Leakage Resilient Secure Two-Party Computation

Ivan Damgård* Carmit Hazay* Arpita Patra†

Abstract

In the traditional *secure function evaluation* setting, some set of distrusting parties jointly compute a function of their respective inputs *securely* as if the computation is executed in an ideal setting where the parties send inputs to a trusted party that performs the computation and returns its result. Almost independently of secure computation, the area of *leakage resilient cryptography* has recently been evolving intensively, studying the question of designing cryptographic primitives that remain secure even when some information about the secret key is leaked.

In this paper we initiate the study of *secure two-party computation in the presence of leakage*, where on top of corrupting one of the parties the adversary obtains leakage from the content of the secret memory of the honest party.

Our study involves the following contributions:

1. *Security Definitions.* We formalize the notion of secure two-party computation in the presence of leakage and introduce security definitions in the *ideal/real framework*. Our formalization induces two types of adversarial attacks. We further study the feasibility of our definitions in the computational setting and explore some of the conditions under which these definitions are met.
2. *Composition Theorems.* We provide compositions theorems for our new modelings. Our results provide compositions theorems for the case where the inputs of the parties are sampled from a min-entropy source distribution.
3. *Leakage resilient oblivious transfer.* We present the first construction for 1-out-of-2 oblivious transfer with security against leakage of a constant fraction of the honest party's memory. Our protocol is based on the OT construction presented by Peikert et al. [PVW08].
4. *Leakage resilient Yao's Garbled Circuit [Yao82].* We provide the first general construction for secure two-party computation and show how to adapt the proof from [LP09] of Yao's protocol into the leakage resilient setting. Our result holds for a restricted set of functions due to technicalities arise in the proof.

1 Introduction

In the traditional *secure function evaluation* (SFE) setting, some set of distrusting parties jointly compute a function of their respective inputs *securely* as if the computation is executed in an ideal setting where the parties send inputs to a trusted party that performs the computation and returns its result. Starting with the work of [Yao82, GMW87, CCD88, BOGW88], it is by now well known that (in various settings) any polynomial-time computation can be generically compiled into a secure function evaluation protocol with polynomial complexity [JS07, LP09, IPS09, LP11].

Almost independently of secure computation, the area of *leakage resilient cryptography* has recently been evolving intensively. Leakage resilient cryptography study the question of designing cryptographic

*Department of Computer Science, Aarhus University, Denmark. Emails: {ivan,carmit}@cs.au.dk.

†ETH Zurich, Switzerland. Email:{arpita.patra}@inf.ethz.ch.

primitives that remain secure even when some information about the secret key is leaked. In other words, the absolute privacy of secret keys is no longer a basic assumption. This setting has been demonstrated to be more realistic after it has been noticed that a large class of attacks, denoted *side-channels attacks*, are not captured by existing models [Koc96, BS97, KJJ99, HSH+09]. In order to deal with these threats much research has been done recently on leakage resilient cryptography with the aim to design cryptographic primitives with resistibility to such attacks. Most of the work has been dedicated to construct leakage resilient encryption schemes [Riv97, MR04, DP08, AGV09, Pie09, NS09, DGK+10, BKKV10, LLW11], but also leakage resilient signatures schemes [KV09, FKPR10, MTVY11, BSW11], identity-based encryption schemes [ADN+10, LRW11] and more.

One may distinguish two types of frameworks for modeling security in the presence of leakage, that are considered in the literature. The *only computational* model introduced by Micali and Rayzin [MR04], and the *memory leakage* model introduced by Akavia et al [AGV09]. The only computational model follows the basic assumption that *only computation leaks information*, that is, only the memory portions which are involved in the computation may leak information. This assumption led to the design of a number of cryptographic primitives, demonstrating security under this assumption [MR04, GKR08, DP08, PSP+08, Pie09]. The memory leakage modeling is more general and enables the adversary to learn an *arbitrary* information about the secret state kept in the memory, as long as the amount of information leaked is bounded.

In this paper we study the problem of *secure two-party computation in the presence of leakage*. More concretely, our setting considers a *distributed* environment where two parties, who do not trust each other, run a protocol to compute some functionality securely. The adversary is strengthened in this setting and is allowed to observe leakage from the content of the secret memory of the honest party (which includes both its *input* and *randomness*), in addition to corrupting the other party. Leakage attacks are formalized by allowing the adversary to submit leakage functions to a leakage oracle to be applied on the secret memory of the honest party during the protocol execution.

The particular case of leakage resilient zero-knowledge was considered by Garg et al. in [GJS11]. In their setting, a cheating verifier is allowed to obtain arbitrary bounded leakage on the state of the prover during the entire protocol execution. In a related concurrent work, Canetti et al [BCH11] consider leakage resilient secure computation of general functionalities. They show that a (multi-party) protocol that is adaptively UC-secure against corruption of a party is also secure if instead the adversary is allowed to leak from that party. The security guarantees that leakage one gets are the same as if that party had been corrupted. This means that in the two-party setting, security guarantees can be given for the case where one party is corrupted and the adversary leaks from the other, if the underlying adaptively secure protocol is secure in the presence of both parties being corrupted at the same time. In their paper, Canetti et al demonstrate a construction of an oblivious transfer protocol built on top of a non-committing encryption scheme. Their construction is proven in the memory leakage model but uses communication with a blowup that is proportional to the security parameter for each bit sent within the oblivious transfer protocol, due to the overhead induced by the non-committing encryption. On the other hand, our construction is proven in the only computational model but is communication efficient. We wish to note that the overhead of [BCH11] is not inherent and depends heavily on the overhead of the non-committing scheme.

Our contribution. In short, our study involves the following contributions:

1. **Security Definitions.** We formalize the notion of secure two-party computation in the presence of leakage and introduce security definitions in the *ideal/real framework*. Our formalization induces two types of adversarial attacks. We further study the feasibility of our definitions in the computational setting and explore some of the conditions under which these definitions are met.

2. **Composition Theorems.** We provide compositions theorems for our new modelings. Our results provide compositions theorems for the case where the inputs of the parties are sampled from a min-entropy source distribution. This captures the scenario that the honest party's input is not entirely known to the adversary.

3. **Leakage resilient oblivious transfer.** We present the first construction for 1-out-of-2 oblivious transfer with security against leakage of a constant fraction of the honest party's memory. Our protocol is based on the OT construction presented by Peikert et al [PVW08]. We get leakage resilient protocols for a more general class of functions (but not all functions) by combining the OT protocol with the construction based Yao-garbled circuits mentioned below. This works because the Yao construction exactly requires the OT's to be called on random inputs, so that they have high entropy, which is a requirement in our model. We believe that we can also use the fact that OT is complete for secure computation to get a leakage resilient protocol for arbitrary functions. At the time of writing, we are not ready to claim this result, but we will report on this in the final version of the paper.

4. **Leakage resilient Yao's Garbled Circuit [Yao82].** We provide the first general construction for secure two-party computation and show how to adapt the proof from [LP09] of Yao's protocol into the leakage resilient setting. Our protocol uses statistically secure secret key scheme for the circuit garbling and thus is only secure for logarithmic depth circuits. Due to technical issues arise in the proof we further require that the computed function would have an additional property of efficiently sampling an input from the domain that is consistent with a given output and an input of the adversary.

1.1 Our Results in More Details

In this work we focus on a *static semi-honest* adversary. Typically, this modeling offers weak security since it restricts the adversary to follow the protocol's description. Nevertheless, when strengthening the adversary in the context of leakage attacks, all known constructions easily break down. This follows from the fact that the adversary is capable of observing leakage from the secret memory content of the honest party (that is not under its complete control). In particular, all building block cryptographic primitives used for constructing protocols must withhold leakage attacks. This poses new challenges already in the presence of semi-honest adversaries, since security of these primitives must be ensured even if the adversary leaks from the randomness used during the involved computations, or even during intermediate computations made on the work tape of the honest party.

Security Definitions. This modification induces two types of adversarial attacks: (**1**) A **global** leakage model, inspired by the memory leakage modeling [AGV09, NS09], where the adversary obtains leakage specified by the output of a probabilistic polynomial (PPT) time function h, applied on the input and the *entire* randomness of the honest party. (**2**) A weaker notion of **local** leakage, inspired by the "only computational" framework of Micali-Rayzin [MR04], is where the adversary chooses different leakage functions at different points of time during the protocol execution based on its view and prior leakage. However, each leakage function is applied only on the *portion* of the memory content required for the current computation.

We formalize the notion of secure two-party computation in the presence of leakage and introduce security definitions in the *ideal/real framework*. Intuitively, leakage attacks are formalized by allowing the adversary to submit leakage functions to be applied on the secret memory of the honest party during the protocol execution. In the ideal setting, the ideal adversary (simulator) can submit leakage queries to the trusted party on behalf of the real adversary. Formalizing this intuition is indeed a challenging task, since one has to make sure that the simulator does not abuse the trusted party for obtaining "too much" information. Typically, this can be ensured by requiring that the loss of entropy from the honest party's input, due to

leakage from the secret memory of the honest party, is *the same* in both real and ideal settings. This is insufficient in our context since the real adversary can conclude further information about the honest party's input when observing the *combination* of the leakage and its view. Therefore, the right notion that captures leakage attacks in this scenario is to observe the min-entropy drop of the honest party's input with respect to *both* the view and leakage obtained by the adversary.

Finally, in Section 3.3 we study the feasibility of our definitions in the computational setting and show for that a broad class of hardness assumptions security with leakage in the global setting cannot be met. Interestingly, in Section 5 we show how to overcome some of these difficulties in the local leakage setting.

Modular composition. Composition theorems constitute an extremely useful and important tool for simplifying proofs of security. In Section 4, we provide composition theorems for the global and local leakage settings for the case where the inputs of the parties are sampled from a min-entropy source distribution.

Leakage resilient oblivious transfer. Oblivious transfer (OT) is one of the basic and important building blocks for cryptographic protocols and has been studied extensively in the literature based on various hardness assumptions, providing different notions of security and efficiency; see [NP05, PVW08] for just two examples. We present the first leakage resilient OT protocol in the local leakage model that is secure against leakage of a constant fraction of the honest party's memory. Our protocol is based on the OT construction presented by Peikert et al [PVW08]. We modify this construction and prove it secure in our model under a variant of the DDH assumption which is equivalent to the modified DDH assumption by Canetti [Can97]. This assumption can be proven in the generic group model. We further show how to circumvent the problem raised in Section 3.3 with respect to computational security against leakage. We note that our simulators for both the OT proof, and the Yao proof specified below, are straight-line and do not rely on any trusted setup assumption.

Leakage resilient Yao's garbled circuit [Yao82]. We present an implementation of Yao's construction in the local model that is secure against leakage of a constant fraction of the honest party's memory. For that, we adapt the proof of Lindell and Pinkas for the semi-honest case [LP09] and show that security holds with respect to local leakage attacks. Our protocol uses statistically secure secret key scheme for the circuit garbling and thus is only secure for logarithmic depth circuits. Due to technical issues arise in the proof we further require that the computed function would have an additional property of efficiently sampling an input from the domain that is consistent with a given output and an input of the adversary. We describe our proof in the $\mathcal{F}_{\mathrm{OT}}$ hybrid model (where OT ideal calls are made), and then use our composition theorem to combine this proof with our leakage resilient OT.

Further research. This work opens a new area of research regarding secure computation with leakage and leaves many open directions to explore. One of the most interesting question left open by this work is to provide a full characterization of which functionalities can be computed in the global and local models. The modeling we considered here applies only for the two-party with passive adversaries. Exploring any dimension of these definitions would be interesting, i.e., considering malicious behavior, the multiparty setting (with and without assuming honest majority), or strengthening the adversary by letting it tamper with the secret memory of the honest party.

1.2 A Road Map

In Section 2 we overview some standard definitions. In Section 3 we present our models and give a detailed discussion about our definitions. In Section 4 we present composition theorems for our settings. In Section 5

we describe our leakage resilient oblivious transfer construction. We conclude with our implementation of Yao garbled circuit in the local leakage setting in Section 6.

2 Definitions

We denote the security parameter by n. A function $\mu(\cdot)$ is negligible if for every polynomial $p(\cdot)$ there exists a value N such that for all $n > N$ it holds that $\mu(n) < \frac{1}{p(n)}$. We use PPT to denote probabilistic polynomial-time in short. We specify some useful definitions.

Definition 2.1 (Computational indistinguishability) *Let $X = \{X_n(a)\}_{n\in\mathbb{N},a\in\{0,1\}^*}$ and $Y = \{Y_n(a)\}_{n\in\mathbb{N},a\in\{0,1\}^*}$ be distribution ensembles. We say that X and Y are* computationally indistinguishable*, denoted $X \approx_c Y$, if for every* PPT *D, there exists a negligible function $\mu(\cdot)$ such that for all $a \in \{0,1\}^*$,*

$$|\Pr[D(X_n(a)) = 1] - \Pr[D(Y_n(a)) = 1]| < \mu(n)$$

Definition 2.2 (Statistical distance) *Let X_n and Y_n be random variables accepting values taken from a finite domain $\Omega \subseteq \{0,1\}^n$. The* statistical distance *between X_n and Y_n is*

$$SD(X_n, Y_n) = \frac{1}{2}\sum_{\omega\in\Omega} |\Pr[X_n = \omega] - \Pr[Y_n = \omega]|.$$

*We say that X_n and Y_n are ϵ-*close *if their statistical distance is at most $SD(X_n, Y_n) \leq \epsilon(n)$. We say that X_n and Y_n are* statistically close*, denoted $X_n \approx_s Y_n$, if $\epsilon(n)$ is negligible in n.*

We state the Leftover Hash Lemma. We say that $H = \{h\colon D \to R\}$ is a pairwise independent family of functions if for all $x \neq y \in D$ and $a, b \in R$, $\Pr_{h\in H}[h(x) = a \;\&\; h(y) = b] = 1/R^2$. We say that X is a k-source over D if it is a distribution over elements $x \in D$ with min-entropy at least $H_\infty(X) \geq k$; where $H_\infty(X) = \min_{v\in D}(-\log \Pr[X = v])$. Denote by $(H, H(X))$ the distribution over pairs $(h, h(x))$ with h, x chosen independently at random from H and X respectively. Denote by $U_{H\times R}$ the uniform distribution over domain $H \times R$.

Theorem 2.3 (Leftover Hash Lemma) *For any integer $k > 0$, real $\epsilon > 0$, a k-source X over D, and a pairwise independent family of functions $H = \{h\colon D \to R\}$,*

$$SD((H, H(X)), U_{H\times R}) \leq \sqrt{|R|/2^k}/2$$

In particular, if $\log|R| \leq k - 2\log(1/\epsilon)$, then $SD((H, H(X)), U_{H\times R}) \leq \epsilon$.

A few useful facts on the statistical distance follow [SV03].

