionally secure leakage resilient two-party computation protocol. The protocol assumes access to correlated randomness in the form of a functionality f_{ORT} that outputs pairs of orthogonal vectors $(\boldsymbol{u}, \boldsymbol{v})$ over some finite field, where the adversary can leak independently from $\boldsymbol{u}$ and from $\boldsymbol{v}$. We also construct a general circuit compiler secure in the same leakage model. Our constructions work, even if the adversary is allowed to corrupt a constant fraction of the calls to f_{ORT} and decide which vectors should be output. On the negative side, we show that unconditionally secure two-party computation and circuit compilation are in general impossible in the plain version of our model. For circuit compilation we need a computational assumption to exhibit a function that cannot be securely computed, on the other hand impossibility holds even if global leakage is not allowed. It follows that even a somewhat unreliable version of f_{ORT} cannot be implemented with unconditional security in the plain leakage model, using classical communication. However, we show that an implementation using quantum communication does exist. In particular, we propose a simple "prepare-and-measure" type protocol which we show secure using a new result on sampling from a quantum population. Although the protocol may produce a small number of incorrect pairs, this is sufficient for leakage resilient computation by our other results.

1 Introduction

In this paper, we consider secure leakage-resilient computation, more precisely solutions to two different types of problems, known as two-party computation and circuit compilation.

In two-party computation, two parties A and B want to compute a (possibly probabilistic) function securely on private inputs. Both parties are assumed to follow the protocol. There is an adversary who may obtain leakage from the internal state of the players, say by some side channel attack, but we still want to keep the private inputs and outputs "as private as possible".

In the closely related problem of circuit compilation, there is only a single device that carries out a computation (usually given as a Boolean circuit) on public input and output. The goal is to make sure that an adversary who gets to choose the input and is given the output, will learn nothing more than the input/output behaviour, even if he can leak from the internal state of the device. The computation may in addition depend on some secret data that should be completely protected. One may think, for instance, of a device that uses a secret key to decrypt or sign data: we are fine with revealing the output but want to hide the secret key, even if an adversary does a side-channel attack on the device.

We need, of course, a meaningful notion of security for protocols claiming to solve such problems. For this, we clearly need to somehow limit the leakage that the adversary can get, and there is a lot of previous work considering various restrictions allowing secure protocols to be built. In this paper we consider a new and enhanced version of the "only computation leaks" model proposed by Micalin and Reyzin in [MR04]. They assume that the computation of a party under attack is split into several smaller parts. The adversary may submit one or more leakage functions, where each function must address a certain part of the computation. The function is applied to only the data that is actually used during this

[⋆] Partially supported by Danish Council for Independent Research via DFF Starting Grant 10-081612. Partially supported by the European Research Commission Starting Grant 279447 part of the computation, and the result is returned to the adversary. It is assumed that only some bounded number of bits can leak from each part of the computation.

Our enhancement to the model is to allow not only leakage from sub-computations, but also a bounded amount of leakage from the *global* state of the party under attack. Typically, the allowed amount of global leakage will be comparable to what is allowed from a single part of the computation. This clearly makes the adversary stronger, but also more realistic: in a real life, an adversary might use a combination of several side channel attacks, and completely avoiding global leakage in such a case may be extremely difficult.

Security for two-party computation now means that the leakage obtained by the adversary can be (efficiently) simulated, however, when the adversary leaks from a part where private inputs or outputs are touched, the simulator is allowed to leak the same number of bits from those inputs/outputs. For circuit compilation, we require that the leakage can be simulated given only the public inputs and outputs. We emphasize that we consider throughout unbounded adversaries and leakage functions, nevertheless we demand that all simulators and honest parties must be efficient.

For two-party computation, a UC based definition of leakage resilience was given in [BCH12]. Also a few examples were given of two party protocols for specific functionalities (based on various computational assumptions). In [BGJK12], generic leakage resilient multiparty computation protocols were constructed, again under a computational assumption.

For circuit compilation, Dziembowski and Faust in [DF12] (see also [DF11]) have shown a general unconditionally secure method. Their construction can be interpreted in a natural way as being derived from a two-party protocol for parties A and B in the only-computation-leaks model (without global leakage, and where a single device plays the parts of both parties). They assume that so-called leak-free gates are available, or equivalently, parties are given access to an ideal functionality f_{ORT} which will on request produce a random pair of vectors $\boldsymbol{u}, \boldsymbol{v}$ over some finite field, subject to $\boldsymbol{u} \cdot \boldsymbol{v} = 0$ and give $\boldsymbol{u}$ to A and $\boldsymbol{v}$ to B. It is assumed that nothing about the internal computation of f_{ORT} leaks to the adversary. The idea is to use the vectors from the leak-free gates to refresh the state in between different parts of the computation, such that leakage from different parts will become (essentially) uncorrelated. The drawback of this solution is that an implementation will require secure hardware components, even if these are much simpler than the device we want to protect. Goldwasser and Rothblum [GR12] improve this using a different method that does not use leak-free gates during the computation, but instead assumes that the device initially has a randomised state, a so-called ciphertext bank. They then use this as well as fresh randomness as a source for refreshing. Also this construction is in the only-computation-leaks model (with no global leakage). The adversary does not have access to the internal computation of the algorithm that generates the ciphertext bank, so one can think of this as allowing access once and for all to a leak-free component (with output size depending only on the security parameter) before the computation starts.

It is of course natural to ask if we can make do with no leak-free components at all? In particular, can we securely implement f_{ORT} by a two-party leakage resilient protocol?

Our Contribution. In this paper, we make the following contributions:

– We generalise the protocol in [DF12] to show that we can do, not just circuit compilation, but general unconditionally-secure leakage-resilient 2-party computation given access to f_{ORT}. To the best of our knowledge, this is the first result of its kind. On the way, we propose a new model of leakage resilient 2-party computation where global leakage from the entire view of the party under attack is (also) allowed. This is a strictly stronger leakage model than that of [DF12, GR12]. The definition is simpler than the UC based definition of [BCH12] while still supporting composition.
– Our construction works even given a partially corrupted version of f_{ORT}, namely one where the adversary is allowed to corrupt a constant fraction of the calls to f_{ORT} and choose the output vectors himself. We believe this makes the assumption on leak free gates much more reasonable, as we make a much weaker assumption on the security of these gates. The result improves on [DF12] and

is incomparable to [GR12]: we need a (partially) leak-free source during the computation but on the other hand we allow active corruption of it under a stronger leakage model.

– We show that general two-party unconditionally secure leakage resilient computation is impossible in the plain version of our model (where no auxiliary functionalities are assumed). We show that circuit compilation is also impossible in the plain model, where for this result, we need a computational assumption to exhibit a function that cannot be computed securely, on the other hand, this impossibility result holds even if global leakage is not allowed.
– From the results mentioned, it follows easily that there is no unconditionally secure and leakage-resilient implementation of f_{ORT}, not even for a somewhat unreliable version. It also follows that the ciphertext bank of [GR12] cannot be securely created, starting from scratch.
– We show that the orthogonal vector problem does have an unconditional solution using quantum communication, in particular we provide a relatively simple "prepare-and-mesaure" type protocol. While it generates a small but noticeable number of incorrect pairs, this is still sufficient for leakage resilience by our other results. We show security using a new technical result of independent interest on sampling from a quantum population.

It should be noted that the impossibility of two-party computation from scratch follows because a secure protocol in our model is also passively secure in the standard (non-leakage) model. Not only does this show that correlated randomness is necessary, it also follows that the amount of such randomness must depend on the size of the secure computation: a generic two-party computation protocol using a fixed amount of randomness could be used to implement any number of oblivious transfers from such fixed-size randomness, and this is well known to be impossible. In our protocol, the number of calls we make to f_{ORT} is proportional to the circuit size of the function we compute, so the argument we just gave says that this is in some sense necessary. Note that the result of [GR12] does not contradict this as it is does only circuit compilation in a weaker leakage model.

We now explain a few additional details about our quantum result: we introduce an extra component Q that one may think of as a replacement for f_{ORT}. Q will prepare quantum states and send them to A and B. However, we do not assume secure hardware so we allow the (unbounded) adversary to remove Q and instead prepare himself the states to be sent[1]. We must therefore allow for the possibility that the protocol terminates with no output. Finally, all random choices by A or B during the protocol are leaked to the adversary as soon as they are made [2]. In this model, we obtain solutions that leak no information on the output $\boldsymbol{u}, \boldsymbol{v}$.

One such solution can be designed as follows: one can use a quantum key distribution protocol by Ekert [Eke91] with security proof by Lo and Chau [LC99] to generate a set of essentially perfect EPR pairs shared by A and B, where in particular the adversary is essentially unentangled with the state held by A and B. We then then ask A to prepare a bipartite state that is a superposition over all pairs of orthogonal vectors. Then we use the EPR pairs to teleport one part of this state to B, this only requires local quantum operations and classical communication. Finally, measuring the state on both sides will produce exactly the output we need.

However, this is not a satisfactory solution: it requires quantum storage and computation, and so is very far from what current technology allows. Also, if A and B store and compute on a quantum state for some time, it makes sense to ask what happens if the adversary can leak from the quantum state. It is not entirely clear what this should mean and how to protect against it.

We therefore propose a new protocol in which A and B measure immediately the states they receive. Although the required states may be non-trivial to prepare, this is certainly much closer to current technology than the first solution. Note that in this case, the only edge we have on the adversary is that he must choose the states to send before seeing the random choices A and B use later. This is a minimal

[1] equivalently, we could let A prepare the states and send them to B, but allow the adversary to tamper arbitrarily with the quantum communication.

[2] this is similar to the standard model for quantum key distribution, where these choices are sent on a classical channel that the adversary can observe but not tamper with.

assumption: if the adversary knows all random choices in advance, A and B have no chance to verify the states they receive.

To summarise, this gives us a way to do leakage resilient computation even if Q is replaced by an arbitrary adversarial (quantum) component, assuming (as usual) that the adversary can only leak independently from different parts of the actual computation. Of course, we do not mean to suggest that this solution is more practical than using leak-free classical gates or ciphertext banks, but we do claim that the solution is fundamentally different since we need no leak-free components at all, not even as a once-and-for-all preprocessing. Finally we also want to emphasise that by showing that the orthogonal vector problem has a quantum solution but no classical solution, we enhance our understanding of which extra power quantum communication can buy us. We believe this is of fundamental interest, independently of leakage resilience.

2 Leakage-Resilient Two-Party Computation from Orthogonal Vectors

In this section we present a leakage-resilient two-party computation protocol in a hybrid model assuming an ideal functionality producing pairs of orthogonal vectors. We take the the circuit compiler from [DF12] as our point of departure. Our goal is to "upgrade" it to two-party computation in a stronger leakage model. This is partly for added generality and partly for the sake of our negative results. We also add some protocols: for input, output and secret random bits.

We consider two-party protocols $\pi = (\pi_1, \pi_2, g)$, where each π_i is a poly-time, randomised function: the running time should be polynomial in $k\alpha$, where k is the security parameter and α is the number of activations done by the adversary (defined below). And, g is a function modelling the communication resources used by the parties. All communication happens through g. In each round g takes a secret input from each party and gives a secret output to each party, plus one public output. We say that π is a protocol for the g-hybrid model. An execution is driven by an adversary A, an interactive Turing machine, and proceeds as in Fig. 1 for security level k. We only consider A which with probability 1 will eventually execute the `done` command , and we use $\mathrm{exec}_{\pi,A}(k)$ to denote the distribution of the g in $C = (\texttt{done}, g)$.

The execution will be compared to a simulation, see Fig. 2, with a simulator $S = (S_0, S_1, S_2)$ which simulates the leakage on the protocol given just leakage on the inputs and outputs of the function. Each of S_0, S_1 and S_2 are interactive Turing machines. Each has an unbounded random tape, and they share this tape. They all have read-only access to the tape and cannot see the tape position of each other, to avoid communication. Each function L will be represented as Turing machine M_T. We let $T(A)$ denote the worst case running time of A plus the worst case running time of each M_L that it submits. Each of S_0, S_1 and S_2 are allowed polynomial running time in $T(A)$. We use $\mathrm{sim}_{f,A,S}(k)$ to denote the distribution of g in $C = (\texttt{done}, g)$.

We mainly a stricter form of simulation where the simulator must simulate the entire view of the party, and then the leakage function is applied to this view. In this so-called *full-view simulation* the command $C = (\texttt{leak}, i, L)$ is handled as follows:

Full-view simulation:

(a) Input L to S_i.
(b) Run S_i to produce a function V.
(c) Compute $s = L(V(\iota_i))$.
(d) Input s to A and S_i.
(e) Go to Step 2.

Definition 1. *We say that π is an ϵ-leakage-resilient implementation of f against $\mathcal{A}$ if there exists simulator S such that it holds for all $A \in \mathcal{A}$ that S runs in polynomial time in $T(A)$ and the statistical distance between $\mathrm{exec}_{\pi,A}(k)$ and $\mathrm{sim}_{f,A,S}(k)$ is at most $\alpha \cdot \epsilon(k)$, where α is the number of activations done by A. We say that π is a full-view ϵ-leakage-resilient implementation of f against $\mathcal{A}$ if the simulation is of the full-view form. We say that π is an ϵ-leakage-resilient implementation of f if π is an*

```
exec_{π,A}(k):

1. Initialise the states and views $\sigma_1^0 = \sigma_2^0 = \sigma^0 = \text{view}_1^0 = \text{view}_1^0 = (1^k)$, and a round counter $r = 1$ and an
   activation counter $a = 1$.
2. Run $A$ to produce a command $C$.
   – If $C = (\texttt{done}, g)$, halt with output $g$.
   – If $C = (\texttt{leak}, i, L)$, input $L(\text{view}_i^r)$ to $A$, and go to Step 2.
   – If $C = (\texttt{next-activation}, (x_1^a, x_2^a))$, do:
     (a) For $i = 1, 2$ set $\sigma_i^r = (\sigma_i^{r-1}, x_i^a)$ and $\text{view}_i^r = (\text{view}_i^{r-1}, x_i^a)$, and set $\sigma^r = \sigma^{r-1}$ and $r \leftarrow r + 1$.
     (b) For $i = 1, 2$, sample $(d_i^r, b_i^r, \sigma_i^r) \leftarrow \pi_i(\sigma_i^{r-1}; \rho_i^r)$.[a]
     (c) • If $d_1^r \neq d_2^r$ return execution-error to $A$ and go to Step 2.
         • If $d_1^r = d_2^2 = 1$, do:
             i. Let $\bar{y}_i^a = b_i^r$ for $i = 1, 2$.
            ii. Input $(\bar{y}_1^a, \bar{y}_2^a)$ to $A$.
           iii. $\text{view}_i^r = (\text{view}_i^{r-1}, \rho_i^r)$.
            iv. $r \leftarrow r + 1$ and $a \leftarrow a + 1$.
             v. Go to Step 2.
         • If $d_1^r = d_2^r = 0$, do:
             i. Sample $(c_1^r, c_2^r, c^r, \sigma^r) \leftarrow g(b_1^2, b_2^2, \sigma^{r-1})$.
            ii. $\text{view}_i^r = (\text{view}_i^{r-1}, \rho_i^r, c_i^r, c^r)$.
           iii. Input $c^r$ to $A$.
            iv. $r \leftarrow r + 1$.
             v. Go to Step 2b
     (d) Go to Step 2.
```
[a] Here ρ_i^i is the randomness used by π_i, d_i^r is a bit indicating whether party i considers the activation done and b_i^r is an output or a message for g.