Fact 2.1 *Let $X = (X_1, X_2)$, $Y = (Y_1, Y_2)$ be random variables accepting values taken from domain $D_1 \times D_2$. If X_1, X_2 are independent random variables, and Y_1, Y_2 are independent random variables, then $SD(X, Y) \leq SD(X_1, Y_1) + SD(X_2, Y_2)$.*

Fact 2.2 *Let X, Y, Z be random variables accepting values taken from domain D. Then,*

1. *$SD(X, Y) \leq SD(X, Z) + SD(Z, Y)$.*
2. *For any randomized procedure A, $SD(A(X), A(Y)) \leq SD(X, Y)$.*

The notion of average min-entropy [DRS04], captures the remaining uncertainty of the random variable X conditioned on the value of another random variable Y. More formally,

$$\begin{aligned}\tilde{H}_\infty(X|Y) &= -\log\left(\mathrm{E}_{y\to Y}\left[\max_x \Pr[X=x|Y=y]\right]\right)\\ &= -\log\left(\mathrm{E}_{y\to Y}\left[2^{-H_\infty(X|Y=y)}\right]\right)\end{aligned}$$

Dodis et al. [DRS04] showed that the leftover hash lemma can be realized using this notion as well. They also prove the following useful lemma.

Lemma 2.3 ([DRS04]) *Let X, Y, Z be random variables.*

1. *For any $\delta > 0$, the conditional entropy $H_\infty(X|Y=y)$ is at least $\tilde{H}_\infty(X|Y) - \log(1/\delta)$ with probability at least $1-\delta$ over the choice of y.*
2. *If Y has at most 2^λ possible values, then $\tilde{H}_\infty(X|(Y,Z)) \geq \tilde{H}_\infty(X|Z) - \lambda$.*

An important tool in our protocols is a strong extractor.

Definition 2.4 ([DRS04]) *A function* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ *is an* average-case (k,ϵ) strong extractor *if for all pairs of random variables (X,I) such that $X \in \{0,1\}^n$ and $\tilde{H}_\infty(X|I) \geq k$ it holds that*

$$SD((\mathsf{Ext}(X,S),S,I),(U_m,S,I)) \leq \epsilon$$

for a uniformly $S \in \{0,1\}^m$.

We specify the concrete parameters when introducing our constructions.

2.1 Secret Key Encryption Schemes

We specify the definitions of secret key encryption and semantic security for secret key schemes.

Definition 2.5 (SKE) *We say that* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a* secret key encryption scheme *if* $\mathsf{Gen}, \mathsf{Enc}$ *and* Dec *are polynomial-time algorithms specified as follows:*

- Gen*, given a security parameter n (in unary), outputs a secret key sk. We denote this by $sk \leftarrow \mathsf{Gen}(1^n)$.*
- Enc*, given the secret key sk and a plaintext message m, outputs a ciphertext c encrypting m. We denote this by $c \leftarrow \mathsf{Enc}_{sk}(m)$; and when emphasizing the randomness r used for encryption, we denote this by $c \leftarrow \mathsf{Enc}_{sk}(m;r)$.*
- Dec*, given the secret key sk and a ciphertext c, outputs a plaintext message m s.t. there exists randomness r for which $c = \mathsf{Enc}_{sk}(m;r)$ (or $\perp$ if no such message exists). We denote this by $m \leftarrow \mathsf{Dec}_{sk}(c)$.*

For a secret key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and a non-uniform adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we consider the following Semantic security game:

$sk \leftarrow \mathsf{Gen}(1^n)$.
$(m_0, m_1, history) \leftarrow \mathcal{A}_1^{\mathsf{Enc}_{sk}(\cdot)}$, s.t. $|m_0| = |m_1|$.
$c \leftarrow \mathsf{Enc}_{sk}(m_b)$, where $b \leftarrow \{0,1\}$.
$b' \leftarrow \mathcal{A}_2^{\mathsf{Enc}_{sk}(\cdot)}(c, history)$.
$\mathcal{A}$ wins if $b' = b$.

Denote by $\mathsf{Adv}_{\Pi,\mathcal{A}}(n)$ the probability that $\mathcal{A}$ wins the semantic security game.

Definition 2.6 (Semantic security) *A secret key encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* semantically secure*, if for every* PPT *adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *there exists a negligible function* negl *such that* $\mathsf{Adv}_{\Pi,\mathcal{A}}(n) \leq \frac{1}{2} + \mathsf{negl}(n)$.

2.2 Hardness Assumptions

We first define the well known Decisional Diffie-Hellman assumption.

Assumption 2.4 (DDH) *We say that* the decisional Diffie-Hellman (DDH) problem is hard relative to $\mathbb{G} = \{\mathbb{G}_n\}$ *if for all* PPT *algorithms* $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$\left| \Pr\left[\mathcal{A}(\mathbb{G}, p, g, g^x, g^y, g^z) = 1\right] - \Pr\left[\mathcal{A}(\mathbb{G}, p, g, g^x, g^y, g^{xy}) = 1\right] \right| \leq \mathsf{negl}(n),$$

where p *is the order of* $\mathbb{G}$ *and the probabilities are taken over the choices of* $g \in \mathbb{G}$ *and* $x, y, z \in \mathbb{Z}_p$.

We consider two additional assumptions that are useful in the context of leakage resilient secure computation. In [Can97] Canetti gave stronger variants of the DDH assumption. We specify the following assumption that holds in the generic group model.

Assumption 2.5 (k-DDH) *We say that* the decisional Diffie-Hellman for k-sources (k-DDH) problem is hard relative to $\mathbb{G} = \{\mathbb{G}_n\}$ *if for all* PPT *algorithms* $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$\left| \Pr\left[\mathcal{A}(\mathbb{G}, p, g, g^x, g^y, g^z) = 1\right] - \Pr\left[\mathcal{A}(\mathbb{G}, p, g, g^x, g^y, g^{xy}) = 1\right] \right| \leq \mathsf{negl}(n),$$

where p *is the order of* $\mathbb{G}$ *and the probabilities are taken over the choices of* $g \in \mathbb{G}$, $y, z \in \mathbb{Z}_p$ *and* $x \in X$ *for a* k*-source* X *over* $\mathbb{Z}_p$.

In his paper Canetti requires that X is a well-spread distribution, i.e., $k = \omega(\log n)$.[1] We reformulate the k-DDH assumption via a game in which all exponents are chosen uniformly, but the adversary gets leakage from one of them. This will fit into our application later. Formally, for $\mathbb{G}, g$ and p as above, we consider the following k-DDH game:

$$\begin{aligned}
&a, b, c \leftarrow \mathbb{Z}_p.\\
&f \leftarrow \mathcal{A}_1.\\
&T = \left(g^a, g^b, g^{(ab)^\alpha c^{1-\alpha}}\right), \text{ where } \alpha \leftarrow \{0,1\}.\\
&\alpha' \leftarrow \mathcal{A}_2(f(a), T).\\
&\mathcal{A} \text{ wins if } \alpha' = \alpha.
\end{aligned}$$

Denote by $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{k-DDH}}(n)$ the probability that $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ wins the k-DDH game. Note that in case $\alpha = 0$ the adversary's view is $T = \left(g^a, g^b, g^c\right)$ and $f(a)$ whereas, in case $\alpha = 1$ the adversary's view is $T = \left(g^a, g^b, g^{ab}\right)$ and $f(a)$. In the following claim we prove that with high probability the min entropy of a conditioned on the leakage is well spread and thus hardness of k-DDH implies hardness in the k-DDH game with high probability.

Claim 2.6 *Let* f *be a function with leakage rate* $1 - \omega(\log n)/\log p$*, and assume that*

$$\left| \Pr\left[\mathcal{A}(\mathbb{G}, p, g, g^a, g^b, g^c) = 1\right] - \Pr\left[\mathcal{A}(\mathbb{G}, p, g, g^a, g^b, g^{ab}) = 1\right] \right| \leq \mathsf{negl}(n),$$

for a, b, c *as in the k-DDH assumption. Then,* $\mathcal{A}$ *wins the k-DDH game with probability at most* $1/2 + \mathsf{negl}'(n)$ *for some negligible function* $\mathsf{negl}'(\cdot)$.

[1] For simplicity we take $n = \log p$.

Proof: Consider the distribution on a in the two cases. In the former case of the k-DDH assumption, a is picked from a $\omega(\log n)$-source over $\mathbb{Z}_p$. We show that with some probability $1-\delta$ over the choice of a and the leakage function f, a is k-source in the k-DDH game as well. This implies that hardness of k-DDH translates to hardness in the k-DDH game with probability (at least) $1-\delta$. More formally, we use here Lemma 2.3, Item 1, that argues that for any $\delta > 0$, $H_\infty(a|f(a)) \geq \tilde{H}_\infty(a|f(a)) - \log(1/\delta)$ with probability $1-\delta$. Item 2 (and a generalization by Naor-Segev [NS09]) shows that $\tilde{H}_\infty(a|f(a)) \geq H_\infty(a) - \lambda$ when $f(\cdot)$ has at most 2^λ possible values. Combining the two claims we get that with probability $1-\delta$ $H_\infty(a|f(a)) \geq H_\infty(a) - \lambda - \log(1/\delta)$.

Since f is a function with leakage rate $1-\omega(\log n)/\log p$ by our assuption, the size of an output of f is $n - \omega(\log n)$. So there can be $2^{n-\omega(\log n)}$ possibilities for $f(\cdot)$. In particular, we fix $\lambda = n - \log^c n$ for $c > 2$ and $\delta = 1/2^{\log^2 n}$. Then with probability $1 - 1/2^{\log^2 n}$ (which is negligibly close to 1), we get that $H_\infty(a|f(a)) \geq \log^c n - \log^2 n \geq \log^{c-1} n$ which implies that the distribution on a is as required for the k-DDH assumption with overwhelming probability. This means that the advantage of $\mathcal{A}$ in game k-DDH can only increase by δ. ∎

Finally, we consider the following indistinguishability assumption k-IND in the only computational model that is implied by k-DDH.

Assumption 2.7 (k-IND) *We say that* indistinguishability for k-sources (k-IND) problem is hard relative to $\mathbb{G} = \{\mathbb{G}_n\}$ *if for all PPT algorithms $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function* negl *such that* $\mathsf{Adv}^{\mathrm{k-IND}}_{\mathcal{A}}(n) \leq \frac{1}{2} + \mathsf{negl}(n)$. *Where* $\mathsf{Adv}^{\mathrm{k-IND}}_{\mathcal{A}}(n)$ *is the probability $\mathcal{A}$ wins the following game:*

$$
\begin{aligned}
&a, b \leftarrow \mathbb{Z}_p.\\
&f \leftarrow \mathcal{A}_1.\\
&g \leftarrow \mathbb{G}\\
&T = \left(g^{(a)^\alpha b^{1-\alpha}}\right), \text{ where } \alpha \leftarrow \{0,1\}.\\
&\alpha' \leftarrow \mathcal{A}_2(f(a), T).\\
&\mathcal{A} \text{ wins if } \alpha' = \alpha.
\end{aligned}
$$

f be a function with leakage rate $1-\omega(\log n)/\log p$.

By an argument similar to the one we gave for the previous game, it can be seen that the effect of giving $f(a)$ to the adversary is that the a priori distribution of a is that of a k-source. Importantly, a generator g is only picked *after* $\mathcal{A}$ pickes the leakage function, since otherwise the assumption would not hold trivially by hardwiring g into the leakage function. In order to see that k-IND is implied by k-DDH consider an intermediate game that is identical to game k-DDH, except that $\mathcal{A}$ is handed $\left(f(a), g^{(ab)^\alpha c^{1-\alpha}}\right)$ instead of $\left(f(a), g^a, g^b, g^{(ab)^\alpha c^{1-\alpha}}\right)$. Clearly, this game is as hard as the k-DDH game. We now claim that it implies game k-IND, where g^b is the new generator g' for k-IND. This is because b or g^b are not leaked at any point and b is picked at uniform. Thus $\mathcal{A}$'s view is either distributed according to $\left(f(a), g^{ab} = g'^a\right)$ or $\left(f(a), g^c = g'^{c'}\right)$ for some $c' \in \mathbb{Z}_p$, exactly as in game k-IND.

3 Modeling Leakage Resilient Secure Two-Party Computation

In this section we formalize secure two-party computation in the presence of leakage attacks. Our modeling captures adversarial attacks in which the adversary obtains leakage from the content of the secret memory of the honest party in addition to passively controlling the corrupted party. Leakage attacks are formalized

by allowing the adversary to submit leakage functions to be applied on the secret memory of the honest party during the protocol execution. Typically, passive (semi-honest) corruption offers a rather weak notion of security since it only allows the adversary to observe the content of the secret memory of the corrupted party. Nevertheless when introducing leakage attacks into the model many subtleties arise already for this corruption type, mainly due to the fact that we aim for simulation based security. In this framework, one first considers an ideal model in which the two parties are joined by an incorruptible trusted third party (TTP) and the computation is performed with the help of a TTP. Next, one considers a real model in which a real two-party protocol is executed (without the help of a TTP). A protocol in the real model is said to be secure with respect to certain adversarial attacks if the real execution, with the specified adversary, can be *simulated* in the corresponding ideal model. Namely, simulation of the joint view of both parties in the real world can be achieved in the ideal world. This guarantees that any attack carried out in a real execution can also be carried out with a trusted party in an ideal execution.

When introducing leakage into the model, we allow the ideal adversary (simulator) to submit leakage queries to the trusted party on behalf of the real adversary, to be applied on the secret memory of the honest party. In order for the security definition to be meaningful one has to ensure that the simulator does not abuse the trusted party for obtaining "too much" information. Typically, this can be enured by requiring that the loss of entropy due to leakage from the secret memory of the honest party is *the same* in both real and ideal settings. Nevertheless, this is insufficient in our context since the adversary can conclude further information about the honest party's input when observing the *combination* of the leakage and its view. Therefore, the right notion that captures leakage attacks in this scenario is to observe the min-entropy drop of the honest party's input with respect to both the view and leakage obtained by the adversary. We further elaborate on this intuition below when introducing our formal definitions.

Denoting the participating parties by P_0 and P_1, we consider a functionality by $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$, where $f = (f_0, f_1)$ mapping pairs of inputs to pairs of outputs (one for each party). That is, for every pair of inputs $x_0, x_1 \in \{0,1\}^n$ the output pair is a random variable $(f_0(x_0, x_1), f_1(x_0, x_1))$ ranging over pair of strings. The first party with input x_0 wishes to receive $f_0(x_0, x_1)$, while the second party with input x_1 wishes to obtain $f_1(x_0, x_1)$.

In order to withhold leakage attacks, the designer of a protocol has to take into account the amount of leakage that the protocol can tolerate. Typically, this is formalized by introducing a new parameter $\alpha < 1$ that corresponds to the leakage rate of the system. Let s be the content of the secret memory of the honest party and h be a polynomial-time function submitted by the adversary. Then the leakage rate can be formalized in one of the following ways:

1. In the *output length shrinking* modeling due to Akavia et al. [AGV09], h must have a short output so that $|h(s)| \leq \alpha|s|$.

2. In the slightly more general *entropy shrinking* modeling due to Naor and Segev [NS09] and Dodis et al. [DHLAW10b], the average min-entropy left in s after learning $h(s)$ is at least $(1-\alpha)$ times the initial min-entropy of the secret. Namely, $\tilde{H}_\infty(s|\, h(s)) \geq (1-\alpha) \cdot H_\infty(s)$.[2]

3. The most general modeling, due to Dodis et al. [DKL09], is to hand the corrupted party the outcome of a hard to invert leakage function h (with an arbitrary output length), applied on the secret memory of the honest party. Here h is a hard-to-invert function picked from some specified family with respect to a hardness parameter t. Note that conditioned on $h(s)$, s might have had no min-entropy.

Our results hold with respect to the former two formalizations.

[2] In some cases, the remaining min-entropy is computed condition on additional public parameters, e.g., the public key. Looking ahead, this is also true in our context where we wish to conditioned on the adversary's view as well.

Input distribution. In order to ensure that the honest party's input is not entirely known to the adversary, we assume that the parties' inputs (x_0, x_1) are drawn from some k-source distribution for $k = \omega(\log n)$. This uncertainty should hold even in the presence of auxiliary information (gathered by the adversary prior to the protocol execution). Note that such a definition is different than a definition where the parties' inputs are chosen adaptively based on the auxiliary input z, with the aim to maximize the ability to distinguish between the real and the simulated executions (as for instance in the UC framework [Can01]). This is because by letting the distinguisher to adaptively pick the inputs, it might pick correlated inputs for the parties. In this case we cannot ensure that the honest party's input maintains any average min-entropy conditioned on the input of the corrupted party. We formalize security relative to an (efficient) k-source distribution ensemble $X_f^k = \{X_n^0, X_n^1\}_{n\in\mathbb{N}}$ where a k-source distribution ensemble is defined as follows.

Definition 3.1 (A k-source distribution ensemble) *We say that $X_f^k = \{X_n^0, X_n^1\}_{n\in\mathbb{N}}$ is a k-*source distribution ensemble *for a function f if for all n, $\tilde{H}_\infty(X_n^i |\ x_{1-i}, z) \geq k$ for all $i \in \{0,1\}$, for $x_{1-i} \leftarrow X_n^{1-i}(1^n)$ and z is the auxiliary input given to the adversary.*

Assuming that $|x_0| = |x_1|$, we point out that k can be any value between 1 and $|x_0|$ and is determined according to the protocol's needs. For most interesting cases $k/|x_0|$ is smaller than the leakage rate α. We model the process of sampling the inputs by a probabilistic polynomial-time machine $M_{D_f^k}$ that is given z and a security parameter 1^n and outputs a pair of inputs x_0, x_1 sampled according to some k-source distribution ensemble D_f^k. We denote such a machine by a k-source input sampler. Note that this requires from the sampler to be given z due to the following reasons. First, we would like to capture a more realistic scenario in which the inputs are picked with consistency based on prior knowledge. Second, and more acute, we cannot ensure that the input of the honest party retains any average min-entropy given the auxiliary input if the later is not given to the sampler. Finally, as we show in Section 4, composition theorems would not hold otherwise. An important issue emerges with this modeling in the context of leakage is that it must be specified whether the randomness used for the sampling process is leaked as well (i.e., the randomness used for producing the inputs). We note that such a leakage is tolerable as long as the average min-entropy of the honest party's input is still above some threshold conditioned on the auxiliary input, the input of the corrupted party and the overall leakage.

Modeling the adversary. We strengthen the adversary by adding a new dimension to its power so that it obtains leakage from the secret memory of the honest party as well. This modification induces two types of adversarial modelings:

1. A **Global** leakage model where the adversary obtains leakage specified by the output of probabilistic polynomial-time functions applied on the input and the *entire* randomness of the honest party at the onset of the computation. For simplicity, we assume that the adversary submits a single leakage function since in this modeling adaptive choices of multiple leakage functions can be hardwired into a single function.

   While the global leakage definition is quite natural, it also makes a very strong requirement. In fact, as we discuss later, it may not even be possible to achieve it. We therefore consider a weaker notion, called as *local leakage model*.

2. The **Local** leakage model enables the adversary to choose different leakage functions at different points of time during the protocol execution based on its view and prior leakage. However, each leakage function is applied only on the *portion* of the memory content required for the computation at the specific time.

Global leakage security implies local leakage security as long as the overall amount of leakage obtained in the local setting does not exceed the overall leakage in the global setting. We present both computational and statistical variants of the definitions. As we discuss later, it is not clear that the standard impossibility results for information theoretic secure two-party computation apply in our setting, so both variants may be relevant.

3.1 Security in the Presence of Global Leakage and Semi-Honest Adversaries

We continue with formalizing the notion of secure computation in the presence of global leakage and semi-honest adversaries, and consider a definition of security for a two-party computation in the "real-vs.-ideal" framework. Formally, let the content of the secret memory of party P_i at the onset of the computation be $S_i = (x_i, R_i)$, where x_i denotes the initial input of P_i and R_i contains all the randomness generated by P_i to be used throughout the protocol execution. Define an ideal model with respect to an ideal adversary (simulator) $\mathcal{S}$ that corrupts party P_{i^*} as follows.

The Ideal Model. An ideal execution of a function $f = (f_0, f_1)$ proceeds as follows:

INPUTS: Let x_i denote the input of party P_i, where $(x_0, x_1) \leftarrow M$ for a k-source input sampler machine M. Moreover, $\mathcal{S}$ is given an auxiliary input z.

SEND INPUTS TO TTP: The honest party P_{1-i^*} sends its input x_{1-i^*} to TTP. In addition, adversary $\mathcal{S}$, corrupting party P_{i^*}, sends x_{i^*} and an efficiently computable bounded output leakage function $h^{\mathrm{sm}}_{1-i^*}$ to TTP. (We note that the choice of $h^{\mathrm{sm}}_{1-i^*}$ may depend on z).

TTP SENDS OUTPUT TO ADVERSARY $\mathcal{S}$: TTP computes $h^{\mathrm{sm}}_{1-i^*}(x_{1-i^*})$, $y_{i^*} = f_{i^*}(x_0, x_1)$ and sends these to $\mathcal{S}$.

TTP SENDS OUTPUT TO HONEST PARTY P_{1-i^*}: TTP computes $y_{1-i^*} = f_{1-i^*}(x_0, x_1)$ and sends it to P_{1-i^*}.

OUTPUTS: The honest party always returns its input and the output received from the TTP.[3] An ideal adversary may output an arbitrary (polynomial-time computable) function of its initial state (including its input, auxiliary input and random tape) and the messages it has obtained from the TTP.

We say that an ideal adversary $\mathcal{S}$ is $(\lambda^{\mathrm{sm}}_0, \lambda^{\mathrm{sm}}_1)$-admissible if the leakage rate of h_{1-i^*} is upper-bounded by $\lambda_{1-i^*} = \lambda_{1-i^*}(n)$ for all $i^* \in \{0, 1\}$. For an ideal adversary $\mathcal{S}$, denote an ideal execution of f on inputs produced by M, auxiliary input z and a security parameter n, by $\mathbf{Gideal}_{f,\mathcal{S},M}(n, z)$ to be the output pair of the honest party P_{1-i^*} and adversary $\mathcal{S}$ in the above ideal execution.

The Real Model. We now turn to describe an execution in the real model. We consider a real model in which a two-party protocol π is executed between P_0 and P_1 (without the help of a third trusted party), to evaluate a function $f = (f_0, f_1)$ on inputs x_0, x_1, where $(x_0, x_1) \leftarrow M$ for a k-source input sampler machine M. We model the real adversary as follows. Upon corrupting party P_{i^*}, $\mathcal{A}$ submits a leakage function h_{1-i^*} and receives back $h_{1-i^*}(S_{1-i^*})$. It then participates in an execution of a protocol π on behalf of P_{i^*}. As noted above, the leakage can be carried out at any point during the protocol execution as long as

[3] Typically, the honest party only returns its output in this setting since by indexing the distributions ensembles of the ideal and real executions with the parties' inputs and auxiliary input, the distinguisher gets these values as input and is able to verify correctness. However, since in this modeling the inputs are picked via a randomized process (with randomness unknown to the distinguisher), it must receive the inputs directly from the honest party. Else, even correctness is not confirmed.

the honest party does not erase its randomness. We define a (λ_0, λ_1)-admissible semi-honest adversary in the presence of global leakage as follows.

Definition 3.2 ((λ_0, λ_1)-global admissible adversary) *An adversary $\mathcal{A}$ is called* (λ_0, λ_1)-global admissible *adversary if for all $i^* \in \{0,1\}$, $\mathcal{A}(1^n, z, x_{i^*})$ submits a* PPT *leakage function h_{1-i^*} to the leakage oracle such that the leakage rate of h_{1-i^*} is upper bounded by $\lambda_{1-i^*} = \lambda_{1-i^*}(n)$, for P_{i^*} the corrupted party and n the security parameter.*

Without loss of generality, we assume that the adversary outputs its entire view and the leakage it obtains during the execution of π. A real execution of π in the global leakage setting on inputs sampled by a k-source sampler machine M, auxiliary input z and a security parameter n is denoted by $\mathbf{Greal}_{\pi,\mathcal{A},M}(n, z)$ and defined as the output pair of the honest party and an adversary $\mathcal{A}$ upon completing the execution of π.

We are now ready to formally define secure computation in the presence of global leakage. Our definition is composed of two parts. First, in order to guarantee that the simulator is not "too powerful" we wish to restrict it so that the set of leakage functions that it submits to the trusted party must have the following property: the remaining (average) min-entropy of the honest party's input conditioned on the ideal leakage is the same when conditioned on the real leakage. Unfortunately, this is insufficient when modeling leakage in the context of secure computation because the adversary can easily obtain more information about the honest party's input when observing the combination of the leakage and its view. For instance, the leakage function can be designed to leak from the messages that would have been sent by the honest party on an arbitrary input, say x'_{1-i^*}, and randomness R. This leakage function does not leak anything about the real input x_{1-i^*}. Nevertheless, when considering this leakage *together with* the real view of the adversary, the simulation is most likely to fail since the adversary can easily compare the leakage against the view and see whether there is a difference (which will be in the simulated execution since it does not know x_{1-i^*}). We therefore allow the simulator to obtain the *same amount* of leakage that is obtained by observing the leakage of the real adversary and its view.

One way to formalize this intuition is by requiring that the remaining average min-entropy of the honest party's input, when conditioned on the ideal leakage, is the same as when conditioned on the real leakage *and* the adversary's real view (where the real view is computed with respect to the same input and randomness given as inputs to the leakage function). The problem with this approach in the computational setting is that typically, the remaining entropy of the input is zero when conditioned on the view. We consider a modified definition that takes a similar approach to entropic public key encryptions [HL11]. Namely, we require that there exists a fake game in which the view of the adversary in this game, denoted $\mathbf{View}^{\mathrm{fk}}_{\pi,\mathcal{A}}(x_0, x_1)$ is indistinguishable from its real view $\mathbf{View}^{\mathrm{rl}}_{\pi,\mathcal{A}}(x_0, x_1)$, even when conditioned on $\mathcal{A}$'s leakage. Moreover, $\mathbf{View}^{\mathrm{fk}}_{\pi,\mathcal{A}}(x_0, x_1)$ is generated with respect to the same input and randomness given as inputs to the leakage function submitted by $\mathcal{A}$. Finally, it is also required that x_{1-i^*} has high min-entropy conditioned on the fake view.

In addition, we require that the ideal and real views are computationally indistinguishable in the presence of leakage. This ensures that even if the adversary learns more information by observing the transcript of the protocol together with the leakage, an ideal adversary (simulator) can still simulate its view. We introduce two flavors of computational and statistical notions of security, starting with the computational case.

Definition 3.3 ((Computational Entropic Security against Global Leakage) *Let $f = (f_0, f_1)$ be a two-party functionality and π be a two-party protocol computing f. We say that π $(\lambda_0^{\mathrm{rl}}, \lambda_1^{\mathrm{rl}})$-*securely computes *$f$ in the presence of global leakage if for every* PPT *semi-honest $(\lambda_0^{\mathrm{rl}}, \lambda_1^{\mathrm{rl}})$-*global admissible *adversary $\mathcal{A}$ (cf. Definition 3.2), there exists a* PPT *ideal $(\lambda_0^{\mathrm{sm}}, \lambda_1^{\mathrm{sm}})$-*admissible *adversary $\mathcal{S}$ such that for all sufficiently large n's, every* PPT *k-source input sampler machine $M(1^n, \cdot)$ and every auxiliary input $z \in \{0,1\}^*$ it holds that,*

1. Equality with respect to remaining min-entropy of the honest party's input.

$$\tilde{H}_\infty\left(x_{1-i^*} \mid h^{\mathrm{sm}}_{1-i^*}(x_{1-i^*})\right) = \tilde{H}_\infty\left(x_{1-i^*} \mid h^{\mathrm{rl}}_{1-i^*}(S_{1-i^*}), \mathbf{View}^{\mathrm{fk}}_{\pi,\mathcal{A}}(x_0,x_1)\right)$$

for $h^{\mathrm{rl}}_{1-i^*}, h^{\mathrm{sm}}_{1-i^*}$ the leakage functions submitted respectively, in the real and ideal executions, $(x_0,x_1) \leftarrow M(1^n,z)$ and $\mathbf{View}^{\mathrm{fk}}_{\pi,\mathcal{A}}(x_0,x_1) \approx_c \mathbf{View}^{\mathrm{rl}}_{\pi,\mathcal{A}}(x_0,x_1)$, so that

(a) $\mathbf{View}^{\mathrm{fk}}_{\pi,\mathcal{A}}$ *is generated by a* PPT *simulator $\mathcal{S}_{\mathrm{fk}}$ with respect to the same input and randomness given as inputs to $h^{\mathrm{rl}}_{1-i^*}$.*

(b) $\tilde{H}_\infty\left(x_{1-i^*} \mid \mathbf{View}^{\mathrm{fk}}_{\pi,\mathcal{A}}(x_0,x_1)\right) = k$.[4]

2. Computational indistinguishability.

$$\{\mathbf{Gideal}_{f,\mathcal{S},M}(n,z)\}_{n,z} \approx_c \{\mathbf{Greal}_{\pi,\mathcal{A},M}(n,z)\}_{n,z}.$$

Observe that if only the input of the honest party leaks then any functionality, that is securely realized with respect to standard ideal/real framework (i.e., without leakage attacks), can be realized under Definition 3.3 as well. This is because one can use the same simulator with a minor change so that now it additionally hands the real adversary the leakage obtained from the trusted party.

We continue with the definition for statistical security which is similar to Definition 3.3 except for an unbounded adversary (yet, still submitting efficiently computable leakage functions). In addition, we require statistically closeness instead of computational closeness.