Fig. 1. Real World Execution

```
sim_{f,A,S}(k):

1. Initialise the ideal states $\iota_1^0 = \iota_2^0 = \iota^0 = (1^k)$ and an activation counter $a = 1$.
2. Run $A$ to produce a command $C$.
   – If $C = (\texttt{done}, g)$, halt with output $g$.
   – If $C = (\texttt{leak}, i, L)$, do:
     (a) Input $L$ to $S_i$.
     (b) Run $S_i$ to produce $L'$ with $|L'| = |L|$.[a]
     (c) Compute $s = L'(\iota_i)$.
     (d) Input $s$ to $A$ and $S_i$.
     (e) Go to Step 2.
   – If $C = (\texttt{next-activation}, (x_1^a, x_2^a))$, do:
     (a) Sample $(y_1^a, y_2^a, y^a, \iota^r) \leftarrow f(x_1^a, x_2^a, \iota^{r-1})$.
     (b) For $i = 1, 2$, let $\iota_i^a \leftarrow (\iota_i^{a-1}, x_i^a, y_i^a, y^a)$.
     (c) Input $((y_1^a, y^a), (y_2^a, y^a))$ to $A$.
     (d) Input $y^a$ to $S_0$, $S_1$ and $S_2$.
     (e) Run $S_0$ to produce a vector $(c^{r'}, \ldots, c^r)$ and for $j = r', \ldots, r$, input $c^j$ to $A$.
     (f) $a \leftarrow a + 1$.
     (g) Go to Step 2.
```
[a] For a function L we use $|L|$ to denote the bit length of the longest output in the image of L.

Fig. 2. Simulation

ϵ-leakage-resilient implementation of f for all adversaries, i.e., without any restrictions on the leakage. Similarly we say that π is a full-view ϵ-leakage-resilient implementation of f if π is a full-view ϵ-leakage-resilient implementation of f for all adversaries.

Discussion of Model The notion of full-view simulation is inspired by the notion of leakage-oblivious simulation in [BCH12], and as in that work, the motivation is composability. However, the model in [BCH12] is an extension of the UC framework, and as such adds complications to a model which is already exquisitely complicated as it has to deal with malicious security, concurrent composition, computational security, session identifiers, and many other issues needed by a general security model. However, the focus of the present work is somewhat simpler, so we find it desirable to have a simple self-contained model aimed at just leakage-resilient, unconditionally secure, two-party computation. The hope is that the simplicity will make it useful by others.

To distinguish the non-full-view leakage-resilience from full-view leakage-resilience we will call it no-view leakage-resilience. We consider full-view simulation "the right" notion of leakage resilience, as it composes. It is true that no-view leakage resilience might be enough in some case and weaker, however, it does not necessarily compose, which makes it useless in many settings, and furthermore, as we will demonstrate in Section 2.6, under reasonable computational assumptions, no-view leakage-resilience actually implies full-view leakage-resilience for most reasonable adversary classes, so our position is that one might as well adopt full-view simulation as the base definition.

Note that A can never receive input `execution-error` in the ideal setting, hence getting input `execution-error` would allow him to perfectly distinguish, so we are essentially requiring that it never happens that $d_1^r \neq d_2^r$.

We are using local simulators, in the sense of [CV12]. This is because this gives a notion of leakage-resilience which composes. It is possible to modify our model to not use local simulators simply by giving S, S_1 and S_2 a shared tape they all can read and write. This notion of leakage-resilience, however, often does not compose well.

We have no explicit notion of a public input. However, this can be modelled by adding the input that should be considered public to the public output y, in which case it will be given to S too.

Notice also that we do not allow the adversary to ask for leakage in the middle of an activation. This is deliberately. We can handle such leakage too, but only if we give the simulator the output of the given activation, even when it simulates leakage in the middle of an activation which did not terminate. Since the simulator will be given the outputs anyway, it is easy to see that the optimal strategy of the adversary is to ask for leakage at the end of the activation, where the view of the party is strictly larger than in the middle of the activation. We build this into the model, for notational simplicity.

We can extend the model to have several hybrid functions $(g_1, \ldots, g_m)$ for communication. They can be build into one g and have each party input the index i of the function to use. If they input different indices, input `execution-error` to A and go to the next round.

2.1 Technical Results

Let $\mathcal{P}_i$ denote the class of adversaries which run for some number of activations, then make a query of the form (i, L), where L outputs a κ-bit string, and then A returns this bit string as g. We say that a class $\mathcal{A}$ allows κ bits of global leakage if $\cup_{i=1}^{2} \mathcal{P}_i \subset \mathcal{A}$.

We call a function f *stateless* if it ignores the state σ_{i-1} and always outputs empty states $\sigma_i = \perp$. We call a function f *one-shot* if it is stateless and it outputs empty outputs $y_1^a = y_2^a = y^a = \perp$ for $a < 1$.

Let g_{SEC} denote the hybrid function for secure communication, i.e., on input x_i from party i it outputs $y_{3-i} = x_i$ to the other party and gives the length of x_i as public output.

Lemma 1. *Let f be a two-party function for the model in [Can00], i.e., it takes one secret input from each parties and gives a secret output to each party. Cast it to a stateless function f for our model here by adding an empty state as input and giving an empty state as output, and an empty public output. If π is an ϵ-leakage-resilient implementation of f against $\mathcal{A}$ and ϵ is negligible and $\mathcal{A}$ allows one bit of* *global leakage and π is for the g_{SEC}-hybrid model, then π is secure in the sense of [Can00] against a computationally unbounded, semi-honest, static adversary corrupting any one party in the model with secure communication.*

Proof (sketch). If the protocol is not secure in the sense of [Can00], there exists a distinguisher D which can distinguish between a the real state of a party or a simulated state. A distinguisher outputs a single bit. We can hence use D as a leakage query, i.e., hardcode D into a leakage query L, let L run D on the state of the party and leak the output bit of D. More details in Appendix A

For a one-shot function f, let f^* be the function computing f in each activation. For a protocol π implementing a one-shot function f, we use π^* to denote the protocol which starts running π anew, with fresh randomness in each activation. We say that a protocol for a one-shot function is ϵ-leakage resilient against λ-leakage if it is ϵ-leakage resilient against the class of adversaries leaking at most λ bits from each party in the sense that the sum of $|L|$ for the leakage functions L queried on each party is upper bounded by λ.

If π_{A} is a protocol for the f_{B}-hybrid model and π_{B} is a protocol for the f_{C}-hybrid model, then we let $\pi_{\mathrm{A}}[\pi_{\mathrm{B}}/f_{\mathrm{B}}]$ denote the protocol for the f_{C}-hybrid model, which runs as π_{A} except that each call to f_{B} on (a_1, a_2) is replaced by a run of π_{B} on inputs (a_1, a_2). We say π_{A} is a protocol for the $(f_{\mathrm{B}}, q_{\mathrm{B}})$-hybrid model if it makes at most q_{B} calls to f_{B} per activation.

We will consider a generic class of adversaries which considers the execution as divided into leakage units (each a sequence of rounds) and which leaks at most λ bits on each unit. Each leakage query might leak simultaneously from several units, but the sum of bits leaked by queries involving a given unit should stay below λ for all units. To be more precise, after activation a of a protocol which has run for a total of r rounds, the adversary class is specified by a division of the rounds into disjoint intervals: $\mathrm{U}_a = \{[1, r_1], [r_1 + 1, r_2], \dots, [r_d + 1, r]\}$. The division must be computable given just the view of the adversary, specifically we assume it can be computed from just a and the knowledge of how many rounds each activation took. We assume that the division is monotone, i.e., $\mathrm{U}_{a-1} \subset \mathrm{U}_a$. We use $I \in \mathrm{U}$ to denote one of the units. We use V_i^I to denote the view of party i in unit I, i.e., the values by which the view is extended in the rounds of unit I. A leakage query is of the form $(i, \mathcal{I}, L)$, where $\mathcal{I}$ is a set of units. The leakage is computed as $L(\{(I, V_i^I)\}_{I \in \mathcal{I}})$. We restrict the adversary as follows: Initially, set counter $c_I^a = 0$ for each V_i^I. On each leakage query $(i, \mathcal{I}, L)$, update $c_i^I := c_i^I + |L|$ for all $I \in \mathcal{I}$. We require that $\max_{i,I} c_i^I \leq \lambda$. We call this the λ-tradeoff class of adversaries. Clearly this class allows λ bits of global leakage. We call it a tradeoff class as it allows global leakage, but it is at the cost of a lot of local leakage: the fewer units a leakage query touches, the less costly it is overall.

Given a λ-tradeoff class for a protocol π_{A} for the f_{B}-hybrid model, we can modify it to be a λ-tradeoff class for the protocol $\pi_{\mathrm{A}}[\pi_{\mathrm{B}}/f_{\mathrm{B}}]$, where the units are the sequences of rounds corresponding to the units of π_{A} but with each individual round expanded into several rounds, as we run π_{B} instead of evaluating f_{B}. We call this the derived tradeoff class.

Theorem 1 (Composition of Full-View Simulatability). *Let f_{B} be a one-shot function. If π_{A} is a full-view ϵ_{A}-leakage-resilient implementation of f_{A} in the $(f_{\mathrm{B}}^*, q_{\mathrm{B}})$-hybrid model against some λ-tradeoff class and π_{B} is a full-view ϵ_{B}-leakage resilient implementation of f_{B} in the $(f_{\mathrm{C}}, q_{\mathrm{C}})$-hybrid model against λ-leakage, then $\pi_{\mathrm{A}}[\pi_{\mathrm{B}}^*/f_{\mathrm{B}}^*]$ is a full-view $(\epsilon_{\mathrm{A}} + q_{\mathrm{B}}\epsilon_{\mathrm{B}})$-leakage resilient implementation of f_{A} in the $(f_{\mathrm{C}}, q_{\mathrm{B}}q_{\mathrm{C}})$-hybrid model against the derived λ-tradeoff adversary class,*

Proof (sketch). To reconstruct a view of party i in $\pi_{\mathrm{A}}[\pi_{\mathrm{B}}^*/f_{\mathrm{B}}^*]$ from just the inputs and outputs of party i, use the full-view simulator for π_{A} to reconstruct a view of party i in π_{A} from just the inputs and outputs of party i. Then for each invocation of f_{B} in the simulated view, take the corresponding inputs and outputs of f_{B} and then use the full-view simulator for π_{B} to reconstruct a view of a run of π_{B} on these inputs. There are more details in Appendix B.

2.2 Our Encodings

We first introduce the leakage-resilient encodings we use. They work over a finite field $\mathbb{F}$. A field element a is encoded as an inner product $a = \langle A_1, A_2 \rangle$, where party i holds A_i. The length n of the vectors A_i is a parameter of the scheme. We later return to how to set n to obtain a given security level.

$\mathcal{E}^n(a)$:

1. Sample uniformly random $(A_1, A_2) \in \mathbb{F}^n \times \mathbb{F}^n$ for which $\langle A_1, A_2 \rangle = a$.
2. Output (A_1, A_2).

$\mathcal{A}^n(a)$:

1. Sample uniformly random $(A'_1, A'_2) \in \mathbb{F}^{n-1} \times \mathbb{F}^{n-1}$.
2. $(A_1, A_2) := ((A'_1, 1), (A'_2, a - \langle A'_1, A'_2 \rangle))$.
3. Output (A_1, A_2).

Throughout this section, when we write an output as (x, y), we mean that x is given to party x and y is given to party 2. Similarly for inputs.

We will use that $\mathcal{E}^n(a)$ and $\mathcal{A}^n(a)$ are leakage-resilient encodings, and we now define leakage-resilience of an encoding. We compare leaky encodings of different messages. In the definition we will need an oracle $\mathcal{O}((v_1, v_2), \cdot)$, where the v_i are bit-strings. On input (i, L), where $i \in \{1, 2\}$ and L is a function, it returns $L(v_i)$. We might put various restrictions on the queries that might be submitted, but we will phrase that by restricting the entity making the queries. An adversary A against an encoding scheme is an interactive Turing machine. The leakage setting proceeds as follows. First sample $(A_1, A_2) \leftarrow \mathcal{E}^n(a)$. Then give A oracle access to $\mathcal{O}((A_1, A_2), \cdot)$ and run A to produce output V, where V is the view of A, i.e., the list of queries to and replies from the oracle. We use $A^{\mathcal{O}(\mathcal{E}^n(a), \cdot)}$ to denote the resulting distribution on V.

Definition 2. *We say that $\mathcal{E}$ is (λ, ϵ)-leakage resilient, if for all A leaking at most λ bits on each A_i and for all $a, b \in \mathbb{F}$ the distributions $(a, b, A^{\mathcal{O}(\mathcal{E}^n(a), \cdot)})$ and $(a, b, A^{\mathcal{O}(\mathcal{E}^n(b), \cdot)})$ have statistical distance at most ϵ.*

Lemma 2. *Let k be a security parameter. If $n \geq 4k + 10 + 4\log_2 |\mathbb{F}|$ and $|\mathbb{F}| \geq n$, then $\mathcal{E}^n$ is $(\lambda(n), 2^{-k})$-leakage resilient, where $\lambda(n) = 0.25n|\mathbb{F}| - k$.*

Proof. Lemma 1 in [DF11], using $\delta = 1/2$, $m = 1$ and $\gamma = 2^{-k-1-\log_2(|\mathbb{F}|)}$. □

Lemma 3. *Let k be a security parameter. If $n \geq 4(k+1) + 10 + 4\log_2 |\mathbb{F}| + 1$ and $|\mathbb{F}| \geq n$, then $\mathcal{A}^n$ is $(\lambda(n), 2^{-k})$-leakage resilient, where $\lambda(n) = 0.25n|\mathbb{F}| - k - 1$.*

Proof. Let V denote the distribution on a obtained by sampling $(A'_1, A'_2) \leftarrow \mathbb{F}^{n-1} \times \mathbb{F}^{n-1}$ uniformly at random and letting $a \leftarrow \langle A'_1, A'_2 \rangle$. Sample $(B'_1, B'_2) \leftarrow \mathcal{E}^{n-1}(a)$ for $a \leftarrow V$. Then (B'_1, B'_2) is uniform on $\mathbb{F}^{n-1} \times \mathbb{F}^{n-1}$. We use this to turn an adversary A for $\mathcal{A}^n$ into an adversary B for $\mathcal{E}^{n-1}$. The reduction B_c will make use of a parameter $c \in \mathbb{F}$. It picks $a \in \mathbb{F}$ according to V and picks $b \in \mathbb{F}$ uniformly at random. Then it asks for an encoding of a or b, using $\mathcal{E}^{n-1}$. This samples (C'_1, C'_2) and B_c gets oracle access to leakage on the C'_i. Define $c_1 = 1$ and $c_2 = c - b$ and $C_i = (C'_i, c_i)$. The reduction B_c can simulate a leakage query (i, L) from A on C_i by a leakage query (i, L') on C'_i, where $L'(\cdot) := L((\cdot, c_i))$. If B_c got an encoding of b, then (C_1, C_2) is by construction distributed exactly as $\mathcal{A}^n(c)$, i.e.,

$$(c_1, c_2, A^{\mathcal{A}^n(c_1)}) = (c_1, c_2, B_{c_1}^{\mathcal{E}^{n-1}(b)}) ,$$

$$(c_1, c_2, A^{\mathcal{A}^n(c_2)}) = (c_1, c_2, B_{c_2}^{\mathcal{E}^{n-1}(b)}) .$$

If B_c got an encoding of a, then the distribution of (C_1, C_2) is by construction distributed independent of c. So,

$$(c_1, c_2, B_{c_1}^{\mathcal{E}^{n-1}(a)}) = (c_1, c_2, B_{c_2}^{\mathcal{E}^{n-1}(a)}) .$$

It follows that the distance between $(c_1, c_2, A^{\mathcal{A}^n(c_1)})$ and $(c_1, c_2, A^{\mathcal{A}^n(c_2)})$ can be no more than twice the distance between $B_{c_2}^{\mathcal{E}^{n-1}(a)}$ and $B_{c_2}^{\mathcal{E}^{n-1}(b)}$. Then apply Lemma 2. □

In the following we pick the parameters according to Lemma 3 such that both encodings are $(\lambda(n), 2^{-k})$-leakage resilient for $\lambda(n) = 0.25n|\mathbb{F}| - k - 1$.

2.3 The Protocol

We will present our protocol by giving full-view $O(2^{-k})$-leakage-resilient implementations of some simple functions first. Later we then put them together, using that full-view leakage-resilience composes. We first list the functions, see Fig. 3, and then discuss how to implement them, see Fig. 4.

$(S_1, S_2) \leftarrow f_{\text{ORT}}{}^n()$:

1. Input: None.
2. Sample $(S_1, S_2) \leftarrow \mathcal{E}^n(0)$.
3. Output (S_1, S_2).

$(B_1, B_2) \leftarrow f_{\text{REF}}(A_1, A_2)$:

1. Input: Encoding $(A_1, A_2) \in \mathbb{F}^m \times \mathbb{F}^m$ for some m.
2. Sample $(B_1, B_2) \leftarrow \mathcal{E}^m(\langle A_1, A_2 \rangle)$.
3. Output (B_1, B_2).

$(C_1, C_2) \leftarrow f^n_{\text{AMULT}}((A_1, B_1), (A_2, B_2))$:

1. Input: Encodings $(A_1, A_2) \in \mathbb{F}^m \times \mathbb{F}^m$ and $(B_1, B_2) \in \mathbb{F}^l \times \mathbb{F}^l$, with $l, m \geq n$.
2. Sample $(C_1, C_2) \leftarrow \mathcal{A}^n(\langle A_1, A_2 \rangle \langle B_1, B_2 \rangle)$.
3. Output (C_1, C_2).

$(B_1, B_2) \leftarrow f_{\text{NEG}}(A_1, A_2)$:

1. Input: Encoding (A_1, A_2), where the last element of A_1 is 1.
2. Let $B_1 = A_1$.
3. Let $B_2 = (0, 0, \ldots, 0, 0, 1) - A_2$.
4. Output (B_1, B_2).

$(B_1, B_2) \leftarrow f_{\text{MINUS}}(A_1, A_2)$:

1. Input: Encoding (A_1, A_2).
2. Let $B_1 = A_1$.
3. Let $B_2 = -A_2$.
4. Output (B_1, B_2).

$(B_1, B_2) \leftarrow f_{\text{PLUSONE}}(A_1, A_2)$:

1. Input: Encoding (A_1, A_2).
2. Let $B_1 = (A_1, 1)$.
3. Let $B_2 = (A_2, 1)$.
4. Output (B_1, B_2).

$(C_1, C_2) \leftarrow f^n_{\text{ANAND}}((A_1, B_1), (A_2, B_2))$:

1. Input: Encoding (A_1, A_2) and (B_1, B_2).
2. Sample $(C_1, C_2) \leftarrow \mathcal{A}^n(1 - \langle A_1, A_2 \rangle \langle B_1, B_2 \rangle)$.
3. Output (C_1, C_2).

$(A_1, A_2) \leftarrow f^n_{\text{RAN}}()$:

1. Input: None.
2. Sample uniformly random $a \in \mathbb{F}$.
3. Sample $(A_1, A_2) \leftarrow \mathcal{E}^n(a)$.
4. Output (A_1, A_2).

$(A_1, A_2) \leftarrow f_{\text{RANBIT}}{}^n()$:

1. Input: None.
2. Sample uniformly random $a \in \{0, 1\}$.
3. Sample $(A_1, A_2) \leftarrow \mathcal{E}^n(a)$.
4. Output (A_1, A_2).

$(A_1, A_2) \leftarrow f^{1,n}_{\text{IN}}(a, 0)$:

1. Input: $a \in \mathbb{F}$ to party 1.
2. $(A_1, A_2) := ((a, 0^{n-1}), 1^n)$.
3. Output (A_1, A_2).

$(B_1, B_2) \leftarrow f^2_{\text{OUT}}(A_1, A_2)$:

1. Input: Encoding (A_1, A_2).
2. Let $a = \langle A_1, A_2 \rangle$.
3. Output $(0, a)$.

Fig. 3. The Functions

In our protocol we will assume that we have an oracle producing encodings of 0. We formalise this as the f_{ORT} two-party function. The function f_{REF} is for refreshing an encoding. The function f_{AMULT} is for multiplication of encoded values. It outputs an alternative encoding. The function f_{NEG} is for negating an encoded bit. Note that for f_{NEG} we have $\langle B_1, B_2 \rangle = \langle A_1, (0, 0, \ldots, 0, 0, 1) \rangle - \langle A_1, A_2 \rangle = 1 - \langle A_1, A_2 \rangle = 1 - a$. In particular, if a is a bit, 0 or 1, then we negate it. The function f_{MINUS} is changing the sign of an encoded value, note in particular that $\langle B_1, B_2 \rangle = -\langle A_1, A_2 \rangle$. The function f_{PLUSONE} is for adding 1 to an encoded value, note in particular that $\langle B_1, B_2 \rangle = \langle A_1, A_2 \rangle + 1$. The function f_{ANAND} is for computing the NAND of encoded values. It outputs an alternative encoding. The function f_{RAN} is for sampling an encoding of a random element, and the function f_{RANBIT} is for sampling an encoding of a random bit. The function $f^{i,\cdot}_{\text{IN}}$ is for taking input from party i. We give the function for input from party

π^n_{AMULT}:

1. Input: (A_1, A_2) and (B_1, B_2).
2. $(D_1, D_2) := (A_1 \otimes B_1, A_2 \otimes B_2)$.
3. $(E_1, E_2) \leftarrow f_{\text{REF}}(D_1, D_2)$.
4. Compute $((E'_1, E''_1), (E'_2, E''_2))$ such that $E_i = E'_i E''_i$ and $|E'_1| = |E'_2| = n - 1$.
5. Party 1 sends E''_1.
6. Party 2 computes $d := \langle E''_1, E''_2 \rangle$.
7. $(C_1, C_2) := ((E'_1, 1), (E'_2, d))$.
8. Output (C_1, C_2).

π^n_{ANAND}:

1. Input: (A_1, A_2) and (B_1, B_2).
2. $(D_1, D_2) \leftarrow f_{\text{AMULT}}((A_1, B_1), (A_2, B_2))$.
3. $(C_1, C_2) \leftarrow f_{\text{NEG}}(D_1, D_2)$.
4. Output $[c] = (C_1, C_2)$.

π^n_{RAN}:

1. Party i samples uniformly random $A_i \in \mathbb{F}^n$.
2. Output (A_1, A_2).

π^2_{OUT}:

1. Input: $[a] = (A_1, A_2)$.
2. $(B_1, B_2) \leftarrow f_{\text{REF}}(A_1, A_2)$.
3. Party 1 sends B_1.
4. Party 2 computes $a = \langle B_1, B_2 \rangle$
5. Output $(0, a)$.

${\pi_{\text{RANBIT}}}^n$:

1. $(R_1, R_2) \leftarrow f^n_{\text{RAN}}()$.
2. $(S_1, S_2) := f_{\text{PLUSONE}}(f_{\text{MINUS}}(R_1, R_2))$.
3. $(T_1, T_2) \leftarrow f^n_{\text{AMULT}}((R_1, S_1), (R_2, S_2))$.
4. $(0, t) \leftarrow f^2_{\text{OUT}}(T_1, T_2)$.
5. Party 2 sends t.
6. If $t = 0$, then output (R_1, R_2). Otherwise, go to step 1.

π^n_f(we assume that f is given as a Boolean circuit consisting of NAND gates):

1. For each input bit x of party 1, sample $(X_1, X_2) \leftarrow f^{1,n}_{\text{IN}}(x, 0)$ and store (X_1, X_2).
2. For each input bit x of party 2, sample $(X_1, X_2) \leftarrow f^{2,n}_{\text{IN}}(0, x)$ and store (X_1, X_2).
3. For each random bit r, sample $(R_1, R_2) \leftarrow {f_{\text{RANBIT}}}^n()$ and store (R_1, R_2).
4. For each NAND gate, in sequence, retrieve the stored encodings (A_1, A_2) and (B_1, B_2) of the input bits and replace them by storing fresh encodings $(A'_1, A'_2) \leftarrow f_{\text{REF}}(A_1, A_2)$ and $(B'_1, B'_2) \leftarrow f_{\text{REF}}(B_1, B_2)$. Then sample $(C_1, C_2) \leftarrow f^n_{\text{ANAND}}((A_1, B_1), (A_2, B_2))$, and store (C_1, C_2).
5. For each output bit of party 1, retrieve the encoding (Y_1, Y_2) and compute $(y, 0) = f^1_{\text{OUT}}(Y_1, Y_2)$. Then replace (Y_1, Y_2) by a fresh encoding $(Y'_1, Y'_2) \leftarrow f_{\text{REF}}(Y_1, Y_2)$.
6. For each output bit of party 2, retrieve the encoding (Y_1, Y_2) and compute $(0, y) = f^2_{\text{OUT}}(Y_1, Y_2)$. Then replace (Y_1, Y_2) by a fresh encoding $(Y'_1, Y'_2) \leftarrow f_{\text{REF}}(Y_1, Y_2)$.

Fig. 4. The Protocols

1. The corresponding function for party 2 is symmetric. The function f^i_{OUT} is for giving output to party i. We give the function for input from party 2. The corresponding function for party 1 is symmetric.

We now describe full-view $O(2^{-k})$-leakage-resilient implementations of these functions. In [DF12] a protocol π_{REF} is given which implements f_{REF} in the f_{ORT}-hybrid model, and which is easily verified to be full-view 0-leakage-resilient.[3] We can construct a full-view 0-leakage-resilient protocol π_{NEG} for f_{NEG} by noting that the parties can compute the function without randomness or communication: any deterministic protocol which does not communicate is trivially full-view 0-leakage-resilient. Similarly for f_{IN}, f_{MINUS} and f_{PLUSONE}.

The protocol π^n_{AMULT} is a full-view 2^{-k}-leakage-resilient implementation of f^n_{AMULT}. It is a modified version of a protocol from [DF12]. The views can be perfectly reconstructed (locally given just the outputs of the protocol and common randomness): the reconstructor/simulator is given (A_i, B_i, C_i). If $i = 1$ it samples E''_1 uniformly at random using the common randomness and parses C_1 as $(E'_1, 1)$. Then it lets $E_1 = E'_1E''_1$. There are no other values to reconstruct. If $i = 2$ the reconstructor parses C_2 as (E'_2, d). Then it samples E''_1 as above, using the common randomness, and then it samples E''_2 uniformly at random such that $\langle E''_1, E''_2\rangle = d$. It lets $E_2 = E'_2E''_2$. Note that $\langle E_1, E_2\rangle = \langle E'_1, E'_2\rangle + \langle E''_1, E''_2\rangle = (\langle C_1, C_2\rangle - d) + d = \langle C_1, C_2\rangle = \langle A_1, A_2\rangle\langle B_1, B_2\rangle$ as it should be. It is straightforward to check that it is a random vector with this property. However, the output distribution of the protocol is not perfectly the same as for the function. For $n > 5$, there is a statistical distance of up to 2^{-k}. Let $c = \langle A_1, A_2\rangle\langle B_1, B_1\rangle$. Both f^n_{AMULT} and π^n_{AMULT} produce $(C_1, C_2) = ((C'_1, 1), (C'_2, d))$ with $d = c - \langle C'_1, C'_2\rangle$, so it is enough to bound the distance between the distribution of (C'_1, C'_2) in f^n_{AMULT} and (E'_1, E'_2) in π^n_{AMULT}. We have that (C'_1, C'_2) is uniform. It is easy to see that (E'_1, E'_2) is 2^{-k}-close to uniform. Namely, assume without loss of generality that $m = l = n$. Let H be the distribution of $h \in \mathbb{F}$ obtained by computing $h \leftarrow \langle H_1, H_2\rangle$ for uniformly random $H_1, H_2 \in \mathbb{F}^{n^2}$. If we sample $h \leftarrow H$ and then sample $(H_1, H_2) \leftarrow \mathcal{E}^{n^2}(h)$, we would get the uniform distribution back. So, if we sample $(H_1, H_2) \leftarrow \mathcal{E}^{n^2}(h)$ and then truncate to $(H_1[1, n-1], H_2[1, n-1])$ we get a uniform distribution on $\mathbb{F}^{n-1} \times \mathbb{F}^{n-1}$. This is the distribution of (C'_1, C'_2). Now instead sample $(E_1, E_2) \leftarrow \mathcal{E}^{n^2}(c)$ and take $(E_1[1, n-1], E_2[1, n-1])$. This is the distribution of (E'_1, E'_2). Since $n - 1 \leq \lambda(n^2)$ when $n > 5$ we have that $(H_1[1, n-1], H_2[1, n-1])$ and $(E_1[1, n-1], E_2[1, n-1])$ are tolerable leakage on a sharing of length n^2 according to Lemma 2.

The protocol π^n_{ANAND} is clearly a full-view 0-leakage-resilient implementation of f^n_{ANAND}. The protocol π^n_{RAN} is a full-view 2^{-k}-leakage-resilient implementation of f^n_{RAN}. It is trivially possible to perfectly reconstruct the view of the protocol given the outputs of the protocol, and it can be seen that the distribution of the output is statistically 2^{-k}-close to the output of f^n_{RAN}. All there is to verify is that the inner product of uniformly random $A_1, A_2 \in \mathbb{F}^n$ is 2^{-k}-close to uniform.

The protocol π^2_{OUT} is a protocol for output in the f_{REF}-hybrid model: The protocol is full-view 2^{-k}-leakage resilient: to reconstruct party 1 sample B_1 uniformly at random from the common randomness. To reconstruct party 2 sample B_1 as above, and then sample B_2 uniformly, except that $\langle B_1, B_2\rangle = a$.

The protocol π_{RANBIT} is a protocol for the $(f_{\text{OUT}}, f_{\text{RAN}}, f_{\text{AMULT}})$-hybrid model using rejection sampling to sample a random encoding of a random bit: It has the same output distribution as ${f_{\text{RANBIT}}}^n$: If $\langle R_1, R_2\rangle = r$, then $t = r(1 - r)$. So $t = 0$ iff $r \in \{0, 1\}$. So, if $t = 0$ then r is uniform on $\{0, 1\}$. The protocol is full-view 0-leakage-resilient. Reconstruct the rejected prefix of the runs by using the common randomness to do random runs. When the first of these runs succeed, throw it away and use the rejected prefix. To reconstruct the successful run, start from the output R_i. Then compute S_i as in the protocol, which is possible as there is no communication involved. Then use the common randomness to sample a random encoding (T_1, T_2) of 0 and use T_i. Then let $t = 0$.