Definition 3.4 (Statistical Entropic Security against Global Leakage) *Let $f = (f_0, f_1)$ be a two-party functionality and π be a two-party protocol computing f. We say that π $(\lambda^{\mathrm{rl}}_0, \lambda^{\mathrm{rl}}_1)$-*statistically computes *f in the presence of global leakage if for every semi-honest $(\lambda^{\mathrm{rl}}_0, \lambda^{\mathrm{rl}}_1)$-*global admissible *adversary $\mathcal{A}$ (cf. Definition 3.2), there exists an ideal $(\lambda^{\mathrm{sm}}_0, \lambda^{\mathrm{sm}}_1)$-*admissible *adversary $\mathcal{S}$ such that for all sufficiently large n's, every* PPT *k-source input sampler machine $M(1^n,\cdot)$ and every auxiliary input $z \in \{0,1\}^*$ it holds that,*

1. Equality with respect to remaining min-entropy of the honest party's input.

$$\tilde{H}_\infty\left(x_{1-i^*} \mid h^{\mathrm{sm}}_{1-i^*}(x_{1-i^*})\right) = \tilde{H}_\infty\left(x_{1-i^*} \mid h^{\mathrm{rl}}_{1-i^*}(S_{1-i^*}), \mathbf{View}^{\mathrm{fk}}_{\pi,\mathcal{A}}(x_0,x_1)\right)$$

for $h^{\mathrm{rl}}_{1-i^*}, h^{\mathrm{sm}}_{1-i^*}$ the leakage functions submitted respectively, in the real and ideal executions, $(x_0,x_1) \leftarrow M(1^n,z)$ and $\mathbf{View}^{\mathrm{fk}}_{\pi,\mathcal{A}}(x_0,x_1) \approx_s \mathbf{View}^{\mathrm{rl}}_{\pi,\mathcal{A}}(x_0,x_1)$, so that

(a) $\mathbf{View}^{\mathrm{fk}}_{\pi,\mathcal{A}}$ *is generated by a* PPT *simulator $\mathcal{S}_{\mathrm{fk}}$ with respect to the same input and randomness given as inputs to $h^{\mathrm{rl}}_{1-i^*}$.*

(b) $\tilde{H}_\infty\left(x_{1-i^*} \mid \mathbf{View}^{\mathrm{fk}}_{\pi,\mathcal{A}}(x_0,x_1)\right) = k$.

2. Statistical indistinguishability.

$$\{\mathbf{Gideal}_{f,\mathcal{S},M}(n,z)\}_{n,z} \approx_s \{\mathbf{Greal}_{\pi,\mathcal{A},M}(n,z)\}_{n,z}.$$

Note that for the special case that the protocol obtains statistical security (without leakage), the fake view can be replaced with the real view.

[4] A potential relaxation of this definition requires that the remaining average min-entropy of x_{1-i^*} greater equals $k-\delta$ for some overhead entropy drop δ that is negligible in the security parameter.

Problems and solutions with global leakage. The global leakage definitions make very strong demands and it is not clear that they can even be achieved without a trusted setup. We discuss some difficulties with a security proof in this model: A standard way to implement the ideal adversary (or simulator) is to let it run a copy of the real adversary in its head and try to emulate its view of the attack, which in particular must include the output of its leakage function h. A natural way to emulate the output of h is for the simulator to design a new leakage function with randomness for the honest party encoded in it. The idea would be that when submitted and computed by the trusted party, this leakage function would emulate the execution of π and leak the adversary's view of an execution with the specified inputs and randomness. This could be used to generate a valid response to the real adversary. The problem with this approach is twofold. First, in the computational setting the remaining min-entropy of the parties' inputs conditioned on the protocol transcript is typically zero. Therefore whenever the leakage rate is strictly less than 1, the simulator cannot submit a function that emulates the transcript of the protocol. This is because this would imply that the simulator asks for more information than it is entitled to. Another problem is that the simulator knows the leakage *as well as* the randomness used to generate it, and this may enable it to learn the honest party's input entirely even if it submits an admissible leakage function.

One can imagine several approaches to getting around the difficulties raised above. One way is by assuming that the secret randomness of each party is split into several components and the leakage attacks are restricted so that each component leaks separately [AGH, HL11, DLWW11]. This setting may capture a more realistic scenario in which the components are stored in different locations in the memory and accessed separately by the device. The assumption then is that the adversary can only obtain leakage from each source separately. Formally, denote the two sources by the bit strings R_1, R_2. Then the adversary submits a pair of leakage functions f_1, f_2 and receives back $f_1(x, R_1), f_2(x, R_2)$. The idea is that the adversary cannot jointly compute R given leakage from R_1 and R_2 and thus cannot compute the messages from the protocol transcript computed by the honest party. This model can be seen as a special case of the only-computation-leaks framework defined by Micali and Reyzin [MR04] since the adversary is only allowed to leak from the randomness source that is in use and cannot leak any computation made by the honest party throughout the protocol execution.

Alternatively, one can assume that the randomness used by a party at each round is determined only at the beginning of that round and therefore can only be leaked after this point. This approach was adopted in [SV03]. A final option, which is the one we adopt here, is to assume what we call **local leakage**, which is described in the following section.

3.2 Security in the Presence of Local Leakage and Semi-Honest Adversaries

In this section we consider the setting of local leakage attacks. Here the adversary is allowed to submit multiple leakage queries, each is applied on the *local* memory parts used by the honest party at the time when the leakage takes place. The leakage function may depend on all public information and prior leakage. This modeling is inspired by the only-computational-leaks axiom made in [MR04] where only the memory parts that are in use may leak.

We modify the ideal world modeling, formalized as in Definition 3.3, so that the simulator submits multiple leakage functions $h_1, \ldots, h_t$. We require that the overall leakage obtained by these functions is bounded by some global leakage rate λ_{1-i^*}, so that

$$\tilde{H}_\infty(x_{1-i^*} |\, h_1(x_{1-i^*}), \ldots, h_t(x_{1-i^*})) \geq (1 - \lambda_{1-i^*}) \cdot H_\infty(x_{1-i^*})$$

for the honest party P_{1-i^*}. Similarly, in the real model we require that the total amount of leakage is bounded by some specified global leakage rate. We formalize the overall leakage obtained by the adversary as follows.

Definition 3.5 (Accumulated Leakage Rate) *For every $\tau = \tau(n)$ and a set of leakage functions $H = \{h_i^k\}_{k\in[\tau]}$, we say that the* accumulated leakage rate *of H is λ if*

$$\tilde{H}_\infty\left(x_i \,|\{h_i^k(S_i^k)\}_{k\in[\tau]}\right) \geq (1-\lambda)\cdot H_\infty(x_i)$$

where S_i^k is the memory parts used by the honest party at the time h_i^k is submitted.

We continue with a definition of a local admissible adversary.

Definition 3.6 ((λ_0,λ_1)-local admissible adversary) *An adversary $\mathcal{A}$ is called* (λ_0,λ_1)-local admissible *adversary if for all $i^* \in \{0,1\}$, $\mathcal{A}(1^n,z,x_{i^*})$ submits* PPT *leakage functions $H = \{h_i^k\}_{k\in[\tau]}$ to the leakage oracle such that the accumulated leakage rate of H is upper bounded by $\lambda_{1-i^*} = \lambda_{1-i^*}(n)$, for P_{i^*} the corrupted party and n the security parameter.*

We are now ready to present the complete definition of secure computation with respect to local leakage attacks. We only provide the computational variant; the statistical variant follows similarly. Denote by $\mathbf{Lideal}_{f,\mathcal{S},M}(n,z)$ the output pair of the honest party P_{1-i^*} and adversary $\mathcal{S}$ in the above ideal execution, and by $\mathbf{Lreal}_{\pi,\mathcal{A},M}(n,z)$ the output pair of the honest party and an adversary $\mathcal{A}$ upon completing the execution of π in the local leakage setting.

Definition 3.7 (Computational Entropic Security against Local Leakage) *Let $f = (f_0,f_1)$ be a two-party functionality and π be a two-party protocol computing f. We say that π $(\lambda_0^{\mathrm{rl}},\lambda_1^{\mathrm{rl}})$-*securely computes *$f$ in the presence of local leakage if for every* PPT *semi-honest $(\lambda_0^{\mathrm{rl}},\lambda_1^{\mathrm{rl}})$-*local admissible *adversary $\mathcal{A}$ (cf. Definition 3.6), there exists a* PPT *ideal $(\lambda_0^{\mathrm{sm}},\lambda_1^{\mathrm{sm}})$-*admissible *adversary $\mathcal{S}$ such that for all sufficiently large n's, every* PPT *k-source input sampler machine $M(1^n,\cdot)$ and every auxiliary input $z \in \{0,1\}^*$ it holds that,*

1. Equality with respect to remaining min-entropy of the honest party's input.

$$\tilde{H}_\infty\left(x_{1-i^*} \mid \left\{h_{1-i^*}^{\mathrm{sm}_k}(x_{1-i^*})\right\}_{k\in[\tau]}\right) = \tilde{H}_\infty\left(x_{1-i^*} \mid \{h_{1-i^*}^{\mathrm{rl}_k}(S_{1-i^*}^k)\}_{k\in[\tau]}, \mathbf{View}_{\pi,\mathcal{A}}^{\mathrm{fk}}(x_0,x_1)\right)$$

   for $\{h_{1-i^*}^{\mathrm{rl}_k}\}_{k\in[\tau]}$ the leakage functions submitted in the real execution, $S_{1-i^*}^k$ the memory parts used by the honest party at the time $h_{1-i^*}^k$ is submitted, $\{h_{1-i^*}^{\mathrm{sm}_k}\}_{k\in[\tau]}$ the leakage functions submitted in the ideal execution, $(x_0,x_1) \leftarrow M(1^n,z)$ and $\mathbf{View}_{\pi,\mathcal{A}}^{\mathrm{fk}}(x_0,x_1) \approx_c \mathbf{View}_{\pi,\mathcal{A}}^{\mathrm{rl}}(x_0,x_1)$, so that

   (a) $\mathbf{View}_{\pi,\mathcal{A}}^{\mathrm{fk}}$ *is generated by a* PPT *simulator $\mathcal{S}_{\mathrm{fk}}$ with respect to the same input and randomness given as inputs to $h_{1-i^*}^{\mathrm{rl}}$.*

   (b) $\tilde{H}_\infty\left(x_{1-i^*} \mid \mathbf{View}_{\pi,\mathcal{A}}^{\mathrm{fk}}(x_0,x_1)\right) \geq k$.

2. Computational indistinguishability.

$$\{\mathbf{Lideal}_{f,\mathcal{S},M}(n,z)\}_{n,z} \approx_c \{\mathbf{Lreal}_{\pi,\mathcal{A},M}(n,z)\}_{n,z}.$$

3.3 The Feasibility of Computational Secure Computation with Leakage

In this section we study the feasibility of the security definition presented in Section 3 in the global setting and explore some of the conditions under which this definition is not feasible. Specifically, we show that for a broad class of hardness assumptions definition 3.3 is infeasible.

Recall first that when reducing the security of a protocol π to the hardness of some problem P, one designs an algorithm $\mathcal{A}_P$ that (given an instance x of P) tries to use π to break P. Since the randomness used to generate x is available to the leakage function as part of the content of the secret memory of the honest party, intractability of P must hold even then. This observation shows that it is problematic to use in the global setting for a large set of assumptions such as DDH, DCR or QR as internal computations may be leaking as well.

For the formal proof we consider the following definitions of hardness and enhanced hardness. Let S_P denote a sampler for a problem P such that given a security parameter 1^n outputs an instance x for a problem P. We formalize the notion of hardness by requiring the existence of a hard to compute function f_P.

Definition 3.8 (Hardness) *We say that a problem P is* hard *if there exists a function f_P such that for every* PPT *algorithm A there exists a negligible function* negl *such that,*

$$\Pr[A(S_P(1^n)) = f_P(S_P(1^n))] \leq \mathsf{negl}(n)$$

where the probability is taken over the randomness used by sampler S_P.

Definition 3.9 (Enhanced hardness) *We say that a problem P is* enhanced hard *if there exists a function f_P such that for every* PPT *algorithm A there exists a negligible function* negl *such that,*

$$\Pr[A(S_P(1^n), R_n) = f_P(S_P(1^n))] \leq \mathsf{negl}(n)$$

where the probability is taken over the choice of R_n the randomness used by sampler S_P.

We say that a problem P is non-enhanced if it is hard but not enhanced hard with respect to some function f_P. For distinguishing problems (e.g., DDH) we modify the definition and consider A's advantage over $\frac{1}{2}$ in guessing $f_P(S_P(1^n))$. It can be easily seen that one-way functions are non-enhanced hard primitives whereas, hash functions [KV09] are indeed enhanced-hard. In particular, since one-way functions can be designed from hash functions this means that a non-enhanced hard primitive can be constructed from an enhanced hard primitive. The other direction is not known and remains open. Note that if the other direction does not hold, it may set as an evident that hash functions cannot be built from one-way functions.

We continue with a theorem showing that for leakage rates greater than $\frac{|f_P(\cdot)|}{|R_n|}$ the hardness of the problem relied on must be enhanced. We consider this lower bound for the case that R_n is the entire randomness used by the protocol. This implies that the output length of $f_P(\cdot)$ is bounded by the size of R_n which is typically the case for most hardness assumptions. For executions in which the protocol uses additional randomness this lower bound is further decreased since the overall randomness is greater than R_n.

Theorem 3.10 *Let $f = (f_0, f_1)$ be a two-party functionality and π be a two-party protocol securely computing f with leakage rate $\lambda \geq \frac{|f_P(\cdot)|}{|R_n|}$ for some $i^* \in \{0,1\}$ (in the sense of Definition 3.3), assuming hardness of problem P. Then P is enhanced hard (cf. Definition 3.9).*

Proof: Let f and π be as above, and let P be a problem such that the security of π relies on the hardness of P. Then consider the following PPT adversary $\mathcal{A}$ and a leakage function h designed by $\mathcal{A}$. Given the input and random tape of P_{1-i^*}, function h emulates the execution of π using the input and random tape of P_{i^*} by computing the transcript of messages exchanged between P_{i^*} and P_{1-i^*}. Let x be an instance of P computed within this transcript and let R_n be the randomness used to produce x. Then h tries to compute $f_P(x)$ using R_n and outputs its guess for this value. Clearly, $\mathcal{A}$ succeeds in computing $f_P(x)$ with a non-negligible probability only for the case when P is not enhanced hard. ■

In Section 5 we show how to overcome this difficulty for specific hardness assumptions under the *only computational leaks assumption*.