The protocol π^n_f securely computes one activation of a Boolean circuit for f consisting of NAND gates. Besides this it has input wires and output wires, which specify who is to provide the input respectively learn the output. Finally, it has random wires, which are assigned a uniformly random bit. We

[3] The protocol in [DF12] actually generates an encoding where the first element of the first half is invertible. However, this is done by rejection sampling, and by simply removing the step where the protocol reruns when the element is 0, we get a protocol for our formulation of the refreshing.

assume that the circuit computes $f(x_1, x_2, r)$. I.e., if the bits of x_i are placed on the input wires on party i and the random wires are assigned uniformly random bits, then the output on the wires of party i will be some y_i such that (x_1, x_2, y_1, y_2) is distributed as $(x_1, x_2, f(x_1, x_2))$. The description of the protocol can be extended to passing an encoded state between activations but for notational convenience we keep the description to one activation of a one-shot function. The protocol is clearly correct. Below we will further analyse the protocol and some consequences.

2.4 Analysis

We will consider a class of adversaries which leaks at most $\lambda(n)/2$ bits on each of the steps of π_f^n. Each query might leak simultaneously from several steps, but the sum of the number of bits leaked from queries where a given step is involved should stay below $\lambda(n)/2$ for all steps. To be more precise, assume each step is uniquely indexed by some identifier I. We use V_i^I to denote the view of party i of the step identified by I. For an input step of party i let $V_i^I = (x, X_i)$. For an input step of party $3-i$ let $V_{3-i}^I = (X_{3-i})$. For a random bit step let $V_i^I = (R_i)$. For a NAND step let $V_i^I = (A_i, B_i, A_i', B_i', C_i)$. For an output step of party i let $V_i^I = (y, Y_i)$. For an output step of party $3-i$ let $V_{3-i}^I = (Y_{3-i})$. A leakage query is of the form $(i, \mathcal{I}, L)$, where $\mathcal{I}$ is a set of identifier. The leakage is computed as $L(\{(I, V_i^I)\}_{I \in \mathcal{I}})$. We restrict the adversary as follows: Initially, set counter $c_i^I = 0$ for each V_i^L. On each leakage query $(i, \mathcal{I}, L)$, update $c_i^I := c_i^I + |L|$ for all $I \in \mathcal{I}$. We require that $\max_{i,I} c_i^I \leq \lambda(n)/2$. We call this the $\lambda/2$-class of adversaries. Clearly this class allows one bit of global leakage. Note that this class is slightly stronger than a $\lambda/2$-tradeoff class as some of the intervals overlap: the outputs of some steps occur as inputs to other steps, and hence we can leak on these values from several units.

Theorem 2. *Let c denote the size of the circuit, counting input wires, output wires, random wires and gates. Then π_f^n is a full-view $O(c2^{-k})$-leakage resilient implementation of f against the $\lambda/2$-class of adversaries.*

Proof. The intermediary values in the protocol are represented as random encodings and there are at most $O(1)$ fresh encodings generated by each step. The simulator will use common randomness to simulate the half of these encodings seen by the simulated party by the corresponding half of a random encoding on 0. Then use that the encodings are leakage-resilient plus a hybrid argument. To be able to do the hybrid argument we need that we leak at most λ bits form each encoding. Since each encoding is touched by up to two steps, we therefore have to divide the leakage bound by 2.

We describe our simulator S in a bit more detail. For an input step of party 1, define the encoding to be a function $(X_1, X_2)(\mathrm{x}) = ((\mathrm{x}, 0^{n-1}), 1^n)$ of the input bit, and define a function of x_i, the input of party i, as $V_1^I(x_i) = (\mathrm{x}, X_i(\mathrm{x}))$, where x is assigned the appropriate bit from x_i. Let $V_2^I = (X_2)$. Similar for input for party 2. For each random bit step sample $(R_1, R_2) \leftarrow \mathcal{E}^n(0)$ and let $V_i^I = (R_i)$. For each NAND step, in the order in which they are computed, retrieve the encodings (A_1, A_2) and (B_1, B_2) simulated above. Then sample $(A_1', A_2') \leftarrow \mathcal{E}^n(0)$, $(B_1', B_2') \leftarrow \mathcal{E}^n(0)$ and $(C_1, C_2) \leftarrow \mathcal{A}^n(0)$. Let $V_i^I = (A_i, B_i, A_i', B_i', C_i)$. For each output step of party i, retrieve the encoding (Y_1, Y_2) simulated above. Sample $(Y_1', Y_2') \leftarrow \mathcal{E}^n(0)$. Then define a function of y_i, the output of party i, as $V_i^I(y_i) = (\mathrm{y}, Y_i, Y_i')$ where y is assigned the appropriate bit from y_i. Let $V_{3-i}^I = (Y_{3-i}, Y_{3-i}')$. This specifies how to simulate the entire view. The value returned to the adversary is on $(i, \mathcal{I}, L)$ is $L(\{(I, V_i^I(x_i, y_i))\}_{I \in \mathcal{I}})$. Note that each step in the protocol generates at most three new encodings. Hence there are at most $3c$ distinct encodings generated by the protocol. The only difference between the simulated setting and the real setting is that some random encodings have been replaced by random encodings of 0. Observe then that our protocol has the property that each encoding is touched by at most two steps: The step that generates it as output, and the step that takes it as input. The reason is that whenever a step takes an encoding as input, it will replace it by a fresh encoding to be used by the next step which needs an encoding of the same value. Hence, if we identify each encoding (A_1^J, A_2^J) by an identifier J and keep a counter d_i^J which keeps track of how many bits could have been leaked from A_i^L, defined as the c_i^J for steps, then $\max_{i,I} d_i^I \leq \lambda(n)$. Hence the leakage from each encoding is within the tolerated leakage bound. So, by a hybrids argument we can replace each random encoding of 0 by a random encoding of the correct value, increasing in each step the distance by at most 2^{-k} by Lemma 2 and Lemma 3.

We now consider the protocol $\pi_f^n[{\pi_{\text{RanBit}}}^n/{f_{\text{RanBit}}}^n, \pi_{\text{ANand}}^n/f_{\text{ANand}}^n, \pi_{\text{Out}}/f_{\text{Out}}]$, where all calls to functions have been replaced by implementations such that we get a protocol for the f_{ORT}-hybrid model. We can extend the $\lambda/2$-class to $\pi_f^n[{\pi_{\text{RanBit}}}^n/{f_{\text{RanBit}}}^n, \pi_{\text{ANand}}^n/f_{\text{ANand}}^n, \pi_{\text{Out}}/f_{\text{Out}}]$. We use the same definition as before, except that we let the view of a step include all the intermediary values on the sub-protocols the step is running. The following theorem follows exactly as the composability of full-view simulation, as each replaced function is embedded in one leakage unit.

Theorem 3. *Let c denote the size of the circuit, counting input wires, output wires, random wires and gates. Then $\pi_f^n[{\pi_{\text{RanBit}}}^n/{f_{\text{RanBit}}}^n, \pi_{\text{ANand}}^n/f_{\text{ANand}}^n, \pi_{\text{Out}}/f_{\text{Out}}]$ is a full-view $O(c2^{-k})$-leakage resilient implementation of f against the derived $\lambda/2$-class of adversaries in the f_{ORT}-hybrid model.*

Similarly we can consider replacing f_{ORT} by a full-view leakage-resilient implementation π_{ORT}. This is mainly for later proving certain such implementations impossible.

Theorem 4. *If there exist a full-view* negl*-leakage resilient implementation π_{ORT} of f_{ORT} secure against an adversary leaking at most $\lambda(n)/2$ bits on each share, then $\pi_f^n[{\pi_{\text{RanBit}}}^n/{f_{\text{RanBit}}}^n, \pi_{\text{ANand}}^n/f_{\text{ANand}}^n, \pi_{\text{Out}}/f_{\text{Out}}, \pi_{\text{ORT}}/f_{\text{ORT}}]$ is a full-view* negl*-leakage-resilient implementation of f against the derived $\lambda/2$-class of adversaries.*

2.5 Negative Results

It follows directly from the above theorem that there exists no full-view negl-leakage resilient implementation π_{ORT} of f_{ORT}, as we would otherwise have for all f a protocol π securely realizing f in the sense of [Can00] against a computationally unbounded, semi-honest, static adversary corrupting any one party in the model with secure communication. It is well known that we cannot have such a thing.

We now prove that also unconditionally secure circuit compilation is impossible in the plain model with only secure communication. By circuit compilation we mean that we compute a function with no secret inputs or outputs, so the output is known to all parties, but there might be a hidden secret state, like a key. To simplify matters, consider a function g_{PRG} which ignores the inputs from the parties and the state, it is only a function of the first k bits of its own random tape, r say. Furthermore, it gives no secret outputs, we can formally set $y_1 = y_2 = \perp$. As public output it gives $y = f(r)$ for a pseudo-random generator f. We will show that it is impossible to compute such a function in the plain model, at least with an implementation which is perfectly correct.

When we prove impossibility of circuit compilation, it seems we have to give up on getting an unconditional result. Instead, we will show impossibility under reasonable computational assumptions, namely those used in [Kil92]. We will assume there exists a collision resistant hash function and a succinct proof of knowledge with soundness $1/2$, where the communication is $\tilde{O}(k)$, see e.g. [Kil92]. It might seem odd that we need computational assumptions to prove that an unconditionally secure protocol is impossible. However, for an intuition why this might be needed, consider the case where P=NP. In that case it might be possible to compute all functions with no secret inputs and outputs in a leakage resilient manner,[4]

Theorem 5. *Under the computational assumptions in [Kil92], there exists no perfectly correct and* negl*-leakage-resilient implementation of g_{PRG} against an adversary class allowing global leakage of $\tilde{O}(k)$ bits, in the plain model with only secure communication.*

[4] Consider, e.g., a function like g_{PRG} but for an easy to invert f. If one can compute r from $s = f(r)$ efficiently, then one might just implement this function by letting party 1 sample r uniformly at random and send $s = f(r)$ to party 2 – technically this is leakage resilient. If, on the other hand, f is a one-way function, then sampling s in a leakage resilient manner appears much harder.

Proof. We can assume that the length of the output of the collision resistant hash function H plus the communication of the succinct proof of knowledge is within the leakage-bound by simply setting n high enough. We will prove that $\pi_g^n[\pi_{\text{RANBIT}}{}^n/f_{\text{RANBIT}}{}^n, \pi^n_{\text{ANAND}}/f^n_{\text{ANAND}}, \pi_{\text{OUT}}/f_{\text{OUT}}, \pi_{\text{ORT}}/f_{\text{ORT}}]$ has the property that there exist an expected poly-time extractor X which given just the output $s = f(r)$ will produce, except with negligible probability, randomness r_1 and r_2 for party 1 respectively party 2, such that running $\pi_g^n[\pi_{\text{RANBIT}}{}^n/f_{\text{RANBIT}}{}^n, \pi^n_{\text{ANAND}}/f^n_{\text{ANAND}}, \pi_{\text{OUT}}/f_{\text{OUT}}, \pi_{\text{ORT}}/f_{\text{ORT}}]$ with this randomness will result in a run of the protocol which has output s. Since our protocol is perfectly correct, it follows that if s is an output of $\pi_g^n[\pi_{\text{RANBIT}}{}^n/f_{\text{RANBIT}}{}^n, \pi^n_{\text{ANAND}}/f^n_{\text{ANAND}}, \pi_{\text{OUT}}/f_{\text{OUT}}, \pi_{\text{ORT}}/f_{\text{ORT}}]$ then it is also an output of f. Hence X is an almost perfect distinguisher of outputs from f from random strings, which breaks the pseudo-randomness.

We conclude by sketching how X works. It starts with a leakage adversary, and it works as follows: Given leakage access to $\pi_g^n[\pi_{\text{RANBIT}}{}^n/f_{\text{RANBIT}}{}^n, \pi^n_{\text{ANAND}}/f^n_{\text{ANAND}}, \pi_{\text{OUT}}/f_{\text{OUT}}, \pi_{\text{ORT}}/f_{\text{ORT}}]$ it will ask each party to leak a hash $h = H(M)$ of the transcript M of all messages exchanged with the other party. These hashes will of course be the same, as we assume the parties have noiseless communication and thus the same transcripts. Then it will ask party i to leak a proof of knowledge of a string r_i and a communication transcript M' with the properties that $H(M') = h$ and that if party i runs the protocol with random string r_i and the incoming message in M' it will produce the outgoing messages in M'. The proof might be interactive, but this is not a problem as it can be simulated by one leakage query per round in the protocol. The communication of the proof is within the leakage bound, so by assumption we can simulate the view of X given just s. Call the simulator P. We can assume we have P. By assumption the simulation cannot be distinguished from the real execution, so in particular the proof will be accepting except with negligible probability, which is bounded far away from the knowledge error of $1/2$. So, from black-box access to $P(s)$ we can extract the proofs in poly-time. So, from the soundness of the proofs we can extract r_1, M_1' from party 1 and r_2, M_2' from party 2. By the soundness of the proof and collision resistance, we have that $H(M_1') = h = H(M_2')$ except with negligible probability. By soundness we have except with negligible probability that r_1 and r_2 are consistent with M_1' respectively M_2'. So, we can let the output of X be (r_1, r_2).

Note that the same proof technique can be applied to show impossibility of protocols satisfying the standard only-computation-leaks definition, i.e., only independent leakage from each unit is allowed (and the protocol is not necessarily derived from a 2-party protocol).

More specifically, we assume that: 1) Each unit which exchanges data with c other units should allow adaptive leakage of $O(c)\tilde{O}(k)$ bits. Furthermore, 2) the protocol should be perfectly correct. Finally, 3) the protocol should allow to compute a PRG on a secret state and give the expanded string as public output and 4) the leakage should be efficiently simulatable given just this output.

In that case we build an adversary that leaks from each unit a hash of the communication with each of the units with which it communicates, plus a succinct proof of knowledge of the hashed transcripts and a proof of knowledge of a random tape consistent with the hashed transcripts. Since these leaked values can be simulated given just the public inputs and outputs, we can turn the efficient simulator into an efficient and successful prover in the argument system. Then we can extract all the proofs, and we will get a consistent run of the protocol as in the above argument, which in particular will allow to verify that the output is a pseudo-randomly generated string. So we have

Corollary 1. *There exists no protocol for computing g_{PRG} satisfying conditions 1–4 above.*

The protocol from [GR12] does allow leakage of $\tilde{O}(k)$ bits from each unit but is not of the specified form because it assumes that the ciphertext bank is already given. However, the above corollary rules out implementing the ciphertext bank perfectly correctly from scratch in a way suitable for plugging into [GR12], as this would result in a protocol contradicting the corollary[5].

[5] Note that it does not matter how many units communicate in the construction of [GR12]: we could choose the security parameter for g_{PRG} and the hash function so small (but polynomially related to k) such that even if all units communicate, we can still leak enough for the proof to go through.

2.6 On Full-View Simulatability as a General Notion

We then return to the issue of full-view simulatability as "the right" notion of simulation-based leakage-resilience. Note that under the computational assumptions used by [Kil92], if there exists a negl-leakage resilient implementation π of f secure against an adversary class allowing λ bits of leakage, then in most cases there also exists a full-view negl-leakage resilient implementation π of f secure against $\lambda/\tilde{O}(k)$ bits of leakage. For each leakage query (i, L) submit a leakage query (i, L') where $L'(\mathrm{view}_i^r)$ first leaks $h = H(\mathrm{view}_i^r)$ and a proof of knowledge of view_i^r such that $h = H(\mathrm{view}_i^r)$. Then it leaks $v = L(\mathrm{view}_i^r)$ plus a proof of knowledge of view_i^r such that $v = L(\mathrm{view}_i^r)$ and $h = H(\mathrm{view}_i^r)$. Given a simulator S for this L' we can construct a full-view constructor V which given just the ideal state will use S to simulate $h = H(\mathrm{view}_i^r)$ and the proof of knowledge of view_i^r. Since this has to be indistinguishable from the real values, the proof accepts with probability essentially 1. Since the simulation is efficient given the ideal state, the view constructor can efficiently run the simulator. So, it can extract the simulator and compute view_i^r such that $h = H(\mathrm{view}_i^r)$. Then it outputs view_i^r. Then the full-view simulation game gives $v = L(\mathrm{view}_i^r)$ to the adversary. This value v will be indistinguishable from the value v' that the simulator S would have returned, as S can prove knowledge of view_i^r such that $v' = L(\mathrm{view}_i^r)$ and $h = H(\mathrm{view}_i^r)$, and H is collision resistant. Since S is a simulator demonstrating security the value v' returned by S would have fooled the adversary, and hence will also the value $v = L(\mathrm{view}_i^r)$ given to the adversary by the full view simulator.

There are many details to consider to make this argument work and many ways around it. E.g., the interactive proofs require that each leakage query turns into many leakage queries, so an unbounded number of adaptive leakage queries should be allowed. Also, $h = H(\mathrm{view}_i^r)$ is a global leakage query, which might not be allowed at all, and might make the leakage bound deteriorate quickly. However, one can counter some of these objections by using, e.g., non-interactive proofs and leaking only hashs of the part of the view accessed by L plus hashs showing it consistent with previously accessed parts. Under all circumstances, the moral argument here is that allowing non-trivial leakage means that you might as well consider full-view leakage simulation, as it is probably what you can prove anyway, and it will give you composition for free.

3 Leakage Resilient Computation From an Imperfect Source of Orthogonal Vectors

In this section we consider again a Boolean circuit C that computes a two-party function $(y_1, y_2) = f_C(x_1, x_2)$. We first show how to compile C into a new circuit C' that computes the same function but is more resilient to errors and specific forms of information leakage. We then show how to use a partial corrupted source of orthogonal vectors together with the previous construction applied to C' to get full leakage resilience against an adversary who may both leak information and corrupt some of the pairs received from f_{ORT}.

Securing against Faults and Some Leakage We now compile C into a circuit C'. The compilation takes the security parameter k as input and the size of C' will be polynomial in the size of C and in k.

It is well known that there exists a perfectly secure MPC protocol, computing any function for $3k+1$ players tolerating k actively corrupted players. We might take the protocol from [BOGW88], for instance which is based on Shamir secret sharing. More precisely, we will use a protocol σ_C that takes as input Shamir shares of x_1, x_2 and computes securely Shamir-shares of the output $(y_1, y_2) = C(x_1, x_2)$ for all players, that is, each player gets as input a share of each of the bits in x_1, x_2 and will output a share of each of the bits in y_1 and y_2. The total computational complexity of σ_C is polynomial in k and the size of C. We let C' be the circuit we get by specifying the internal computation of each player as a circuit, and putting wires between players at each point where a message is to be sent. Since in σ_C each player outputs a share of each bit in the result y_1, y_2, C' will output all such shares. We specify all wires carrying a share of y_1 as output wires for player 1 and wires carrying a share of y_2 as output for player 2.

Let N be the number of players in σ_C, and let C'_i denote that part of C' that corresponds to the i'th player in σ_C.

Note that, by security of σ_C, the outputs of corrupt players reveal nothing beyond $C(x_1, x_2)$. In fact, much more is true: even if the parts of C' corresponding to at most k players malfunction and/or leak information, we would still get the correct result and nothing beyond $C(x, y)$ would be revealed.

Getting Full Leakage Resilience We now show how we can compile C' into a fully leakage resilient computation, however, we now consider a more adversarial setting than in the previous section. Namely, we still assume access to f_{ORT}, but we allow the adversary to initially point out a set D of the calls to f_{ORT}. For these calls, he will be allowed to choose the output vectors in any way he wants. Assuming there are m calls in total, the set D can be of size at most ϵm for a constant $0 \leq \epsilon < 1$. In addition, he may leak η bits in each leakage query. Such an adversary is called a (η, ϵ)-adversary.

Our protocol works as follows:

Leakage Resilient Protocol for C, π_{f_C}

1. Invoke f_{ORT} $m = 2|C'|k$ times, where $|C'|$ is the size of C'. The output vectors are stored by A and B.
2. A chooses a random test subset of the pairs of size $m/2$ and sends it to B. The parties exchange the vectors in the test set and check that they are orthogonal. If a non-orthogonal pair is found, the protocol aborts. Otherwise the pairs in the test set are discarded.
3. A chooses a random permutation ξ on $m/2 = k|C'|$ elements and sends it to B. We think of ξ as defining a random division of the remaining pairs into $|C|$ groups of k pairs each. Starting indices from 0, group j consists of pairs number $\xi(jk), \xi(jk+1), ..., \xi(jk+k-1)$.
4. We now start the actual computation: party i secret-shares the bits in his input x_i and uses the shares as input in the protocol we run in the following step.
5. Run the protocol $\pi^n_{f_{C'}}[\pi_{\text{RanBit}}{}^n/f_{\text{RanBit}}{}^n, \pi^n_{\text{ANand}}/f^n_{\text{ANand}}, \pi_{\text{Out}}/f_{\text{Out}}]$, where $f_{C'}$ is the function computed by C', with the following modification: Recall that the protocol runs in the f_{ORT}-hybrid model. When the protocol makes call number j to do a refresh operations (which happens once for every gate) we will instead do k refresh operations sequentially using each pair in group j as input.
6. For each bit in y_i, party i receives $3k+1$ shares from the protocol in the previous step. He considers these shares as a Reed-Solomon codeword with at most k errors, decodes and outputs the result.

Theorem 6. *π_{f_C} is a $\mathsf{negl(k)}$-leakage resilient implementation of the function computed by C, against the $(\lambda/4, \epsilon)$-class of adversaries, where $\mathsf{negl()}$ is a negligible function.*

Proof. Before we start describing the simulator, we note a few technical facts: First, since we test a random subset of size $m/2$ of the m pairs for orthogonality, if there are more than k non-orthogonal pairs, we will continue past the test stage with probability at most 2^{-k}. We may therefore assume in the following that at most k non-orthogonal pairs will be used for the actual computation.

Next, consider the following subset of gates in C':

$$C_{bad} = \{C'_i | \text{ the group of pairs used for a gate in } C'_i \text{ contains a non-orthogonal pair}\}.$$

We can think of C_{bad} as sub-circuit of C', and we will let C_{good} be the rest of C'. Note that C_{bad} contains at most k parts of C', and we may think of the corresponding k players in σ_C as having been actively corrupted by an adversary.

Finally, consider the random groups we form in step 3. A group is said to be bad if it consists only of pairs coming from corrupted calls to f_{ORT}. Clearly, a single group is bad with probability at most ϵ^k, so by the union bound the probability that some group is bad is negligible, namely at most $|C'|\epsilon^k$. So we will assume below that all groups are good.

A simulator S for π_{f_C} is now built as follows: it will first execute the test phase, where it emulates the actions of A and B according to the protocol. For corrupted called to f_{ORT}, it lets the adversary specify the vectors to be used, for the other calls, it generates random orthogonal pairs of its own.

If the protocol aborts, there is nothing further to do. Otherwise, S will start running the simulator S_{σ_C} for the protocol σ_C, telling it that the players corresponding to C_{bad} are corrupted. It chooses a random set of $2k$ Shamir shares for each bit in x_1, x_2 to represent the inputs of corrupt players and gives these as input for C_{bad} to the adversary and to S_{σ_C}. We refer to these later as the *fake shares for the input stage*. S also creates a set of $2k$ random shares for each bit in the output y_1, y_2 to represent the output of corrupt players, these are given to S_{σ_C}.

S will now execute itself the entire computation done for the gates in C_{bad}, starting from leakage resilient encodings of the fake shares for the input stage. When a wire going into C_{good} gets a leakage resilient encoding assigned to it, S will decode it and give the result to S_{σ_C} (note that this is part of a message from a corrupt to an honest player in σ_C). When S needs an encoding for a wire coming out of C_{good}, S lets S_{σ_C} produce the corresponding message and makes a random leakage resilient encoding of it. This way, S will a have full view for both parties of the computation for gates in C_{bad} and will use this view to answer leakage queries addressing this part of the computation.

In parallel, S also runs a simulator S' for $\pi^n_{f_{C_{good}}}[\pi_{\text{RANBIT}}{}^n/f_{\text{RANBIT}}{}^n, \pi^n_{\text{ANAND}}/f^n_{\text{ANAND}}, \pi_{\text{OUT}}/f_{\text{OUT}}]$, in other words, S' will simulate the leakage coming from the computation done by C_{good}.

To see that such a simulator S' exists, note that the groups of pairs we use for each gate in C_{good} contain only orthogonal pairs and each group has at least one vector coming from an uncorrupted call to f_{ORT}. Now, Theorem 3 can still be shown when such groups of pairs are used for refreshing encodings, instead of a single pair: the pairs that are known to the adversary do not hurt the correctness since they are orthogonal, but as soon as one good pair has been used, the state is refreshed as usual. Note also that we need the variant of Theorem 3 mentioned earlier where the protocol is invoked several times, with encoded state passed between invocations, this is because C_{good} may exchange data with C_{bad} several times during the computation. Finally, note that we touch each pair twice, namely when we get it from f_{ORT}, and when it is used for a refresh, but since we limit the leakage to $\lambda/4$ bits per query, the total leakage from each pair is not more than we can tolerate.

Each time the adversary issue a leakage query to a part of the computation inside C_{good}, we let S' handle those queries. If a query asks for leakage from parts both in C_{bad} and in C_{good}, S will hardwire the complete view of the relevant parts of C_{bad} into a leakage function of its own and submit this to S'. This function is designed to compute the adversary's leakage function on the view from C_{bad} as well as the relevant part of the view from C_{good}.

The only leakage queries we have to take special care of are those that leak from the input stage or the output stage. For a leakage function leaking from the input stage of party i, S hardwires the fake shares for the input stage into its own leakage function. Once this function is submitted by S and has access to the input x_i, it interpolates the set of of Shamir shares consistent with x_i and the fake shares. This gives a perfect simulation of party i's view of the input stage and the adversary's leakage function is evaluated on this view. For a leakage function leaking from the outputs stage of party i, note that S_{σ_C} has simulated some messages from honest players to corrupt in the output stage of σ_C. These messages determine the k shares of (each bit in) y_i that corrupt players receive. S hardwires these shares into its own leakage function. Once this function is submitted by S and has access to the output y_i, it interpolates the set of of Shamir shares consistent with y_i and the shares of corrupt players. This gives a perfect simulation of party i's view of the output stage and the adversary's leakage function is evaluated on this view.

We can argue that S is good simulator as follows: Consider a real execution of the protocol, and note that only uncorrupted calls to f_{ORT} are made when computing C_{good}. This implies that 1) the leakage resilient encodings on wires coming out of C_{good} reveal nothing beyond the actual values on these wires (since the internal refreshing of encodings in C_{good} will work) and 2) the computation done in C_{good} correctly emulates the honest player's behaviour in σ_C. Therefore, leakage from C_{bad} effectively gives the adversary access to the interface between honest and corrupt players. This interface is simulated perfectly by S_{σ_C} by security of σ_C (and since S_{σ_C} is given input and output for corrupt players with the correct distribution). Hence the view held by S of the computation in C_{bad} is a perfect simulation of the real view, and so its responses to leakage queries from this part have the correct distribution. Finally, again because only good groups of pairs of vectors are used when computing C_{good}, S' simulates leakage from C_{good} with a statistically close distribution.

4 A Quantum Solution for the Orthogonal Vector Problem

As a warm-up for our positive results, we argue that a solution to the orthogonal vector problem exists using quantum communication: First note that the quantum key distribution protocol by Ekert [Eke91] with security proof by Lo and Chau [LC99] works by first generating a set of essentially perfect EPR pairs that are shared between A and B. In their protocol A and B would now simply measure their particles to generate the shared key. However, given a set of EPR pairs, one may also convert this to a different bipartite state. In particular, in [NV01], Nielsen and Vidal show that if a state $|\phi\rangle_{\mathsf{AB}}$ is *majorized* by another state $|\psi\rangle_{\mathsf{AB}}$, then $|\phi\rangle_{\mathsf{AB}}$ can be converted into $|\psi\rangle_{\mathsf{AB}}$ using only local quantum operations and classical communication. The details of this can be found in [NV01] and are not important here, except that it is easy to see that n copies of the EPR state is majorized by any other bipartite state of the same size. Therefore, we can get a solution to the orthogonal vector problem by converting the EPR pairs to the state

$$|\psi\rangle_{\mathsf{AB}} := \frac{1}{\sqrt{K}} \sum_{\boldsymbol{u},\boldsymbol{v}\in\mathbb{F}^n} |\boldsymbol{u}\rangle_{\mathsf{A}} \otimes |\boldsymbol{v}\rangle_{\mathsf{B}} \qquad \big(\text{where } K = (|\mathbb{F}|^n - 1)|\mathbb{F}|^{n-1} + |\mathbb{F}|^n\big)\,, \tag{1}$$

where A and B are Hilbert spaces of dimension $|\mathbb{F}|^n$. This state contains a superposition of all pairs of orthogonal vectors, so measuring on both sides produces the result we want. Since the QKD part ensures that the adversary is (essentially) unentangled with A and B and this cannot change when only classical communication is done, the adversary has no information on the measurement results, beyond the fact that a pair of orthogonal vectors is produced.

5 A simple quantum protocol

While the protocol from the previous section will work, it would require A and B to perform rather complex quantum operations: already the entanglement-based quantum key distribution part (to generate the EPR pairs) requires full-scale quantum error correction and storage. It would therefore be desirable to come up with a protocol that limits the amount of quantum processing required. Ideally, one would want a prepare-and-measure protocol, in the spirit of quantum key distribution, which would only require A and B to prepare individual qubits, send them to the other side, and then measure the received qubits immediately upon reception. While we will not quite achieve this level of simplicity, we can get much closer to this ideal than the protocol from the last section.

5.1 Basic protocol

Like the EPR-based protocol outlined above, this protocol works by trying to get A and B to share multiple copies of the state $|\psi\rangle_{\mathsf{AB}}$ (see Equation 1). In this protocol, however, instead of starting with EPR pairs, A and B receive their respective shares of this state directly from an untrusted source. After A and B get their possibly corrupted copies of $|\psi\rangle$, they sample some of them using a checking procedure that will ensure that the error rate is sufficiently low. Once this is done, A and B measure the remaining states in the computational basis and use the measurement results as their orthogonal vectors. The checking procedure ensures that the vast majority of the vectors they have are indeed orthogonal and secret.

The protocol for distributing and checking copies of $|\psi\rangle$ is described in Fig. 5.