4 Modular Composition

Composition theorems constitute an extremely useful and important tool for simplifying proofs of security. They enable to design modular protocols such that the security of each sub-protocol can be proven separately. We begin with a description of the standard f-hybrid model, where a sub-protocol implementing the functionality f is replaced with ideal calls to a trusted party that computes this functionality. Applying compositions theorems, it is possible to prove that the original protocol, invoking a sub-protocol for f, is *as secure as* the new protocol that utilizes such ideal calls. Former composition theorems (e.g., [Can00]) are proven with respect to a security definition where a quantification over all possible inputs is made. This implies that in order to obtain a contradiction, it should be shown that there exists a single pair of inputs for which simulation is not achieved. This, however, does not apply when considering security with respect to input distributions. Since now the quantification is over all possible k-source input samplers (for a particular choice of k) such that the randomness for this sampling process is not fixed ahead of time, rather it is picked at the onset of the experiment. This implies that existing composition proofs do not immediately hold, since now one has to show the existence of a good input sampler that enables to break the security of the underlying composed protocol (good in the sense of Definition 3.1). For the sake of completeness, in section 4.1 we reexamine and provide a composition theorem for the modeling of sampling the inputs according to a prefixed distribution (i.e., ensuring some min entropy for the inputs), in the standard setting. In Section 4.2 we describe the hybrid model for the global leakage setting and present a composition theorem for this setting (cf. Theorem 4.3, Section 4.2). Finally, in Section 4.3 we show how to extend the composition theorems from prior sections into a theorem in the local leakage setting. We believe that these theorems are of great importance for abstracting the security proofs in the leakage setting.

4.1 Modular Composition with Input Distributions

We begin with the detailed description of the hybrid model.

The f-hybrid model. Traditionally, the *hybrid model* enables to prove the security of a protocol π, that uses a sub-protocol computing a functionality f, in an abstracting setting where the the two parties directly interact with each other (as in the real model) as well as have access to a trusted implementation of f (as in the ideal model). Namely, in an execution of π the parties issue ideal calls to a trusted party for computing f instead of invoking a sub-protocol for f. These ideal calls are just instructions to send an input to the trusted party, which, upon receiving the inputs from the parties, computes f and sends each party its corresponding output. After receiving these outputs back from the trusted party, the protocol π continues with standard messages sent amongst the parties. We note that this description corresponds to a sequential composition, i.e., the parties do not send messages in π between the time that they send input to the trusted party and the time that they receive back output. The trusted party may be used a number of times throughout the execution of π. However, each invocation of the functionality f is independent (i.e., the trusted party does not maintain any state between these calls). We call the regular messages of π that are sent amongst the parties standard messages and the messages that are sent between parties and the trusted party ideal messages.

Let f and π be as above, and let ρ be a protocol. Consider the real protocol π^ρ that is defined as follows. All standard messages of π are unchanged. When a party P_i is instructed to send an ideal message α_i to the trusted party, it begins a real execution of ρ with input α_i instead. When this execution of ρ concludes with output β_i, party P_i continues with π as if β_i was the output received by the trusted party (i.e. as if it were running in the f-hybrid model). Then, the composition theorem of [Can00] states that if ρ securely computes f with security against semi-honest adversaries, then the output distribution of a protocol π in a

hybrid execution with f is computationally indistinguishable from the output distribution of the real protocol π^ρ. Thus, it suffices to analyze the security of π when using ideal calls to f; security of the real protocol π^ρ is derived via this composition theorem. *To prove composition for our setting we first state a security definition. For simplicity, we keep this discussion informal and only specify that the real and ideal modelings are as described for security definition 3.3, except that here the adversary does not submit a leakage function at the onset of the execution rather it follows a semi-honest arbitrary strategy (for further discussion regarding the semi-honest setting see [Gol04]).* We denote these ideal and real executions of π on auxiliary input z given to adversary $\mathcal{A}$ and a security parameter n, by $\mathbf{Ideal}_{f,\mathcal{A}^{\mathrm{sm}},M}(n,z)$ and $\mathbf{Real}_{\pi,\mathcal{A},M}(n,z)$, respectively. Then, security is defined as follows.

Definition 4.1 ((Computational) Semi-honest secure computation on input distributions) *Let $f=(f_0,f_1)$ be a two-party functionality and π be a two-party protocol computing f. We say that π,* securely computes *f in the presence of semi-honest adversaries if for every* PPT *semi-honest adversary $\mathcal{A}$, there exists a* PPT *ideal adversary $\mathcal{A}^{\mathrm{sm}}$ such that for all sufficiently large n's, every* PPT *k-source input sampler machine $M(\cdot,\cdot)$ and every auxiliary input $z\in\{0,1\}^*$ it holds that,*

$$\{\mathbf{Ideal}_{f,\mathcal{A}^{\mathrm{sm}},M}(n,z)\}_{n,z} \approx_c \{\mathbf{Real}_{\pi,\mathcal{A},M}(n,z)\}_{n,z}.$$

Before stating our composition theorem, we wish to elaborate about a technicality that arises in this setting when considering a protocol composed out of π and ρ. Recalling that the inputs for each protocol are sampled from some min-entropy source, it must hold that this is also the case with respect to sub-protocol ρ, invoked by π^ρ. Namely, it must be that the inputs entered by the parties for the execution of ρ within the execution of π^ρ maintain the k-source property given in Definition 3.1, for some k that may be different than the k_π specified for the input used in π. This in particular implies that π (until the point where it calls f which is identical to π^ρ), induces a k-source sampling machine for ρ, since the inputs for ρ are computed based on the parties' view so far and (perhaps) additional randomness. We will use this observation in our proof for the composition theorem below. For simplicity we consider first the case where π makes only a single ideal call to f. Then the composition theorem is formalized as follows.

Theorem 4.2 (Semi-honest composition on input distributions) *Let π be a two-party protocol in the f-hybrid model where no more than one ideal evaluation call is made, let ρ be a two-party protocol securely computes f (in the sense of definition 4.1), and let π^ρ be the composed protocol described above. Then, for any* PPT *semi-honest adversary $\mathcal{A}$, there exists a* PPT *adversary $\mathcal{A}^{\mathrm{Hyb}}$ in the f-hybrid model, such that*

$$\{\mathbf{Hybrid}^f_{\pi,\mathcal{A}^{\mathrm{Hyb}},M}(n,z)\}_{n,z} \approx_c \{\mathbf{Real}_{\pi^\rho,\mathcal{A},M}(n,z)\}_{n,z}.$$

Proof: Assume by contradiction that there exists a PPT semi-honest adversary $\mathcal{A}$ such that for any PPT adversary $\mathcal{A}^{\mathrm{Hyb}}$ in the f-hybrid model there exist a PPT k-source sampling machine M, an auxiliary input z, a PPT distinguisher D and a polynomial poly such that

$$\Pr[D(\mathbf{Hybrid}^f_{\pi,\mathcal{A}^{\mathrm{Hyb}},M}(n,z))=1]-\Pr[D(\mathbf{Real}_{\pi^\rho,\mathcal{A},M}(n,z))=1]\geq 1/\mathsf{poly}(n) \qquad (1)$$

for infinitely many n's. Then, we construct a PPT semi-honest adversary $\mathcal{A}_\rho$ such that for any ideal adversary $\mathcal{A}^{\mathrm{sm}}_\rho$ there exists a PPT k_ρ-source sampling machine M_ρ, an auxiliary input z_ρ, a distinguisher D_ρ and a polynomial poly' such that

$$\Pr[D_\rho(\mathbf{Gideal}_{f,\mathcal{A}^{\mathrm{sm}}_\rho,M_\rho}(n,z))=1]-\Pr[D_\rho(\mathbf{Greal}_{\rho,\mathcal{A}_\rho,M_\rho}(n,z))=1]\geq 1/\mathsf{poly}'(n)$$

for infinitely many n's. We continue with the descriptions of z_ρ, M_ρ and adversary $\mathcal{A}_\rho$ attacking protocol ρ. For that, we introduce some notations first. Let Π be a two-party protocol. Then, an internal state of party

P_i at some round l of the execution of P_i consists of its input, random tape, auxiliary input and incoming messages from rounds $1, \ldots, l-1$. The next message function of a protocol Π for party P_i is a PPT function that takes its internal state for some round $l-1$, and outputs its next outgoing message for round l, according to the specification of Π. Without loss of generality, we assume that all the randomness used by the parties is generated at the onset of the protocol. Therefore a party's random tape will have two portions in any given round l. The first portion what we call as *used randomness* is the randomness used until round l for the computation. The second part of the random tape is referred as *unused randomness*. Therefore a party's internal state also includes the unused randomness.

The description of z_ρ. We begin with a description of the auxiliary input z_ρ given to $\mathcal{A}_\rho$ and M_ρ. The auxiliary input z_ρ encodes the internal state of adversary $\mathcal{A}$ at round l_ρ excluding the unused (until round l_ρ) randomness in the random tape of $\mathcal{A}$. This means that z_ρ encodes a run of $\mathcal{A}$ on an input, sampled according to M, auxiliary input z and a random string of appropriate length used until round l_ρ.

The description of M_ρ. We now turn to describing the input sampler machine that picks the inputs for functionality f. Denote by l_ρ the round where protocol π^ρ calls ρ. Then, machine M_ρ is defined by the next message function of protocol π^ρ for round l_ρ. Specifically, the input for M_ρ is the auxiliary input z_ρ and the the unused (up and till round l_ρ) randomness in the random tape of $\mathcal{A}$. M_ρ may use randomness taken from the unused part of $\mathcal{A}$'s random tape. In essence, the input to M_ρ is the internal state of adversary $\mathcal{A}$ at round l_ρ of π^ρ. Note that this sampler machine is a k-source sampler by the fact that the next message function produces the input used by the adversary for ρ. We note that z_ρ is fixed before M_ρ is invoked, implying that we fix the input of $\mathcal{A}$, its auxiliary input and randomness used until round l_ρ. This is important since we would like to assume that the auxiliary input z_ρ encodes this information, which can only happen if this information is fixed *before* we fix z_ρ.

The description of $\mathcal{A}_\rho$. We continue with the description of adversary $\mathcal{A}_\rho$. Loosely speaking, $\mathcal{A}_\rho$ emulates a run of $\mathcal{A}$ starting at the internal state described in z_ρ.

1. Upon receiving an input $x_{i^*}^f$ for ρ (sampled according to a sampler M_ρ defined above), and an auxiliary input z_ρ (distributed as specified above), $\mathcal{A}_\rho$ invokes $\mathcal{A}$ on the internal state encoded within z_ρ. Note that $x_{i^*}^f$ is consistent with z_ρ since both M_ρ and z_ρ correspond to the *same* internal state.
2. Whenever $\mathcal{A}$ reaches the point in the execution of π where it calls ρ, $\mathcal{A}_\rho$ forwards the messages between $\mathcal{A}$ and the external party playing the role of the honest party for ρ.
3. Upon concluding the execution of ρ, $\mathcal{A}_\rho$ outputs its internal state.

Note that $\mathcal{A}_\rho$ is a PPT adversary for ρ. Therefore, by the assumption made in Theorem 4.2, there exists an ideal adversary $\mathcal{A}_\rho^{\mathrm{sm}}$ for $\mathcal{A}_\rho$ such that

$$\{\mathbf{Ideal}_{f,\mathcal{A}_\rho^{\mathrm{sm}},M_\rho}(n,z)\}_{n,z} \approx_c \{\mathbf{Real}_{\rho,\mathcal{A}_\rho,M_\rho}(n,z)\}_{n,z}.$$

Next, consider an adversary $\mathcal{A}'$ who follows the same strategy of $\mathcal{A}$ except that upon calling ρ, $\mathcal{A}'$ invokes adversary $\mathcal{A}_\rho$. Then, after $\mathcal{A}_\rho$ returns its internal state, $\mathcal{A}'$ invokes $\mathcal{A}$ on this state and concludes the execution of π^ρ. It is immediate that the output distribution produced by $\mathcal{A}$ and $\mathcal{A}'$ is identical. Finally, consider a hybrid adversary $\mathcal{A}'^{\mathrm{Hyb}}$ for $\mathcal{A}'$ (attacking π), which invokes $\mathcal{A}_\rho$ in the same manner as $\mathcal{A}'$ did, except that upon reaching the point when π calls ρ, $\mathcal{A}'^{\mathrm{Hyb}}$ invokes ideal adversary $\mathcal{A}_\rho^{\mathrm{sm}}$ on auxiliary input z_ρ instead of invoking $\mathcal{A}_\rho$. When $\mathcal{A}_\rho^{\mathrm{sm}}$ concludes the execution, $\mathcal{A}'^{\mathrm{Hyb}}$ continues by running $\mathcal{A}$ from the

internal state output by $\mathcal{A}_\rho^{\mathrm{sm}}$. Then by the assumption made above in Equation (1), it must hold that the views produced by $\mathcal{A}'$ and $\mathcal{A}'^{\mathrm{Hyb}}$ are not computationally indistinguishable. Namely, there exist a PPT k-source sampling machine M, an auxiliary input z, a PPT distinguisher D and a polynomial poly such that

$$\Pr[D(\mathbf{Hybrid}^{f}_{\pi,\mathcal{A}'^{\mathrm{Hyb}},M}(n,z))=1]-\Pr[D(\mathbf{Real}_{\pi^\rho,\mathcal{A}',M}(n,z))=1]\geq 1/\mathsf{poly}(n) \tag{2}$$

for infinitely many n's.

On the other hand, this contradicts the security of ρ since the only difference between the real execution of $\mathcal{A}'$ and the hybrid execution of $\mathcal{A}'^{\mathrm{Hyb}}$ is that $\mathcal{A}'$ runs $\mathcal{A}_\rho$, whereas $\mathcal{A}'^{\mathrm{Hyb}}$ runs $\mathcal{A}_\rho^{\mathrm{sm}}$ instead. Specifically if these views are distinguishable, then it means that $\mathcal{A}_\rho^{\mathrm{sm}}$ does not simulate $\mathcal{A}_\rho$ as it should, in contradiction to the assumption made in the theorem. More formally, if Equation (2) does not hold, such that the probabilities are taken on the randomness of the adversary and the honest party. Then it must be that the views are distinguishable. Now, since up until the point where ρ is invoked, the views are identical it must be that D distinguishes due to the execution of sub-protocol ρ. Therefore, it is possible to fix the prefix of $\mathcal{A}$'s view up until the round that ρ begins as part of the auxiliary input given to adversary $\mathcal{A}_\rho$ (which includes fixing the input, random tape and auxiliary input of $\mathcal{A}$), and then invoke $\mathcal{A}_\rho$ on this auxiliary input. ■

Handing multiple ideals calls. The theorem above generalizes to the case where π calls ρ multiple times using a hybrid argument. Namely, we first design adversary $\mathcal{A}_{\rho_i}$ so that $\mathcal{A}_{\rho_i}$ behaves like $\mathcal{A}_\rho$ for the ith call to ρ. The rest of the proof follows similarly where adversary $\mathcal{A}'^{\mathrm{Hyb}}$ invokes the ideal adversary $\mathcal{A}_\rho^{\mathrm{sm}}$, i times. See also [Can00] for more details.

4.2 Modular Composition in the Global Leakage Model

In this section we provide a composition theorem for the global leakage setting; for simplicity we consider first the case where π calls functionality f only once. We continue with the formalization of the hybrid model in the global leakage setting.