In the protocol, Test 1 ensures that the input is supported on a subspace which contains very few non-orthogonal pairs, and Test 2 ensures that the state contains coherent superpositions of at least half of all possible pairs of vectors. Since only about a fraction $1/|\mathbb{F}|$ of the pairs are orthogonal, the only states that can pass both tests with non-negligible probability are those that contain the state $|\psi\rangle$ in most positions. In the next section, we analyze the sampling procedure being used here, and then apply the results to our specific protocol in Section 5.3.

Generate-Pairs:
1. A and B receive a state $\rho_{\mathsf{A}^{\otimes m+2\ell}\mathsf{B}^{\otimes m+2\ell}}$ from an untrusted adversary.
2. A and B choose two disjoint random subsets $T_1, T_2 \subseteq \{1, \ldots, m+2\ell\}$, each of size ℓ, to test.
3. (Test 1) For all states in T_1, A and B measure them in the computation basis and reject if any of the results are not orthogonal vectors.
4. (Test 2) For all states in T_2, A and B measure them in the Hadamard basis, and accept only if at least $\frac{\ell K}{|\mathbb{F}|^{2n}}(1-\delta)$ measurement results are $|0^{tn}\rangle|0^{tn}\rangle$.
5. If both tests succeed, then output the remaining m states.

Fig. 5. A simpler quantum protocol.

5.2 A quantum sampling theorem

To analyze the protocol, we will first rephrase it in more general language as a type of sampling procedure that may be of independent interest. We describe the procedure in Fig.6. Like our protocol, this procedure takes an arbitrary state as input, and the goal is to accept if it contains a large fraction of some pure state $|\varphi\rangle$. To ensure this, we again perform two tests: the first test ensures that the vast majority of the positions are in the support of some projector M, and the second test ensures that if we measure the POVM $\{|\theta\rangle\langle\theta|, \mathrm{id} - |\theta\rangle\langle\theta|\}$ for some pure state $|\theta\rangle$ such that $M|\theta\rangle = \sqrt{\gamma}|\varphi\rangle$, then roughly a fraction γ of the sample comes out as $|\theta\rangle$. Our goal will be to show that if the input has a non-negligible probability of passing both tests, then the output will with high probability contain at least $m(1-\epsilon)$ copies of the state $|\varphi\rangle$, for an appropriate choice of ϵ.

Sampling operation $\mathcal{S}$:
1. Input: An unknown state ρ on $\mathsf{H}^{\otimes m+2\ell}$.
2. Choose two disjoint random subsets $T_1, T_2 \subseteq \{1, \ldots, m+2\ell\}$, each of size ℓ, to test.
3. (Test 1) For all states in T_1, measure them with the POVM $\{M, \mathrm{id} - M\}$, accept only if the result is M for all of them.
4. (Test 2) For all states in T_2, measure them using POVM $\{|\theta\rangle\langle\theta|, \mathrm{id} - |\theta\rangle\langle\theta|\}$, and accept only if at least $\gamma\ell(1-\delta)$ measurement results are $|\theta\rangle\langle\theta|$, where γ is such that $M|\theta\rangle = \sqrt{\gamma}|\varphi\rangle$.
5. If both tests succeed, then output the remaining m states.

Fig. 6. The sampling operation analyzed in Section 5.2

To formalize this statement, we will view the whole sampling procedure as a completely positive, trace-preserving map $\mathcal{S}$ from the Hilbert space $\mathsf{H}^{\otimes m+2\ell}$ to $\mathsf{H}^{\otimes m} \oplus \mathrm{span}\{|\mathrm{fail}\rangle\}$, the $|\mathrm{fail}\rangle$ flag being used to denote the case when the sampling procedure aborts due to a failed test. We also define the projector $\Pi_{\epsilon,\mathrm{good}}$, which projects onto the subspace of $\mathsf{H}^{\otimes m}$ spanned by vectors containing at least $m(1-\epsilon)$ copies of $|\varphi\rangle$, along with $|\mathrm{fail}\rangle$:

$$\Pi_{\epsilon,\mathrm{good}} := |\mathrm{fail}\rangle\langle\mathrm{fail}| + \sum_{x\in\{0,1\}^m, |x|\leqslant\epsilon m} \Pi_{x_1} \otimes \cdots \otimes \Pi_{x_m},$$

where $\Pi_0 = |\varphi\rangle\langle\varphi|$, $\Pi_1 = \mathrm{id} - \Pi_0$, and $|x|$ denotes the Hamming weight of x. We will now prove the following:

Theorem 7. *Let $\mathcal{S}_{\mathrm{ideal}}$ be the "ideal" version of $\mathcal{S}$, defined as $\mathcal{S}_{\mathrm{ideal}}(\rho) := \Pi_{\epsilon,\mathrm{good}}\mathcal{S}(\rho)\Pi_{\epsilon,\mathrm{good}}$, with $\epsilon = 4\delta + 8\gamma^{-1}\sqrt{\delta}$. Then, we have that*

$$\|\mathcal{S} - \mathcal{S}_{\mathrm{ideal}}\|_{\diamond} \leqslant \exp(-\Omega(\min(\ell, m))). \tag{2}$$

Furthermore, if we run on the protocol on the honest input $|\varphi\rangle^{\otimes m+2\ell}$, then the probability of outputting $|\mathrm{fail}\rangle$ is at most $\ell\exp(-\ell\gamma^2\delta^2)$.

Note that the diamond norm $\|\cdot\|_\diamond$ can be interpreted as follows: if we are given a box that implements either $\mathcal{E}$ or $\mathcal{F}$ and we must distinguish which of the two using the box only once, the probability that we will guess correctly using the best strategy is given by $\frac{1}{2}+\frac{1}{4}\|\mathcal{E}-\mathcal{F}\|_\diamond$. Hence, the above theorem means that one cannot distinguish the real sampling protocol from the "ideal" protocol that never outputs something outside of the support of $\Pi_{\epsilon,\text{good}}$ except with negligible probability. The proof strategy will be as follows: we will first show that it is sufficient to consider inputs of the form $\rho_{\mathsf{H}^{\otimes m+2\ell}} = \sigma_{\mathsf{H}}^{\otimes m+2\ell}$ using a variant of the "postselection" technique from [CKR09], and we then show that the only σ's that survive both tests have a high fidelity with $|\varphi\rangle$.

Proof. First, we can rephrase Equation (2) to the condition that

$$\|(\mathcal{S}\otimes \mathrm{id}_{\mathsf{R}})(\rho) - (\mathcal{S}_{\text{ideal}}\otimes \mathrm{id}_{\mathsf{R}})(\rho)\|_1 \leqslant \exp(-\Omega(\min(\ell,m)))$$

holds for any density operator ρ on $\mathsf{H}^{\otimes m+2\ell}\otimes \mathsf{R}$ where R is some auxiliary space. Now, using the fact that $\mathcal{S}_{\text{ideal}}(\rho) = \Pi_{\epsilon,\text{good}}\mathcal{S}(\rho)\Pi_{\epsilon,\text{good}}$ together with the gentle measurement lemma [Win99, Lemma 9], we get that

$$\begin{aligned}\max_\rho \|(\mathcal{S}\otimes\mathrm{id})(\rho) - \Pi_{\epsilon,\text{good}}(\mathcal{S}\otimes\mathrm{id})(\rho)\Pi_{\epsilon,\text{good}}\|_1 &\leqslant \max_\rho \sqrt{8\,\mathrm{tr}\left[(\Pi^{\perp}_{\epsilon,\text{good}}\otimes \mathrm{id}_{\mathsf{R}})(\mathcal{S}\otimes\mathrm{id}_{\mathsf{R}})(\rho)\right]}\\ &\leqslant \max_\rho \sqrt{8\,\mathrm{tr}\left[\Pi^{\perp}_{\epsilon,\text{good}}\mathcal{S}(\rho)\right]},\end{aligned}$$

where $\Pi^{\perp}_{\epsilon,\text{good}} := \mathrm{id}-\Pi_{\epsilon,\text{good}}$. At this point, we will proceed in two steps: the first step will be to show that it is sufficient to prove that the sampling procedure works for independent, identically-distributed (iid) states, as in [CKR09]. The second step will then be to show that it does indeed work for these states; this will follow almost directly from results about typical sequences.

Loosely speaking, the postselection technique from [CKR09] enables us to get bounds involving permutation-invariant superoperators. The first difficulty we encounter if we try to apply it directly here is that the bound is exponential in the dimension of the Hilbert space H, which may be very large (in fact, in the case of our protocol, it is $|\mathbb{F}|^{2n}$). To get around this problem, we will show that there exists an equivalent procedure which acts on a much smaller space. First, let us define the following:

$$\begin{aligned}\left|\varphi^\perp\right\rangle &:= \frac{(\mathrm{id}-M)|\theta\rangle}{\|(\mathrm{id}-M)|\theta\rangle\|} = \frac{(\mathrm{id}-M)|\theta\rangle}{\sqrt{1-\gamma}}\\ Q_0 &:= |\varphi\rangle\langle\varphi|\\ Q_1 &:= M - |\varphi\rangle\langle\varphi|\\ Q_2 &:= \left|\varphi^\perp\right\rangle\left\langle\varphi^\perp\right|\\ Q_3 &:= \mathrm{id} - M - \left|\varphi^\perp\right\rangle\left\langle\varphi^\perp\right|\end{aligned}$$

Furthermore, let us define the space $\mathsf{K} = \mathrm{span}\{|0\rangle,|1\rangle,|2\rangle,|3\rangle\}$, and the superoperator $\mathcal{M}$ which acts as follows:

$$\mathcal{M}(\rho) := \left(|0\rangle\langle\varphi| + |2\rangle\left\langle\varphi^\perp\right|\right)\rho\left(|\varphi\rangle\langle 0| + \left|\varphi^\perp\right\rangle\langle 2|\right) + |1\rangle\langle 1|\,\mathrm{tr}[\rho Q_1] + |3\rangle\langle 3|\,\mathrm{tr}[\rho Q_3].$$

In other words, $\mathcal{M}$ measures ρ using the POVM $\{Q_0+Q_2, Q_1, Q_3\}$; if the first result comes out, then we rotate $|\varphi\rangle$ and $|\varphi^\perp\rangle$ into $|0\rangle$ and $|2\rangle$ respectively. If Q_1 or Q_3 come out, we output $|1\rangle$ or $|3\rangle$ as the case may be. Note that this operation effectively commutes with the sampling operation: if we first apply $\mathcal{M}^{\otimes m+2\ell}$, there is a CPTP map $\mathcal{E}$ on $\mathsf{K}^{\otimes m+2\ell}$ that will produce the same results as $\mathcal{S}$. Indeed, whenever $\mathcal{S}$ measures a position using the POVM $\{M, \mathrm{id}-M\}$ (corresponding to Test 1), $\mathcal{E}$ will measure the corresponding position using $\{|0\rangle\langle 0|+|1\rangle\langle 1|, |2\rangle\langle 2|+|3\rangle\langle 3|\}$, and whenever $\mathcal{S}$ measures using $\{|\theta\rangle\langle\theta|, \mathrm{id}-|\theta\rangle\langle\theta|\}$ (corresponding to Test 2), $\mathcal{E}$ will measure using $\{|\tilde{0}\rangle\langle\tilde{0}|, \mathrm{id}-|\tilde{0}\rangle\langle\tilde{0}|\}$ where

$|\tilde{0}\rangle = \sqrt{\gamma}|0\rangle + \sqrt{1-\gamma}|2\rangle$; this follows from the fact that $|\theta\rangle = \sqrt{\gamma}|\varphi\rangle + \sqrt{1-\gamma}|\varphi^\perp\rangle$. Furthermore, there exists a projector $\tilde{\Pi}^\perp_{\epsilon,\mathrm{good}}$ that corresponds to $\Pi^\perp_{\epsilon,\mathrm{good}}$. We can then continue the above as:

$$\max_\rho \sqrt{8\,\mathrm{tr}\left[\Pi^\perp_{\epsilon,\mathrm{good}}\mathcal{S}(\rho)\right]} = \max_{\rho\in\mathsf{D}(\mathsf{H}^{\otimes m+2\ell})} \sqrt{8\,\mathrm{tr}\left[\tilde{\Pi}^\perp_{\epsilon,\mathrm{good}}\mathcal{E}(\mathcal{M}(\rho))\right]}$$
$$\leqslant \max_{\rho\in\mathsf{D}(\mathsf{K}^{\otimes m+2\ell})} \sqrt{8\,\mathrm{tr}\left[\tilde{\Pi}^\perp_{\epsilon,\mathrm{good}}\mathcal{E}(\rho)\right]}.$$

Now, let us define $\tau_{\mathsf{K}^{\otimes m+2\ell}}$ as in Lemma 4. This state has the property that, for any permutation-invariant density operator ω, $\omega \leqslant g\tau$ where $g \leqslant (m+2\ell+1)^{15}$. We now use the fact that both the sampling procedure and $\Pi_{\epsilon,\mathrm{good}}$ are permutation invariant: for any permutation $\pi \in \mathcal{S}_{m+2\ell}$ of all $m+2\ell$ subsystems, we have that $\mathrm{tr}[\tilde{\Pi}^\perp_{\epsilon,\mathrm{good}}\mathcal{E}(\pi\rho\pi^\dagger)] = \mathrm{tr}[\tilde{\Pi}^\perp_{\epsilon,\mathrm{good}}\mathcal{E}(\rho)]$. We then continue as follows:

$$\max_{\rho\in\mathsf{D}(\mathsf{K}^{m+2\ell})} \sqrt{8\,\mathrm{tr}\left[\tilde{\Pi}^\perp_{\epsilon,\mathrm{good}}\mathcal{E}(\rho)\right]} = \max_\rho \sqrt{8\,\mathrm{tr}\left[\tilde{\Pi}^\perp_{\epsilon,\mathrm{good}}\mathcal{E}\left(\mathbb{E}_{\pi\in\mathcal{S}_{m+2\ell}}\pi\rho\pi^\dagger\right)\right]}.$$

In the above, $\mathbb{E}_{\pi\in\mathcal{S}_{m+2\ell}}$ denotes the expected value when π is chosen uniformly over the symmetric group on $m+2\ell$ elements. Clearly $\mathbb{E}_\pi\pi\rho\pi^\dagger$ is permutation invariant, and therefore $\mathbb{E}_\pi\pi\rho\pi^\dagger \leqslant g\tau$. Hence:

$$\max_\rho \sqrt{8\,\mathrm{tr}\left[\tilde{\Pi}^\perp_{\epsilon,\mathrm{good}}\mathcal{E}\left(\mathbb{E}_\pi\pi\rho\pi^\dagger\right)\right]} \leqslant \sqrt{8g\,\mathrm{tr}\left[\tilde{\Pi}^\perp_{\epsilon,\mathrm{good}}\mathcal{E}\left(\tau\right)\right]}.$$

Here, we use the fact that τ can be expressed as $\tau = \int \sigma^{\otimes m+2\ell}\mu(\sigma)$ for some measure μ over normalized density operators on K (see Lemma 4) to bound this as follows:

$$\sqrt{8g\,\mathrm{tr}\left[\tilde{\Pi}^\perp_{\epsilon,\mathrm{good}}\mathcal{E}\left(\tau\right)\right]} \leqslant \max_{\sigma\in\mathsf{D}(\mathsf{K})} \sqrt{8g\,\mathrm{tr}\left[\tilde{\Pi}^\perp_{\epsilon,\mathrm{good}}\mathcal{E}\left(\sigma^{\otimes m+2\ell}\right)\right]}$$
$$= \max_{\sigma\in\mathsf{D}(\mathsf{H})} \sqrt{8g\,\mathrm{tr}\left[\Pi^\perp_{\epsilon,\mathrm{good}}\mathcal{S}\left(\sigma^{\otimes m+2\ell}\right)\right]}.$$

We have now reduced the problem to showing that the sampling procedure works on independent, identically-distributed inputs.