The f-hybrid model in the global leakage setting. Recall that in this setting the adversary is given access (via the leakage function) to the entire secret state of the honest party that is going to be used through the execution of protocol π. This, in particular, means that the adversary is also given (a leakage) access to the input used for the computation of the functionality f in the ideal execution. Alteratively, the reader may think of a leakage function that is designed to emulate the entire execution between the honest party and the adversary within π until the point where π issues a call to f on input $x^f_{1-i^*}$, from which the leakage function leaks. Therefore in our formalization, when designing a protocol π in the hybrid setting with leakage rates (λ_0,λ_1), the adversary still submits a single leakage function at the onset of π and the parties carry out the ideal calls as follows. For f,ρ and π as above, consider the execution of π in the f-hybrid global leakage setting so that the parties interact with each other for executing the instructions of π as specified in the real execution **Greal** for the global setting (cf. Definitions 3.3, 3.4). Then, when issuing an ideal call to f the parties send their inputs to the trusted party that computes f and returns their corresponding outputs, such that the ideal adversary submits an empty leakage function. This models the fact that the adversary has already gained leakage from the honest party's input $x^f_{1-i^*}$ to f. Similarly, when running protocol π^ρ, the adversary may obtain leakage on the secret state used for running sub-protocol ρ, denoted by $S^\rho_{1-i^*}$.

Next, when designing a protocol π^ρ with leakage rates (λ_0,λ_1),[5] we show that the views induced by running this protocol are computationally indistinguishable from the views induced by running π as long

[5] We require the same leakage rates in both the hybrid and real settings, since otherwise it will be easy to distinguish the executions. We emphasize that this does not stop different leakage functions with different leakage rate to be applied for the ideal call of f and real execution of protocol ρ (refer to the discussion following Definition 3.3).

as: (1) ρ securely computes f in the sense of Definition 3.3 with leakage parameters $(\tilde{\lambda}_0, \tilde{\lambda}_1)$, and (2) the entropy loss of secret states (S_0^ρ, S_1^ρ) within π^ρ (i.e., the entropy loss from the secret states used for running sub-protocol ρ within π^ρ) are bounded by $(\tilde{\lambda}_0, \tilde{\lambda}_1)$. This necessarily implies that $\lambda_i \leq \tilde{\lambda}_i$ for all $i \in \{0,1\}$, as otherwise one can always design an adversary that submits a leakage function focusing only on $S^\rho_{1-i^*}$, thus leaking too much. In particular, λ is taken as a fraction of a string that is at least as long as the input and randomness lengths for ρ, and most likely longer. Note that in both hybrid and real runs a single leakage function is submitted at the beginning of the execution. Nevertheless, it may not be the same function as it is picked based on the randomness and auxiliary input given to the adversary. Finally, we note that a global admissible adversary in the hybrid setting is defined as a real global admissible adversary (namely, submitting a leakage function and then engaging with the honest party in an execution of π).

Modular composition. We are now ready to present our composition theorem for the global setting. Formally, denote the f-hybrid execution of π on auxiliary input z given to adversary $\mathcal{A}^{\mathrm{Hyb}}$ and a security parameter n, by $\mathbf{Ghybrid}^f_{\pi,\mathcal{A}^{\mathrm{Hyb}},M}(n,z)$. For f, ρ and π as above, denote the execution of the composed protocol π^ρ on auxiliary input z given to the adversary $\mathcal{A}$ and a security parameter n, by $\mathbf{Greal}_{\pi^\rho,\mathcal{A},M}(n,z)$. Then our composition theorem for the global leakage settings says that the distribution of $\mathbf{Ghybrid}^f_{\pi,\mathcal{A}^{\mathrm{Hyb}},M}(n,z)$ shoud be computationally indistinguishable from the distribution of $\mathbf{Greal}_{\pi^\rho,\mathcal{A},M}(n,z)$. Moreover, we require that the average min-entropy left on the input of honest party for f remains the same in π as well as in π^ρ. We denote the internal states of honest party on the onset of the execution of π in the hybrid execution and real execution by $S^{\mathrm{Hyb}}_{1-i^*}$ and $S^{\mathrm{rl}}_{1-i^*}$, respectively.

Theorem 4.3 (Semi-honest composition in the global setting) *Let π be a two-party protocol in the f-hybrid model where no more than one ideal evaluation call is made, let ρ be a two-party protocol $(\tilde{\lambda}_0, \tilde{\lambda}_1)$-securely computes f (in the sense of definition 3.3), and let π^ρ be the composed protocol described above. Then, for any* PPT *(λ_0, λ_1)-global admissible adversary $\mathcal{A}$ (cf. Definition 3.2), there exists a* PPT *(λ_0, λ_1)-global admissible adversary $\mathcal{A}^{\mathrm{Hyb}}$ in the f-hybrid model, such that*

$$\{\mathbf{Ghybrid}^f_{\pi,\mathcal{A}^{\mathrm{Hyb}},M}(n,z)\}_{n,z} \approx_c \{\mathbf{Greal}_{\pi^\rho,\mathcal{A},M}(n,z)\}_{n,z}$$

conditioned on,

1. $$\tilde{H}_\infty\left(S^\rho_{1-i^*} \mid h^{\mathrm{rl}}_{1-i^*}(S^{\mathrm{rl}}_{1-i^*})\right) \geq (1-\tilde{\lambda}_{1-i^*}) \cdot H_\infty(S^\rho_{1-i^*})$$

   for $S^\rho_{1-i^*}$ the secret state of the honest party used for running sub-protocol ρ within π^ρ.

2. $$\tilde{H}_\infty\left(x^f_{1-i^*} \mid h^{\mathrm{Hyb}}_{1-i^*}(S^{\mathrm{Hyb}}_{1-i^*})\right) = \tilde{H}_\infty\left(x^f_{1-i^*} \mid (h^{\mathrm{rl}}_{1-i^*}(S^{\mathrm{rl}}_{1-i^*}), \mathbf{View}^{\mathrm{fk}}_{\rho,\mathcal{A}}(x^f_0, x^f_1))\right)$$

   for $h^{\mathrm{Hyb}}_{1-i^*}, h^{\mathrm{rl}}_{1-i^*}$ the leakage functions submitted in the hybrid and real runs, respectively, and $x^f_{1-i^*}$ the honest party's input for f and $\mathbf{View}^{\mathrm{fk}}_{\rho,\mathcal{A}}(x^f_0, x^f_1) \approx_c \mathbf{View}^{\mathrm{rl}}_{\rho,\mathcal{A}}(x^f_0, x^f_1)$, so that

   (a) $\mathbf{View}^{\mathrm{fk}}_{\rho,\mathcal{A}}$ *is generated by a* PPT *simulator $\mathcal{S}^f_{\mathrm{fk}}$ with respect to the same input and randomness that $h^{\mathrm{rl}}_{1-i^*}$ is applied on.*

   (b) $\tilde{H}_\infty\left(x^f_{1-i^*} \mid \mathbf{View}^{\mathrm{fk}}_{\rho,\mathcal{A}}(x_0, x_1)\right) = k_\rho$. [6]

[6] A potential relaxation of this definition requires that the remaining average min-entropy of $x^f_{1-i^*}$ greater equals $k_\rho - \delta$ for some overhead entropy drop δ that is negligible in the security parameter.

Note that we still wish to bound the amount of information learnt in the hybrid setting when compared to the real setting and we therefore enforce the first condition. Recall that we have that π (until the point where it calls f which is identical to π^ρ), induces a k_ρ-source sampling machine for ρ, since the inputs for ρ are computed based on the parties' view so far and (perhaps) additional randomness. We still want π to induce a k_ρ-source sampling machine for ρ even in the presense of global leakage.

We begin by focusing the attention on a difficulty arises when proving this theorem with respect to an adversary attacking π^ρ. Recall that such an adversary obtains leakage from the randomness used for the execution of ρ as well. Specifically, the setting of global leakage enables the adversary to observe leakage from the randomness used for all future computations. This implies that the proof for Theorem 4.2 cannot be directly applied here since the auxiliary input z_ρ cannot be constructed as before. More specifically, recalling that z_ρ is encoded with an internal state of an adversary $\mathcal{A}$ at round l_ρ, this means that z_ρ must be encoded with the leakage obtained *prior* to the execution of π^ρ. Now, when building a reduction to the security of ρ, this means that z_ρ is incorporated with leakage obtained also from a randomness that was not fixed yet (since the randomness is fixed as part of the experiment, after the auxiliary input is fixed). In order to correct that, we consider a leakage function of a specific type that enables to generate z_ρ. Namely, we design an adversary $\mathcal{A}_\rho$ so that it first submits a leakage function that returns the internal view of $\mathcal{A}$ including its randomness, and then invokes $\mathcal{A}$ on this internal state as in the previous proof. Details follow.

Proof: Assume by contradiction that there exists a PPT global (λ_0, λ_1)-admissible adversary $\mathcal{A}$ such that for any PPT global (λ_0, λ_1)-admissible adversary $\mathcal{A}^{\mathrm{Hyb}}$ there exist a PPT k-source sampling machine M, an auxiliary input z, a PPT distinguisher D and a polynomial poly such that

$$\Pr[D(\mathbf{Ghybrid}^{f}_{\pi,\mathcal{A}^{\mathrm{Hyb}},M}(n,z))=1]-\Pr[D(\mathbf{Greal}_{\pi^\rho,\mathcal{A},M}(n,z))=1]\geq 1/\mathsf{poly}(n)$$

for infinitely many n's and assume that the conditions in the theorem are met. Then, construct a PPT global $(\tilde{\lambda}_0, \tilde{\lambda}_1)$-admissible adversary $\mathcal{A}_\rho$ such that for any ideal adversary $\mathcal{A}^{\mathrm{sm}}_\rho$ there exist a PPT k_ρ-source sampling machine M_ρ, an auxiliary input z_ρ, a distinguisher D_ρ and a polynomial poly' such that

$$\Pr[D_\rho(\mathbf{Gideal}_{f,\mathcal{A}^{\mathrm{sm}}_\rho,M_\rho}(n,z))=1]-\Pr[D_\rho(\mathbf{Greal}_{\rho,\mathcal{A}_\rho,M_\rho}(n,z))=1]\geq 1/\mathsf{poly}'(n)$$

for infinitely many n's. We continue with the descriptions of M_ρ, z_ρ and adversary $\mathcal{A}_\rho$ attacking protocol ρ.

The description of z_ρ. Auxiliary input z_ρ includes the inputs of $\mathcal{A}$ and the honest party for π^ρ, their randomness used for running protocol π^ρ up and till round l_ρ, and $\mathcal{A}$'s auxiliary input z.[7] For simplicity we assume that π^ρ calls ρ in its last step. Therefore, the random tape of the parties contain randomness that is used by ρ and the rest of the part of code in π^ρ.

The description of M_ρ. The description of M_ρ is similar to the description of M_ρ in the proof of Theorem 4.2. Namely, machine M_ρ is defined by the next message function of protocol π^ρ for round l_ρ. Now the input for M_ρ is the auxiliary input z_ρ and the the unused (up and till round l_ρ) randomness in the random tape of $\mathcal{A}$. Given the above input, M_ρ by itself can generate the leakage obtained by $\mathcal{A}$ at the onset of the computation and the messages communicated by the parties up and till l_ρ. Therefore in essence, the input to M_ρ is the internal state of adversary $\mathcal{A}$ and the honest party at round l_ρ of π^ρ. As in the proof of Theorem 4.2, M_ρ may be using randomness taken from the unused part of $\mathcal{A}$ and the honest party's random tape in order to generate the input for ρ.

[7] The reader may wonder whether it may be that the honest party's input for ρ is embedded within the honest party's input for π^ρ, thus no meaningful notion of security can be achieved when seeing z_ρ. Nevertheless, we recall that the input for ρ must be sampled by a k_ρ-source sampler that guarantees some uncertainty within the input of the honest party's to ρ, even in the presence of z_ρ. Thus, the fact that $\mathcal{A}_\rho$ observes z_ρ does not imply that $\mathcal{A}_\rho$ learns the input of the honest party to ρ.

The description of $\mathcal{A}_\rho$. We continue with the description of adversary $\mathcal{A}_\rho$.

1. Upon receiving an input $x_{i^*}^f$ for ρ (sampled according to a sampler M_ρ defined above), and an auxiliary input z_ρ (distributed as specified above), $\mathcal{A}_\rho$ invokes $\mathcal{A}$ which outputs a description of a leakage function, recall that the choice of h is made based on $\mathcal{A}$'s input, randomness and auxiliary input.

2. $\mathcal{A}_\rho$ then submits a leakage function h_ρ designed as follows. $\mathcal{A}_\rho$ encodes h_ρ with z_ρ, the code of h and the specification of π^ρ until round l_ρ so that upon getting the input of the honest party and the randomness used for the execution of ρ, h_ρ first emulates the output of h on the input and randomness specified by the honest party's input and randomness for π^ρ until round l_ρ (taken from z_ρ), concatenated with the randomness used by the honest party for ρ.

3. Upon receiving the leakage, $\mathcal{A}_\rho$ invokes $\mathcal{A}$ on this leakage and emulates its internal state until round l_ρ using the input and randomness given in z_ρ. Finally, h_ρ outputs this internal state.

4. $\mathcal{A}_\rho$ invokes $\mathcal{A}$ on the internal state encoded returned by h_ρ. Then, whenever $\mathcal{A}$ reaches the point in the execution of π^ρ where it calls ρ, $\mathcal{A}_\rho$ forwards the messages between $\mathcal{A}$ and the external party playing the role of the honest party for ρ.

5. Upon concluding the execution of ρ, $\mathcal{A}_\rho$ outputs its internal state.

A key point observation here is that $\mathcal{A}_\rho$ is a $(\tilde{\lambda}_0, \tilde{\lambda}_1)$-global admissible adversary for ρ (cf. Definition 3.2) in the sense that the submitted leakage function h_ρ is $(\tilde{\lambda}_0, \tilde{\lambda}_1)$-admissible for ρ (i.e., the entropy loss of the parties' input and randomness for ρ are respectively bounded by these rates). Furthermore, h incorporated into h_ρ is a (λ_0, λ_1)-admissible for π^ρ (i.e., the entropy loss of the parties' input and randomness for π^ρ are respectively bounded by these rates). This is due to the fact that by the assumption above, $\mathcal{A}$ is a (λ_0, λ_1)-global admissible adversary for π^ρ, thus it submits a good leakage function when running π^ρ. In addition, from condition 1 in the theorem, it holds that $S^\rho_{1-i^*}$ is not leaked "too much" in the sense that the entropy loss is bounded by $\tilde{\lambda}_{1-i^*}$. Therefore, $\mathcal{A}^\rho$ is an admissible adversary for ρ. The rest of the proof follows similarly to the proof of Theorem 4.2. Namely, we design an adversary $\mathcal{A}'$ who follows the same strategy of $\mathcal{A}$ except that upon calling ρ, $\mathcal{A}'$ invokes adversary $\mathcal{A}_\rho$. Then, after $\mathcal{A}_\rho$ returns its internal state, $\mathcal{A}'$ invokes $\mathcal{A}$ on this state and concludes the execution of π^ρ. A technicality arises here as both adversaries $\mathcal{A}$ and $\mathcal{A}_\rho$ are expecting to submit a leakage function. This is solved by submitting the leakage function on the behalf of $\mathcal{A}$ and then, whenever $\mathcal{A}_\rho$ is invoked simply emulating the leakage oracle's answer by returning $\mathcal{A}_\rho$ the answer returned originally to $\mathcal{A}$ (since both adversaries submit the same leakage function). It is immediate that the output distribution produced by $\mathcal{A}$ and $\mathcal{A}'$ is identical. Finally, consider a hybrid adversary $\mathcal{A}'^{\mathrm{Hyb}}$ for $\mathcal{A}'$ (attacking π), which invokes $\mathcal{A}_\rho$ in the same manner as $\mathcal{A}'$ did, except that upon reaching the point when π calls ρ, $\mathcal{A}'^{\mathrm{Hyb}}$ invokes ideal adversary $\mathcal{A}_\rho^{\mathrm{sm}}$ on auxiliary input z_ρ instead of invoking $\mathcal{A}_\rho$. When $\mathcal{A}_\rho^{\mathrm{sm}}$ concludes the execution, $\mathcal{A}'^{\mathrm{Hyb}}$ continues by running $\mathcal{A}$ from the internal state output by $\mathcal{A}_\rho^{\mathrm{sm}}$. The rest of the proof follows as above. ■

4.3 Modular Composition in the Local Leakage Model

Proving composition on the local setting is simpler than the global setting as the adversary only sees the state used for the current computation. This simplifies the proof since we do not need to deal with the subtlety from above of enabling the adversary to leak from randomness of all future computation (i.e., the computation of ρ). In particular, the modeling in this setting follows the modeling of the standard setting with the exception that the adversary sees leakage as well. I.e., in the hybrid setting **Lhybrid** the parties submit their inputs generated for the ideal computation of f together with a leakage function as well to be applied on the honest party's input.

This means that the proof of Theorem 4.2 can be applied here as well with the following modification. The auxiliary input given to adversary $\mathcal{A}_\rho$ which is encoded with the internal state of $\mathcal{A}$ until round l_ρ is also encoded with the leakage obtained by $\mathcal{A}$ at the outset of the execution of π^ρ. Then, when $\mathcal{A}_\rho$ is invoked, it is simply submits the leakage function that would have been submitted by $\mathcal{A}$ when reaching round l_ρ. The rest of the proof follows similarly. We begin with a theorem for a single ideal call.

Theorem 4.4 (Semi-honest composition in the local setting) *Let π be a two-party protocol in the f-hybrid model where no more than one ideal evaluation call is made, let ρ be a two-party protocol $(\tilde{\lambda}_0, \tilde{\lambda}_1)$-securely computes f (in the sense of definition 3.7), and let π^ρ be the composed protocol described above. Then, for any* PPT *(λ_0, λ_1)-local admissible adversary $\mathcal{A}$ (cf. Definition 3.2), there exists a* PPT *(λ_0, λ_1)-local admissible adversary $\mathcal{A}^{\mathrm{Hyb}}$ in the f-hybrid model, such that*

$$\{\mathbf{Lhybrid}^f_{\pi,\mathcal{A}^{\mathrm{Hyb}},M}(n,z)\}_{n,z} \approx_c \{\mathbf{Lreal}_{\pi^\rho,\mathcal{A},M}(n,z)\}_{n,z}$$

conditioned on,

1. $$\tilde{H}_\infty\left(S^\rho_{1-i^*} \mid h^{\mathrm{rl}}_{1-i^*}(S^{\mathrm{rl}}_{1-i^*})\right) \geq (1-\tilde{\lambda}_{1-i^*}) \cdot H_\infty(S^\rho_{1-i^*})$$

   for $S^\rho_{1-i^*}$ the secret state of the honest party used for running sub-protocol ρ within π^ρ.

2. $$\tilde{H}_\infty\left(x^f_{1-i^*} \mid \left\{h^{\mathrm{Hyb}_k}_{1-i^*}(S^{\mathrm{Hyb}}_{1-i^*})\right\}_{k\in[\tau]}\right) = \tilde{H}_\infty\left(x^f_{1-i^*} \mid \left(\left\{h^{\mathrm{rl}_k}_{1-i^*}(S^{\mathrm{rl}}_{1-i^*})\right\}_{k\in[\tau]}, \mathbf{View}^{\mathrm{fk}}_{\rho,\mathcal{A}}(x^f_0, x^f_1)\right)\right)$$

   for $\left\{h^{\mathrm{Hyb}_k}_{1-i^*}\right\}_{k\in[\tau]}, \left\{h^{\mathrm{rl}_k}_{1-i^*}\right\}_{k\in[\tau]}$ the leakage functions submitted in the hybrid and real runs, respectively, and $x^f_{1-i^*}$ the honest party's input for f. Rest is same as in Definition 4.3.

Handing multiple ideals calls. Handling multiple ideal calls for this setting is dealt similarly to the standard setting using a hybrid argument. Only that the definition should limit the amount of information obtained by the simulator with respect to each sub-protocol. Namely, for each ideal call replaced by a sub-protocol ρ we require that the remaining entropy of the honest party's input for ρ is the same in both hybrid and real runs.

5 Leakage Resilient Oblivious Transfer

The 1-out-of-2 oblivious transfer (OT) functionality $\mathcal{F}_{\mathrm{OT}}$ [Rab81, EGL85] is defined by $((x_0, x_1), \sigma) \mapsto (\phi, x_\sigma)$ for ϕ the empty string, where the sender Sen has a pair of inputs (x_0, x_1) and the receiver Rec has a bit σ. The aim of the protocol is that the receiver learns x_σ (and x_σ only) without revealing anything about σ to the sender. OT is one of the basic and important building blocks for cryptographic protocols and has been studied extensively in the literature based on various hardness assumptions, providing different notions of security and efficiency; see [NP05, PVW08] for just two examples.

In this section we design a leakage resilient OT with security in the presence of semi-honest adversaries. Our main building block is the OT construction presented by Peikert et al. [PVW08] in the UC setting with security under the DDH assumption (see Assumption 2.4). In order to obtain UC security against malicious adversaries Peikert et al. use a common reference string $\mathsf{crs} = (g_0, h_0, g_1, h_1)$ with $g_0, h_0, g_1, h_1 \leftarrow \mathbb{G}$ so that the tuple (g_0, h_0, g_1, h_1) is a non Diffie-Hellman tuple. Namely, there exist two distinct nonzero exponents $\alpha_0, \alpha_1 \in \mathbb{Z}_p$ so that $h_b = g_b^{\alpha_b}$ for $b \in \{0, 1\}$.

Our first observation is that in the semi-honest setting the receiver can generate the common reference string by obliviously sampling four random group elements from $\mathbb{G}$. This results in a non Diffie-Hellman tuple with overwhelming probability and also guarantees that a corrupted sender cannot leak α_0 and α_1. We continue with a high level description of the PVW protocol (adapted into the semi-honest setting), and then show how to modify it in order to achieve security against leakage.

Protocol 1 (π_{OT} – PVW Semi-Honest OT)

- **Inputs:** *Sender* Sen *has* $x_0, x_1 \in \mathbb{G}$ *and receiver* Rec *has* $\sigma \in \{0,1\}$.
- **Auxiliary Input:** *A group description* $\mathbb{G} = \mathbb{G}_{g,p}$ *specified by a generator* g *and its prime order* p.
- **The Protocol:**

  1. Rec *picks at random* $(g_0, h_0, g_1, h_1) \leftarrow \mathbb{G}$ *and* $r \leftarrow \mathbb{Z}_p$ *and sends* Sen *the following:* $((g_0, h_0, g_1, h_1), (g_\sigma^r, h_\sigma^r))$.
  2. *Upon receiving* $((g_0, h_0, g_1, h_1), (g', h'))$, Sen *picks two pairs of random exponents* $(v_0, u_0), (v_1, u_1) \leftarrow \mathbb{Z}_p^2$ *and returns the following:*
     - $(g_0^{v_0} h_0^{u_0}, g'^{v_0} h'^{u_0} \cdot x_0)$,
     - $(g_1^{v_1} h_1^{u_1}, g'^{v_1} h'^{u_1} \cdot x_1)$.
  3. *Upon receiving* $((v_0', u_0'), (v_1', u_1'))$, Rec *outputs* $u_\sigma' / (v_\sigma')^r$.

The proof of Protocol 1 follows from the DDH assumption, where for the case the sender is corrupted the simulator plays the role of the receiver with an arbitrary input bit (obtaining computational security). Whereas, for the case the receiver is corrupted the simulator plays the role of the sender with (x_σ, x_σ) (obtaining statistical security).

We continue with our leakage resilient OT construction. Note first that the message sent by the receiver can be seen as a public key, used by the sender to encrypt x_0 and x_1. So that, if the sender uses the right combination to encrypt, i.e., $(g_\sigma, h_\sigma, g_\sigma^r, h_\sigma^r)$, the receiver would be able to decrypt properly. Otherwise, the receiver does not learn anything (information theoretically) about $x_{1-\sigma}$. In order to protect $x_{1-\alpha}$ against leakage attacks we modify the PVW protocol so that the encryption scheme used by the parties is resilient to leakage.

More concretely, we use an encryption scheme Π from [NS09] that is CPA secure against key leakage attacks. Formally, let $\mathsf{Ext} : \mathbb{G} \times \{0,1\}^t \to \{0,1\}^m$ denote an average-case $(\log p - \lambda, \epsilon)$-strong extractor for some negligible $\epsilon = \epsilon(n)$ (cf. Definition 2.4). Define the secret key by a pair (a_1, a_2) of random exponents and the public key by $h = g_1^{a_1} g_2^{a_2}$ for random and public parameters g_1, g_2. In order to encrypt a plaintext $x \in \{0,1\}^m$ pick first random $r \leftarrow \mathbb{Z}_p, s \leftarrow \{0,1\}^t$ and then compute the ciphertext $c = (g_1^r, g_2^r, \mathsf{Ext}(h^r, s) \oplus x)$. Decryption follows easily given the secret key. As proven by Naor and Segev [NS09], this encryption scheme is CPA secure even in the presence of leaking $1/2 - o(1)$ bits from the secret key. We modify the PVW protocol so that the sender encrypts its inputs using Π by generating the secret keys by itself.

Protocol 2 (π_{LOT} – PVW Local Leakage Semi-Honest OT)

- **Inputs:** *Sender* Sen *has* $x_0, x_1 \in \{0,1\}^m$ *and receiver* Rec *has* $\sigma \in \{0,1\}$.
- **Auxiliary Input:** *A group description* $\mathbb{G} = \mathbb{G}_{g,p}$ *specified by a generator* g *and its prime order* p.
- **The Protocol:**

  1. Rec *picks at random* $(g_0, h_0, g_1, h_1) \leftarrow \mathbb{G}$ *and* $r \leftarrow \mathbb{Z}_p$ *and sends* Sen *the following:* $((g_0, h_0, g_1, h_1), (g_\sigma^r, h_\sigma^r))$.
  2. *Upon receiving* $((g_0, h_0, g_1, h_1), (g', h'))$, Sen *picks two pairs of random exponents* $(v_0, u_0), (v_1, u_1) \leftarrow \mathbb{Z}_p^2$ *and* $s_0, s_1 \leftarrow \mathbb{G}$ *and returns the following:*
     - $(g_0^{v_0} h_0^{u_0},\ \mathsf{Ext}(g'^{v_0} h'^{u_0}, s_0) \oplus x_0,\ s_0)$,

– $(g_1^{v_1} h_1^{u_1},\ \mathsf{Ext}(g'^{v_1} h'^{u_1}, s_1) \oplus x_1,\ \ s_1)$.

3. *Upon receiving* $((v'_0, u'_0, s_0), (v'_1, u'_1, s_1))$, Rec *outputs* $u'_\sigma \oplus \mathsf{Ext}((v'_\sigma)^r, s_\sigma)$.

We prove the following,

Theorem 5.1 (Leakage resilient OT) *Assuming the k-DDH assumption holds in $\mathbb{G}$. Then, Protocol 2 $((\frac{1}{3} - o(1))|sk_{\mathsf{Sen}}|, (1 - \omega(\log n)/\log p)|sk_{\mathsf{Rec}}|)$-securely computes the oblivious transfer functionality in the presence of local leakage (cf. 3.7), for $|sk_{\mathsf{Rec}}|, |sk_{\mathsf{Sen}}|$ the bit length of the secret memory content of the receiver and the sender, respectively.*[8]

For simplicity we assume that the inputs of the sender (x_0, x_1) are uniform. This accommodates our needs when employing this protocol as part of Yao's construction in Section 6, since the sender's inputs for the OT's in this protocol are uniformly chosen secret keys. We note that our theorem can be extended for any min-entropy source distribution.

When proving security in the only computational model a protocol designer has to take into consideration the fact that not only the content of the secret memory leaks, but also intermediate computations made by the honest party. When examining the leakage attacks this is made explicit as follows. First, we consider the case when the receiver is corrupted. Whenever the receiver is corrupted then the adversary leaks from $x_{1-\sigma}$ and $v_{1-\sigma}, u_{1-\sigma}$. We neglect x_σ since the receiver learns it anyway. Moreover, since (v_0, u_0) are chosen independently of (v_1, u_1) and since the simulator knows x_σ, we assume that the simulation of the encryption of x_σ and the leakage from (v_σ, u_σ) (picked by the simulator), are perfect. We therefore focus on $(v_{1-\sigma}, u_{1-\sigma}, x_{1-\sigma})$. Now a subtlety emerges here, as we must ensure that the adversary cannot obtain leakage from intermediate values computed while generating the public key and the ciphertext for encrypting $x_{1-\sigma}$. Otherwise, the statistical privacy of $x_{1-\sigma}$ that follows by the Naor-Segev proof breaks down, as it only holds in the presence of leakage from the secret key. To address the subtlety, we first restrict the set of leakage functions that can be submitted by the adversary so that, $(g_{1-\sigma}^{v_{1-\sigma}}, h_{1-\sigma}^{u_{1-\sigma}}, g'^{v_{1-\sigma}}, h'^{u_{1-\sigma}})$ are statistically close to uniform even when conditioned on the leakage from the internal memory of the sender. Formally,

Definition 5.2 *Denote by $\mathcal{H}_{\mathsf{Rec}}$ the class of leakage functions so that for any $f_1, \ldots, f_t \in \mathcal{H}_{\mathsf{Rec}}$, denoting the set of leakage functions submitted by an adversary corrupting the receiver, it holds that*

$$SD\left(((g_{1-\sigma}^{v_{1-\sigma}}, h_{1-\sigma}^{u_{1-\sigma}}, g'^{v_{1-\sigma}}, h'^{u_{1-\sigma}}), (u_1, u_2, u_3, u_4))|\ \tilde{f}_1, \ldots,\ \tilde{f}_t\right) \leq \mathsf{negl}(n)$$

for $u_1, u_2, u_3, u_4 \leftarrow \mathbb{G}$ and $\tilde{f}_i$ the outcome of f_i.