Now, to finish the proof, note that the trace term above corresponds to the probability that the sampling procedure does not output $|\mathrm{fail}\rangle$, and yet the output falls outside of the support of $\Pi_{\epsilon,\mathrm{good}}$ when applied to an iid state $\sigma^{\otimes m+2\ell}$. At this point, we consider two cases. First, assume $\langle\varphi|\sigma|\varphi\rangle \geqslant 1-\frac{\epsilon}{2}$. Then, by Lemma 6, $\mathrm{tr}\left[\Pi^\perp_{\epsilon,\mathrm{good}}\sigma^{\otimes m}\right] \leqslant m\exp(-m\epsilon^2/2)$, so the bound holds regardless of whether the tests are passed or not. Now, suppose $\langle\varphi|\sigma|\varphi\rangle < 1-\frac{\epsilon}{2}$. Then, we show that, with high probability, either Test 1 or Test 2 will fail. Using the fact that $\epsilon = 4\delta + 2\gamma^{-1}\sqrt{16\delta}$, we get that by Lemma 5, we have that either $\mathrm{tr}[M\sigma] < 1-2\delta$ or $\langle\theta|\sigma|\theta\rangle < \gamma(1-2\delta)$. Now, if $\mathrm{tr}[M\sigma] < 1-2\delta$, then the probability that Test 1 will pass is at most $(1-2\delta)^\ell$, and if $\langle\theta|\sigma|\theta\rangle < \gamma(1-2\delta)$, then the probability that Test 2 will pass is at most $\ell\exp\{-2\ell\gamma^2\delta^2\}$ using Lemma 6. This concludes the proof of the first part of the statement.

Finally, to prove the last part of the statement, we need to show that the protocol succeeds with high probability on the honest input $|\varphi\rangle^{\otimes m+2\ell}$. In this case, it is clear that the output is correct whenever both tests are passed, and furthermore that Test 1 always succeeds. The only remaining possibility is that Test 2 fails, and according to Lemma 6 once again, we see that the probability of passing Test 2 is at least $1-\ell\exp(-2\ell\gamma^2\delta^2)$. □

Lemma 4. *Let $\rho_{\mathsf{H}^{\otimes k}}$ be a normalized density operator on some Hilbert space $\mathsf{H}^{\otimes k}$ invariant under permutations of the k subsystems. Then, $\rho \leqslant g\tau$, for some $\tau = \int \sigma^{\otimes k}\mu(\sigma)$ where μ is some probability measure over normalized density operators on H, and $g = \binom{k+d^2-1}{k} \leqslant (k+1)^{d^2-1}$, with $d = \dim \mathsf{H}$.*

Proof. Since ρ is permutation-invariant, according to [Ren05, CKMR07] we can find a purification $|\theta\rangle_{\mathsf{H}^{\otimes k}\mathsf{K}^{\otimes k}}$, with $\mathsf{K} \cong \mathsf{H}$ and such that $|\theta\rangle$ is in the symmetric subspace $\mathrm{Sym}^k(\mathsf{H} \otimes \mathsf{K})$ of $\mathsf{H}^{\otimes k} \otimes \mathsf{K}^{\otimes k}$. Now, let P be the projector onto $\mathrm{Sym}^k(\mathsf{H} \otimes \mathsf{K})$ and $\tau_{\mathsf{H}^{\otimes k}\mathsf{K}^{\otimes k}} = \frac{1}{g}P$ (note that $\dim \mathrm{Sym}^k(\mathsf{H} \otimes \mathsf{K}) = g$). Then, it is clear that

$$|\theta\rangle\langle\theta|_{\mathsf{H}^{\otimes k}\mathsf{K}^{\otimes k}} \leqslant P = g\tau_{\mathsf{H}^{\otimes k}\mathsf{K}^{\otimes k}}. \tag{3}$$

Now, it is an elementary fact about the symmetric subspace that $\tau = \int_U U^{\otimes k}|\psi\rangle\langle\psi|^{\otimes k}U^{\dagger\otimes k}$ for any $|\psi\rangle \in \mathsf{H} \otimes \mathsf{K}$, where the integral is taken over the Haar measure on the unitary group on $\mathsf{H} \otimes \mathsf{K}$. Then, tracing out $\mathsf{K}^{\otimes k}$ on both sides of (3) yields the lemma. □

Lemma 5. *Let σ_{H} be a normalized density operator on H. Then, if $\mathrm{tr}[M\sigma] \geqslant 1 - \epsilon_1$ and $\mathrm{tr}[|\theta\rangle\langle\theta|\sigma] \geqslant \gamma(1-\epsilon_2)$, then $\langle\varphi|\sigma|\varphi\rangle \geqslant 1 - \epsilon_2 - \gamma^{-1}\sqrt{8\epsilon_1}$.*

Proof. First, note that by the gentle measurement lemma,

$$\|M\sigma M - \sigma\|_1 \leqslant \sqrt{8\epsilon_1}.$$

We can then calculate the fidelity between $|\varphi\rangle$ and σ:

$$\begin{aligned}
\langle\varphi|\sigma|\varphi\rangle &= \mathrm{tr}[|\varphi\rangle\langle\varphi|\sigma] \\
&= \gamma^{-1}\,\mathrm{tr}[M|\theta\rangle\langle\theta|M\sigma] \\
&= \gamma^{-1}\left(\mathrm{tr}[|\theta\rangle\langle\theta|\sigma] - \mathrm{tr}[|\theta\rangle\langle\theta|(\sigma - M\sigma M)\right) \\
&\geqslant \gamma^{-1}\left(\mathrm{tr}[|\theta\rangle\langle\theta|\sigma] - \|\sigma - M\sigma M\|_1\right) \\
&\geqslant \gamma^{-1}\left(\gamma(1-\epsilon_2) - \sqrt{8\epsilon_1}\right) \\
&= 1 - \epsilon_2 - \gamma^{-1}\sqrt{8\epsilon_1}.
\end{aligned}$$

□

Lemma 6. *Let ρ be a density matrix on a Hilbert space K, let $\{Q_0, Q_1\}$ be a two-outcome POVM on K such that $\mathrm{tr}[Q_1\rho] = q$, and let $\mathcal{B}(p)$ be the set of n-bit strings with at least pn 1's. Now, let $Q = \sum_{x\in\mathcal{B}(p)} Q_{x_1} \otimes \ldots \otimes Q_{x_n}$. Then, if $p > q$, we have that*

$$\mathrm{tr}[Q\rho^{\otimes n}] \leqslant n\exp(-nD(p\|q)) \leqslant n\exp(-2n(p-q)^2),$$

and if $p \leqslant q$,

$$\mathrm{tr}[Q\rho^{\otimes n}] \geqslant 1 - n\exp(-nD(p\|q)) \geqslant 1 - n\exp(-2n(p-q)^2).$$

where $D(p\|q) := p\log\frac{p}{q} + (1-p)\log\frac{1-p}{1-q}$ is the relative entropy between two binary distributions with parameters p and q.

Proof. This is a consequence of well-known facts regarding classical typical sequences. Indeed, $\mathrm{tr}[Q\rho^{\otimes n}]$ can be rewritten as

$$\sum_{x\in\mathcal{B}_\epsilon(\lambda)} q^{|x|}(1-q)^{n-|x|}$$

which can be upper-bounded by $n\exp(-nD(p\|q))$ as in [Csi98, Equation (II.5)]. We then use the fact that $D(p\|q) \geqslant 2(p-q)^2$ to get the simpler bound. To get the lower bound in the second case, we apply the same inequality, but with $\mathrm{id} - Q$ instead. □

5.3 Security of the protocol

We will apply Theorem 7 to prove the security of the protocol given in Fig.5. Before we proceed, we will need to introduce some notation. In the following, let

$$P := \sum_{\boldsymbol{u}\cdot\boldsymbol{v}=0} |\boldsymbol{u}\rangle|\boldsymbol{v}\rangle\langle\boldsymbol{u}|\langle\boldsymbol{v}|$$

be the projector onto the set of all orthogonal vectors, and let

$$|0_H\rangle_{\mathsf{AB}} := H^{\otimes 2n}|0^n\rangle_{\mathsf{A}}|0^n\rangle_{\mathsf{B}}$$

be the coherent superposition of all possible pairs. Now, we can easily see that P and $|0_H\rangle$ play the roles of M and $|\theta\rangle$ respectively in Theorem 7. All that is left to check is that $P|0_H\rangle = \sqrt{\gamma}|\psi\rangle$ for an appropriate choice of γ. We do this in the following lemma:

Lemma 7. *We have that $P|0_H\rangle = \sqrt{c}|\psi\rangle$, where $c = \frac{1}{|\mathbb{F}|} + \frac{1}{|\mathbb{F}|^n}\left(1 - \frac{1}{|\mathbb{F}|}\right)$.*

Proof. Recall that $K = (|\mathbb{F}|^n - 1)|\mathbb{F}|^{n-1} + |\mathbb{F}|^n$ is the number of pairs of orthogonal vectors. We have that

$$\begin{aligned}
P|0_H\rangle &= \left(\sum_{\boldsymbol{u}\cdot\boldsymbol{v}=0} |\boldsymbol{u}\rangle|\boldsymbol{v}\rangle\langle\boldsymbol{u}|\langle\boldsymbol{v}|\right)\left(\frac{1}{|\mathbb{F}|^n}\sum_{\boldsymbol{w},\boldsymbol{x}}|\boldsymbol{w}\rangle|\boldsymbol{x}\rangle\right)\\
&= \frac{1}{|\mathbb{F}|^n}\sum_{\boldsymbol{w}\cdot\boldsymbol{x}=0}|\boldsymbol{w}\rangle|\boldsymbol{x}\rangle\\
&= \sqrt{\frac{K}{|\mathbb{F}|^{2n}}}\left(\sum_{\boldsymbol{w}\cdot\boldsymbol{x}=0}\frac{1}{\sqrt{K}}|\boldsymbol{w}\rangle|\boldsymbol{x}\rangle\right)\\
&= \sqrt{c}|\psi\rangle,
\end{aligned}$$

where we have used the fact that $c = \frac{K}{|\mathbb{F}|^{2n}}$. □

Now, we can directly apply Theorem 7 to the protocol at hand. The condition involving the diamond norm directly implies security once it is embedded in a larger protocol: it shows that no procedure can distinguish between the actual protocol and an ideal protocol that cannot output a state outside the support of $\Pi_{\epsilon,\mathrm{good}}$. Since we could consider the larger protocol in which the sampling procedure is embedded as an attempt to distinguish the two, we can simply assume that the larger protocol is actually calling the ideal sampling procedure rather than the real one.

6 Leakage Resilience against a Quantum Adversary.

6.1 Security for Bounded-Time Quantum Memory.

It might seem natural to expect that we could immediately get leakage resilience for a quantum adversary by using the protocol from the previous section as an implementation of a source of orthogonal pairs, and then run our classical protocol π_{f_C} from Theorem 6, which was designed to tolerate a constant fraction of corrupted pairs. Indeed, the parameters of the quantum protocol can be chosen such that the state created is very close to an ideal state where at most a constant fraction of the pairs come from measuring an incorrect state.

However, the problem is that the adversary may be in superposition of having corrupted several different sets of pairs, and therefore we cannot assume that one single set of pairs is bad, as we did in the proof of Theorem 6[6].

[6] Of course, each pair we generate is either orthogonal or not, but we cannot point out a single set of pairs that are known to the adversary.

On the other hand, if the adversary cannot keep a coherent state alive until the computation is done, his superposition will collapse, and then Theorem 6 can indeed be applied. The only price we pay is an added term in the statistical distance between simulation and real execution that comes from the distance between real and ideal state in the quantum protocol.

Since long-term quantum memory is much beyond current technology, and we don't assume quantum memory for honest parties, limiting the adversary's memory in this way can be a very reasonable assumption.

Nevertheless, we can get results for an adversary with unlimited quantum memory, as we will now show.

6.2 Leakage Resilience for Unlimited Quantum Memory.

For this case, we will only be able to get a circuit compilation type of result, rather than a 2-party computation protocol.

We recall informally the notion of leakage resilient circuit compilation: here a Boolean circuit C is given that takes a secret input y and a public input chosen by the adversary. The output $C(x, y)$ is given to the adversary. The goal is now to compile C into a leakage resilient computation that is split in several parts where the adversary is allowed to leak independently from different parts, as well as a small amount of global leakage, as defined in our model. The secret input y is assumed to be given in some specially encoded form. A simulator must exist that simulates the adversary's view given only $x, C(x, y)$, i.e., a public-input-output simulator as described earlier. This is a special case of our model of two-party computation, namely we will give x as input to both parties and both parties get $C(x, y)$ as output. Furthermore, we assume that the protocol is given access to an oracle function $g_y()$ that takes no input, and outputs y to the parties encoded in some form that is suitable for the protocol. In an implementation there would be a single device executing all parts of the computation, but here we will stick to the interpretation as a two-player protocol for consistency with the rest of the paper.

We now define an oracle function $g_y()$, which will fit into the way our protocol π_{f_C} represents data. It will make $3k + 1$ Shamir secret shares of each bit in y and output leakage resilient encodings of these shares to the two parties.

We can then construct a protocol called $\pi_{f_C,y}$ which we define to be our protocol π_{f_C} (which will compute $C(x, y)$), except that instead of getting an encoding of shares of y from the parties, it calls $g_y()$ initially and otherwise evaluates C as usual, assuming access to f_{ORT}. While $\pi_{f_C,y}$ is only secure against a classical adversary, we can still use it to construct a protocol that is secure against a quantum adversary.

The idea is as follows: we build from C a new circuit $\bar{C}$ taking inputs $x, \bar{y}$, which works as follows: It parses $\bar{y}$ as the concatenation of a bit b and a string y. If $b = 0$, it outputs $C(x, y)$, while if $b = 1$ it outputs y. One may think of $\bar{C}$ as being like C but with a built-in trapdoor: it "usually" works like C, but if you can choose the secret input, you can force the output value. The idea is now to simply run $\pi^k_{f_{\bar{C}},\bar{y}}$ instead of $\pi_{f_C,y}$. That is, the same protocol is run, but we replace C by $\bar{C}$. This also means that the protocol will call initially an oracle function $g_{\bar{y}}()$ that will produce leakage resilient encodings of Shamir shares in $\bar{y}$. Finally, note that the protocol will construct a new circuit $\bar{C}'$ from $\bar{C}$, just as we built C' from C earlier, based on an MPC protocol secure against k corrupted players.

Our protocol will be used only with secret input of form $\bar{y} = (0, y)$. The other option is only something we need for the proof. Our leakage resilient protocol now works as follows:

Quantum Leakage Resilient Protocol for C, $Q_{f_C,\bar{y}}$

1. Invoke the quantum protocol to obtain $m = 2|\bar{C}'|k$ pairs of (hopefully) orthogonal vectors. For each pair, the process of measuring and storing the pair is defined to be a separate part of the computation from which the adversary can leak a bounded amount of information.
2. Run $\pi_{f_{\bar{C}},\bar{y}}$ based on the list of pairs of vectors from the quantum protocol (instead of getting the list from f_{ORT}). Output whatever $\pi_{f_{\bar{C}},\bar{y}}$ outputs.

We will consider an adversary that is quantum and unbounded, and for simplicity we assume he is *non-adaptive*, that is, he specifies the collection F of leakage functions he wants to use before the actual computation starts. We sketch later how we may get rid of this restriction. If furthermore all these leakage functions output at most $\lambda/4$ bits, we call this quantum, unbounded and non-adaptive $\lambda/4$-adversary.

Definition 3. *Consider a protocol π for computing circuit C on public input x chosen by adversary E and secret input y. The adversary may issue a leakage query for every separate part of the computation. Let Φ_E^{real} be the state of the adversary after executing π. We say that π is leakage resilient against an adversary E if there exists a simulator S that interacts with E (where, after E sends the public input x, S is given $C(x, y)$). This interaction results in state Φ_E^{sim} for E. We require that the trace norm distance between Φ_E^{real} and Φ_E^{sim} be negligible in the security parameter.*

Theorem 8. *$Q_{f_C,\bar{y}}$ is leakage resilient against a quantum, unbounded and non-adaptive $\lambda/4$-adversary.*

Proof. The simulator S we need is very simple to describe: on input $x, C(x, y)$, S will choose a secret input $\bar{y} = (1, C(x, y))$ and will now execute $Q_{f_C,\bar{y}}$ with the adversary, exactly as described above. Note that since S executes the entire computation itself, it knows all the data involved and can hence trivially answer all leakage queries, and furthermore, by construction of $\bar{C}'$, the output will be $C(x, y)$ as it should be.

To show that S satisfies the requirement we first observe that since the simulator plays the real protocol, only with a different secret input, it is enough to show that for any fixed choice of two secret inputs $\bar{y}, \bar{y}'$ to $\bar{C}'$, public input x with $\bar{C}'(x, \bar{y}) = C(x, \bar{y}')$, and allowed leakage functions F, executing $Q_{f_C,\bar{y}}$, respectively $Q_{f_C,\bar{y}'}$ will leave the quantum adversary in states that are at negligible trace distance from each other.

We first need some notation and terminology. Recall that after the quantum protocol A and B hold m states, that we hope are all copies of the correct state for a pair of vectors. Let D be a set of at most k indices among the set $\{1, 2, \dots, m\}$, and let $\mathcal{D}_k$ be the collection of all such sets. The state held by A and B at the end of the quantum protocol is said to be D-corrupt if it consists of copies of the correct state on all positions outside D, while some arbitrary (possibly mixed) state is in the positions inside D. Clearly, by measuring a D-corrupt state, A and B would obtain pairs of vectors that might also have been generated by an adversary corrupting k calls to f_{ORT} as we considered earlier.

We will now consider the state of the adversary E and A, B after the quantum protocol is done and the adversary has specified x, F, also let the secret input be $\bar{y}$. We keep these choices fixed in the following. From the analysis of the quantum protocol, we know that this state is negligibly close to an ideal state that is a superposition of only cases where A and B hold a D-corrupt state ρ_D, for some $D \in \mathcal{D}_k$. So we will assume in the following that we have exactly such an ideal state. For fixed $x, F, \bar{y}, D, \rho_D$, we let

$$P = \{p_D(v) | v \in \mathcal{V}\}$$

denote the probability distribution of the resulting view given to the adversary. The distribution is taken over the distribution coming from measuring ρ_D, and over the random coins used internally by A and B.

Observe that each pair of vectors we use is touched twice in the protocol, namely when the pair is measured and when it is used later. Since we consider a $\lambda/4$-adversary, at most $\lambda/2$ bits are leaked from each pair and so we can use the proof of Theorem 6 to conclude that there exists a classical simulator that will simulate the adversary's view given x, F and assuming that the adversary has corrupted a subset D of the pairs[7]. So for the fixed x, F, D, ρ_D, we let

$$Q = \{q_D(v) | v \in \mathcal{V}\}$$

[7] The theorem talks about a classical adversary, but that since we fixed D and the view consists of a single classical message that is sent to the adversary, it does not matter here that the adversary is now quantum.

denote the probability distribution we get from running the classical simulator on input x, F and the result of measuring ρ_D. By Theorem 6 we may assume that the statistical distance from P to Q is at most a negligible function ϵ of the security parameter.

Recalling that the state after the quantum protocol contains only D-corrupt sets, we will assume that the adversary E has a register E_1 that explicitly stores D, this only gives him more information. Then $|\Phi_D\rangle_{\mathsf{E}_2}$ denotes the normalised state of his entire additional memory in the branch of the superposition where D is the corrupted set of pairs. The state we obtain after running the real game can now be written as follows

$$|\Psi_{real}(\bar{y})\rangle = \sum_{D \in \mathcal{D}_k} \alpha_D |D\rangle_{\mathsf{E}_1} |\Phi_D\rangle_{\mathsf{E}_2} \sum_{v \in \mathcal{V}} \sqrt{p_D(v)} |v\rangle_{\mathsf{E}_3} |v\rangle_{\mathsf{AB}},$$

where we emphasise that this state depends on the secret input $\bar{y}$. We have purified the random choice of v by A and B, such that by tracing out the register AB we obtain exactly the state the adversary has at the end. The α_D's form a vector of length 1. We can now construct a (slightly) different state that does not depend on the secret input:

$$|\Psi_{sim}\rangle = \sum_{D \in \mathcal{D}_k} \alpha_D |D\rangle_{\mathsf{E}_1} |\Phi_D\rangle_{\mathsf{E}_2} \sum_{v \in \mathcal{V}} \sqrt{q_D(v)} |v\rangle_{\mathsf{E}_3} |v\rangle_{\mathsf{AB}} .$$

Note that we do not have to generate $|\Psi_{sim}\rangle$ efficiently, we just need that it exists.

Lemma 8. *The trace-norm distance between $|\Psi_{sim}\rangle$ and $|\Psi_{real}(\bar{y})\rangle$ is negligible.*

Proof. Since $|D\rangle, |D'\rangle$ are orthogonal for distinct D, D' and similarly for the $|v\rangle$'s, we easily obtain the fidelity for $|\Psi_{sim}\rangle$ and $|\Psi_{real}(\bar{y})\rangle$:

$$\langle \Psi_{real}(\bar{y}) | \Psi_{sim} \rangle = \sum_{D \in \mathcal{D}_k} |\alpha_D|^2 \sum_{v \in \mathcal{V}} \sqrt{p_D(v) q_D(v)} \geq 1 - \epsilon ,$$

where in the last step we used the standard inequality linking fidelity and trace-norm distance (in a classical version) to conclude that $\sum_{v \in \mathcal{V}} \sqrt{p_D(v) q_D(v)} \geq 1 - \epsilon$. Another standard inequality now tells us that if 1 minus the fidelity is negligible, then the trace norm distance is also negligible. □

Now note that the state the adversary has at the end of the game is obtained by tracing out the AB register. This cannot increase the trace distance as it corresponds to the adversary trying to distinguish by looking at only a part of the state. This means that at the end of the game, for any secret $\bar{y}$, the adversary's state is negligibly close to one fixed state that does not depend on $\bar{y}$. It trivially follows, as desired, that the adversary cannot distinguish between different secret inputs. □

Adaptive choice of leakage functions. In the adaptive model, the adversary specifies a single leakage function, gets the output, specifies the next, and so on. To handle this situation, we first purify all the choices of functions by E and responses from A, B, similarly to what we did for the single response in the above proof: each choice will be determined by an entangled state shared between E and A, B, with a superposition branch for each possible value. Each party can still evaluate its response by working in superposition. At the end we obtain a joint state from which we can obtain the adversary's end state by tracing out some registers and measuring those that correspond E's classical choices. Now we construct a state that does not depend on $\bar{y}$ by replacing the algorithm run by A and B by that of the simulator. This only changes the state by a negligible amount again by the result on classical simulation.

Bibliography

[BCH12] Nir Bitansky, Ran Canetti, and Shai Halevi. Leakage-tolerant interactive protocols. In *Theory of Cryptography*, pages 266–284. Springer, 2012.

[BGJK12] Elette Boyle, Shafi Goldwasser, Abhishek Jain, and Yael Tauman Kalai. Multiparty computation secure against continual memory leakage. In *Proceedings of the 44th symposium on Theory of Computing*, pages 1235–1254. ACM, 2012.

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. pages 1–10, 1988.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.

[CKMR07] Matthias Christandl, Robert König, Graeme Mitchison, and Renato Renner. One-and-a-half quantum de Finetti theorems. *Communications in Mathematical Physics*, 273(2):473–498, 2007.

[CKR09] Matthias Christandl, Robert König, and Renato Renner. Postselection technique for quantum channels with applications to quantum cryptography. *Physical Review Letters*, 102:020504, Jan 2009.

[Csi98] Imre Csiszár. The method of types. *IEEE Transactions on Information Theory*, 44(6):2505–2523, oct 1998.

[CV12] Ran Canetti and Margarita Vald. Universally composable security with local adversaries. In Ivan Visconti and Roberto De Prisco, editors, *SCN*, volume 7485 of *Lecture Notes in Computer Science*, pages 281–301. Springer, 2012.

[DF11] Stefan Dziembowski and Sebastian Faust. Leakage-resilient cryptography from the inner-product extractor. In *Advances in Cryptology–ASIACRYPT 2011*, pages 702–721. Springer, 2011.

[DF12] Stefan Dziembowski and Sebastian Faust. Leakage-resilient circuits without computational assumptions. In *Theory of Cryptography*, pages 230–247. Springer, 2012.

[Eke91] Artur K Ekert. Quantum cryptography based on bell's theorem. *Physical review letters*, 67(6):661–663, 1991.

[GR12] Shafi Goldwasser and Guy N Rothblum. How to compute in the presence of leakage. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 31–40. IEEE, 2012.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *STOC*, pages 723–732. ACM, 1992.

[LC99] Hoi-Kwong Lo and Hoi Fung Chau. Unconditional security of quantum key distribution over arbitrarily long distances. *Science*, 283(5410):2050–2056, March 1999.

[MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography. In *Theory of Cryptography*, pages 278–296. Springer, 2004.

[NV01] Michael A Nielsen and Guifré Vidal. Majorization and the interconversion of bipartite states. *Quantum Information & Computation*, 1(1):76–93, 2001.

[Ren05] Renato Renner. *Security of quantum key distribution*. PhD thesis, ETH Zürich, 2005.

[Win99] Andreas Winter. *Coding Theorems of Quantum Information Theory*. PhD thesis, Department of Mathematics, University of Bielefeld, 1999.

A Proof of Lemma 1

Proof. In the real world of [Can00] a computationally unbounded, semi-honest, static adversary B corrupting party i will receive the complete view of party i and will then output an arbitrary value b. In the ideal world there is a poly-time (in the running time of B) simulator S who gets just the inputs and outputs of party i and then must output a value b'. It is then required that for all A there exists S such that b along with the inputs and outputs of the protocol, (I, O), is statistically indistinguishable from b' concatenated with the inputs and outputs, (I', O') of f, i.e., the statistical distance is negligible in k. For unconditional security B is allowed unbounded time.

By the assumption of the lemma, there exists a poly-time simulator T demonstrating that π is a ϵ-leakage-resilient implementation of f against $\mathcal{P}_i$. This simulator T outputs a leakage function L' which is given the inputs and outputs of party i and then outputs a single bit, hence T can up to a few syntactical changes act as a simulator S_T for the model [Can00].

Assume that it is not the case that π is secure in the sense of [Can00] against a computationally unbounded, semi-honest, static adversary corrupting any one party. Let B be an adversary demonstrating the insecurity, in particular, for the simulator S_T the distributions (b, I, O) and (b', I', O') are not statistically indistinguishable. Then let D be an unbounded time Turing machine which can distinguish (b, I, O) and (b', I', O') with non-negligible probability in k: Using S_T the distinguisher can compute the distributions of (b, I, O) and (b', I', O') up to any precision, and then given a sample from one of them it can essentially make a maximum likelihood guess at which one it got a sample from.

Now consider the adversary $A_{B,D}$ which submits a leakage query (i, L), which has B and D hard-coded along with all the inputs and outputs to the protocol, which are known to A in our model. The leakage function is given view_i. Then it runs B on the view of party i to get b. Then it runs D on b and the inputs and outputs of the protocol, to get a bit g. Then it outputs g. Hence g is given to $A_{B,D}$ which can then output g. Clearly $A_{B,D} \in \mathcal{P}_i$, so by the assumption that π is a ϵ-leakage-resilient implementation of f against $\mathcal{A}$, the statistical distance between $\mathrm{exec}_{\pi, A_{B,D}}(k)$ and $\mathrm{sim}_{f, A_{B,D}, S_T}(k)$ is negligible in k.

However, in $\mathrm{exec}_{\pi, A_{B,D}}(k)$ the input to D inside the computation of the leakage function is distributed exactly as (g, I, O), and in $\mathrm{sim}_{f, A_{B,D}, T}(k)$ the input to D inside the computation of the leakage function is distributed exactly as (g', I', O'). This is a contradiction.

B Proof of Thm. 1

Proof. To simulate a leakage query $(i, \mathcal{I}, L)$, output a full-view simulator V which computes its output as follows: Use the full-view simulator for π_{A} to reconstruct the view of party i in π_{A} from the inputs and the outputs of party i, which are given as input to V. Then for each invocation of f_{B} in the simulated view, take the corresponding inputs and outputs of f_{B} and then use the full-view simulator for π_{B} and run it on these inputs and outputs to simulate a full view of π_{B}. Then insert this view of π_{B} into π_{A} to produce a view of $\pi_{\mathrm{A}}[\pi_{\mathrm{B}}/f_{\mathrm{B}}]$ – we say that we patch the evaluation of f_{B} with a simulation of π_{B}. The proof then follows from a hybrid argument showing that the simulated view cannot be distinguished from a real view given λ-tradeoff leakage. Start with a real run of $\pi_{\mathrm{A}}[\pi_{\mathrm{B}}/f_{\mathrm{B}}]$. Then one by one replace the runs of π_{B} by evaluations of f_{B} patched with a simulated view. The only difference between two hybrids is the difference between a simulated view of π_{B} and a real run of π_{B}. Call the one evaluation that was changed the disputed evaluation. Since the units of the λ-tradeoff adversary class are expanded original rounds of π_{A} it follows that the disputed evaluation of π_{B} is embedded in a leakage unit, i.e., we only leak on the disputed evaluation when we leak on this unit. Since we allow at most λ bits of leakage on each unit it follows that the difference can be distinguished with advantage at most ϵ_{B}. There are at most q_{B} runs of π_{B} per activation, which gives a total advantage of at most $\alpha q_{\mathrm{B}} \epsilon_{\mathrm{B}}$. Now we have a hybrid which is a run of π_{A} with each evaluation of f_{B} patched. Now replace the run of π_{A} with a simulation of π_{A}, and then patch this simulation instead. The leakage on these hybrids can be seen as λ-tradeoff leakage on a real run of π_{A} or a simulation of π_{A}: first patch and then leak. The advantage in distinguishing is therefore at most $\alpha \epsilon_{\mathrm{A}}$, for a total advantage of at most $\alpha(\epsilon_{\mathrm{A}} + q_{\mathrm{B}} \epsilon_{\mathrm{B}})$, as needed.