In Section 5.2.1 we show how to securely implement the role of the sender so that $(g_{1-\sigma}^{v_{1-\sigma}}, h_{1-\sigma}^{u_{1-\sigma}}, g'^{v_{1-\sigma}}, h'^{u_{1-\sigma}})$ are statistically close to uniform in the only computational setting, without restricting the class of leakage functions.

The case that the sender is corrupted is proven based on the k-DDH and k-IND assumptions (cf. Assumptions 2.5, 2.7, respectively). Here the content of the secret memory stored by the receiver is σ, r and $(g_0, h_0), (g_1, h_1)$ (which are then sent to the sender). Here, the adversary leaks from (g_σ, h_σ) and (σ, r) and from the computation yields that (g_σ^r, h_σ^r). The proof for this case is more challenging since the adversary may leak g_σ, h_σ and later compare this leakage to (g_0, h_0, g_1, h_1) (that receiver sends on clear), concluding the real value of σ without leaking from it at all. Another problem arises when relying on these hardness assumptions is the need to prevent the adversary from telling whether (g_σ^r, h_σ^r) were computed by the honest party or by an external challenger as happens in the reduction (see Section 3.3 for further discussion). We

[8] Recall that for simplicity we take $n = \log p$.

prove security in two phases: (**1**) We first design a simulator $\mathcal{S}_{\mathsf{Sen}}$ such that the proof against a corrupted sender works only for a restricted class of leakage functions $\mathcal{H}_{\mathsf{Sen}}$, made formal below. (**2**) We then show how to boost the security of this proof against any PPT leakage function by using $\mathcal{S}_{\mathsf{Sen}}$ as a building block for a new simulator $\mathcal{S}'_{\mathsf{Sen}}$.

Let us define the class of $\mathcal{H}_{\mathsf{Sen}}$ first.

Definition 5.3 *Let $\mathcal{A}$ denote an adversary corrupting the sender, then $f_1, \ldots, f_t \in \mathcal{H}_{\mathsf{Sen}}$ if the following holds:*

1. $\Pr[\mathcal{A}\left(\tilde{f}_1, \ldots, \tilde{f}_t, \mathbf{View}_{\pi_{\mathrm{LOT}},\mathcal{A}}(x_0, x_1)\right) = \sigma] \leq \Pr[\mathcal{A}\left(\tilde{f}_1, \ldots, \tilde{f}_t\right) = \sigma] + \mathsf{negl}(n)$

   for $\mathbf{View}_{\pi_{\mathrm{LOT}},\mathcal{A}}(x_0, x_1)$ the view of $\mathcal{A}$ within π_{LOT}, $\tilde{f}_i$ the outcome of f_i and a negligible function $\mathsf{negl}(\cdot)$. Namely, the probability of guessing σ given the leakage is essentially the same when given the adversary's view as well.

2. *For any i, f_i does not leak from intermediate computations made when computing g_σ^r, h_σ^r. Namely, f_i leaks from r* independently *of g_σ and h_σ.*

Proof: We continue with an overview of our proof for Theorem 5.1. We begin with a proof for Lemma 5.1 in Section 5.1 for proving security against restricted classes of leakage functions; $\mathcal{H}_{\mathsf{Rec}}$ and $\mathcal{H}_{\mathsf{Sen}}$. We then complete the proof by providing implementations for avoiding restrictions on leakage functions $\mathcal{H}_{\mathsf{Rec}}$ and Item 2 for $\mathcal{H}_{\mathsf{Sen}}$ in Section 5.2.1. Finally, we remove the restriction of Item 1 from the set $\mathcal{H}_{\mathsf{Sen}}$ in Section 5.2.2. ■

Lemma 5.1 *Assuming the k-DDH assumption holds in $\mathbb{G}$. Then, Protocol 2 $((\frac{1}{3} - o(1))|sk_{\mathsf{Sen}}|), (1 - \omega(\log n)/\log p)|sk_{\mathsf{Rec}}|$-securely computes the oblivious transfer functionality in the presence of local leakage (cf. 3.7), for leakage functions taken from $\mathcal{H}_{\mathsf{Rec}}$ and $\mathcal{H}_{\mathsf{Sen}}$ respectively, and $|sk_{\mathsf{Rec}}|, |sk_{\mathsf{Sen}}|$ the bit length of the secret memory content of the receiver and the sender, respectively.*[9]

5.1 Proof of Lemma 5.1

We consider two corruptions cases and build a simulator for each corruption case separately.

The case of a corrupted receiver Rec**.** We recall that the corrupted receiver obtains leakage from the secret memory of the sender which includes $(v_{1-\sigma}, u_{1-\sigma}, x_{1-\sigma}, v_\sigma, u_\sigma)$ (we neglect x_σ since the receiver learns this anyway). Moreover since (v_0, u_0) are independent of (v_1, u_1) and since the simulator knows x_σ, we assume that the simulation of the encryption of x_σ and the leakage from (v_σ, u_σ) (picked by the simulator), are perfect. We therefore focus on leaking from $(v_{1-\sigma}, u_{1-\sigma}, x_{1-\sigma})$ and proves that the adversary's view in a simulated and real executions is statistically close.

Denoting the message sent by the receiver by $((g_0, h_0, g_1, h_1), (g' = g_\sigma^r, h' = h_\sigma^r))$, we view r as the randomness picked for the ciphertext that encrypts x_σ. Namely, the ciphertext encrypting x_σ is defined by $c_\sigma = (g', h', \mathsf{Ext}(g'^{v_\sigma} h'^{u_\sigma}, s_\sigma) \oplus x_\sigma, s_\sigma)$ with respect to the secret key (v_σ, u_σ) and the public key $g_\sigma^{v_\sigma} h_\sigma^{u_\sigma}$.

The ciphertext encrypting $x_{1-\sigma}$, on the other hand, is defined by:

$$c_{1-\sigma} = (g', h', \mathsf{Ext}(g'^{v_{1-\sigma}} h'^{u_{1-\sigma}}, s_{1-\sigma}) \oplus x_{1-\sigma}, s_{1-\sigma})$$

[9] Recall that for simplicity we take $n = \log p$.

with respect to the secret key $(v_{1-\sigma}, u_{1-\sigma})$ and the public key $g_{1-\sigma}^{v_{1-\sigma}} h_{1-\sigma}^{u_{1-\sigma}}$. Note that $g' = g_\sigma^r$ can be seen as $g_{1-\sigma}^{r_1}$ for some r_1 which is uniformly distributed since g' is picked at random from $\mathbb{G}$. Similarly, $h' = h_\sigma^r$ can be seen as $h_{1-\sigma}^{r_2}$ for some random r_2. This implies that $c_{1-\sigma}$, defined by

$$c_{1-\sigma} = (g_{1-\sigma}^{r_1}, h_{1-\sigma}^{r_2}, \mathsf{Ext}(g_{1-\sigma}^{r_1 v_{1-\sigma}} h_{1-\sigma}^{r_2 u_{1-\sigma}}, s_{1-\sigma}) \oplus x_{1-\sigma}, s_{1-\sigma}).$$

is a ciphertext encrypted using the secret key $(v_{1-\sigma}, u_{1-\sigma})$. As shown in [NS09], this variant induces statistical security even in the presence of leaking $\log p - \omega(\log n) - m$ bits from the secret key. The fact that the receiver knows r does not compromise the privacy of $x_{1-\sigma}$ since r is independent of r_1 and r_2. The proof below formalizes this initiation. Formally, let $\mathcal{A}$ denote a PPT adversary controlling the receiver Rec, design a simulator $\mathcal{S}_{\mathsf{Rec}}$ as follows.

1. On input σ the simulator picks random exponentiations (v_0, u_0, v_1, u_1) and random group elements (s_0, s_1) as specified in Protocol 2.
2. If the adversary submits a leakage function f to be applied on $v_{1-\sigma}, u_{1-\sigma}$ and $x_{1-\sigma}$, the simulator defines a new leakage function f' so that f is incorporated with the same code of f expect that $v_{1-\sigma}, u_{1-\sigma}$ are already hardwired within it, instead of handing it to f as an input. $\mathcal{S}_{\mathsf{Rec}}$ submits f' to the trusted party and receives back the leakage.
3. $\mathcal{S}_{\mathsf{Rec}}$ concludes the execution by handing the adversary the leakage from the trusted party and playing the role of the honest sender with input $x_{1-\sigma} = 0^{\log p}$.

Note first that $\mathcal{S}_{\mathsf{Rec}}$ runs in polynomial time. We continue with a proof that shows that the joint output of the corrupted receiver and the honest sender is statistically indistinguishable in both real and simulated executions. Let $\mathbf{Output}_{\mathrm{OT}}^{\mathrm{rl}}(\mathsf{Sen}(x_0, x_1, n), \mathcal{A}(\sigma, n))$ (similarly, $\mathbf{Output}_{\mathrm{OT}}^{\mathrm{sm}}(\mathsf{Sen}(x_0, x_1, n), \mathcal{A}(\sigma, n))$) denote the joint outputs of the parties in the real (similarly, ideal) execution of Protocol 2 with inputs (x_0, x_1) for the sender and σ for the receiver and a security parameter n (we remind the reader that the without loss of generality, the adversary outputs its view and leakage). Then,

Claim 5.2 $\mathbf{Output}_{\mathrm{OT}}^{\mathrm{rl}}(\mathsf{Sen}(x_0, x_1, n), \mathcal{A}(\sigma, n)) \approx_s \mathbf{Output}_{\mathrm{OT}}^{\mathrm{sm}}(\mathsf{Sen}(x_0, x_1, n), \mathcal{A}(\sigma, n))_{x_1, x_2, n}$.

Since the sender does not return any output, the proof of this claim completes the proof.

Proof: Note that these distributions differ only with respect to the plaintext $x_{1-\sigma}$ encrypted by the real sender and the simulator that emulates the role of the honest sender. We claim that if the sender is modified so that it picks g', h' by itself (i.e., the randomness used for computing the encryption of $x_{1-\sigma}$), then our proof is identical to the proof of Naor and Segev [NS09]. Loosely speaking, they showed that the difference between these distributions is upper bounded by the difference induced by the extractor Ext, even in the presence of $1/2 - o(1)$ key leakage from the secret key.

We observe first that the fact that $\mathcal{A}$ picks g', h' by itself does not change the argument of Naor and Segev since the adversary picks these values at random as required. Specifically, their statistical argument holds even if $\mathcal{A}$ knows $\log_g g', \log_g h'$ *before* the ciphertext is computed, since these are picked independently of the rest of the computation and are part of the ciphertext. In addition, we recall that the proof works for the leakage functions in $\mathcal{H}_{\mathsf{Rec}}$ that ensures that the secure exponentiation assumption holds, so that the result of exponentiation is statistically close to a uniform element in $\mathbb{G}$. Assuming that $\mathcal{A}$ cannot leak $g_{1-\sigma}^{v_{1-\sigma}}, h_{1-\sigma}^{u_{1-\sigma}}, g'^{v_{1-\sigma}}, h'^{u_{1-\sigma}}$, the proof is concluded since the adversary's view is exactly as in the Naor-Segev proof. Below we show how to implement such a computation.[10] Finally, since we consider a larger

[10] A subtlety emerges here from the fact that the adversary may leak the exponents $v_{1-\sigma}, u_{1-\sigma}$ based on g' and h' (and not separately as required) since it picks these values itself. We show how to deal with this problem as well.

secret state with three group elements (i.e. $v_{1-\sigma}, u_{1-\sigma}$ and $x_{1-\sigma}$) we reduce the leakage rate into $1/3 - o(1)$ to ensure that the Naor-Segev proof holds. We remark that this proof holds in the memory leakage setting where the entire memory leaks. The only computational assumption is required for proving that intermediate computations from generating the public key (i.e. $g_{1-\sigma}^{v_{1-\sigma}} h_{1-\sigma}^{u_{1-\sigma}}$) and the ciphertext for encrypting $x_{1-\sigma}$, do not leak. ■

The case of a corrupted sender Sen**.** By the assumption made on $\mathcal{H}_{\mathsf{Sen}}$, a corrupted sender obtains leakage from the secret memory of the receiver which includes σ, r and the random elements that are sent to the sender. We let the leakage rate $((1 - \omega(\log n)/\log p)$ denote the overall leakage rate from r through the entire execution (capturing the leakage from r twice). As for the single bit σ, we show in Section 5.2.2 that it either leaks entirely or does not leak at all. Let $\mathcal{A}$ denote a PPT adversary controlling the sender Sen, design a simulator $\mathcal{S}_{\mathsf{Sen}}$ as follows.

1. On input (x_0, x_1) the simulator picks four random group elements (g_0, h_0, g_1, h_1) as specified in Protocol 2.

2. Next, $\mathcal{S}_{\mathsf{Sen}}$ picks at random $r \leftarrow \mathbb{Z}_p$. If the adversary submits a leakage function f, the simulator defines a new leakage function f' so that f is incorporated with the same code of f except that r and (g_0, h_0, g_1, h_1) are already hardwired within it. Instead of handing to f, $\mathcal{S}_{\mathsf{Sen}}$ submits f' to the trusted party and receives back the leakage.

3. $\mathcal{S}_{\mathsf{Sen}}$ concludes the execution by handing the adversary the leakage from the trusted party and playing the role of the honest verifier with input $\sigma = 0$.

Note first that $\mathcal{S}_{\mathsf{Sen}}$ runs in polynomial time. We continue with a proof that shows that the joint view of the corrupted sender and the honest receiver is computationally indistinguishable in both real and simulated executions. Let $\mathbf{Output}^{\mathrm{rl}}_{\mathrm{LOT}}(\mathcal{A}(x_0, x_1, n), \mathsf{Rec}(\sigma, n))$ (similarly, $\mathbf{Output}^{\mathrm{sm}}_{\mathrm{OT}}(\mathcal{A}(x_0, x_1, n), \mathsf{Rec}(\sigma, n))$) denote the joint outputs of the parties in the real (similarly, simulated) execution of Protocol 2 with inputs (x_0, x_1) for the sender and σ for the receiver and a security parameter n. Then,

Claim 5.3 $\mathbf{Output}^{\mathrm{rl}}_{\mathrm{LOT}}(\mathcal{A}(x_0, x_1, n), \mathsf{Rec}(\sigma, n))_{x_1, x_2, n} \approx_c \mathbf{Output}^{\mathrm{sm}}_{\mathrm{LOT}}(\mathcal{A}(x_0, x_1, n), \mathsf{Rec}(\sigma, n))_{x_1, x_2, n}$.

Proof: Clearly, in case $\sigma = 0$ the real and the simulated views are identically distributed. Therefore, we are interested in proving this claim in case $\sigma = 1$. The proof follows by a reduction to the hardness of k-DDH and k-IND assumptions. Formally, define a new hybrid game $\mathbf{Hybrid}_1$ with a non-interactive simulator $\mathcal{S}^1_{\mathsf{Sen}}$ that is identical to simulator $\mathcal{S}_{\mathsf{Sen}}$ except for the following. In this game there is no trusted party and $\mathcal{S}^1_{\mathsf{Sen}}$ answers the leakage queries of $\mathcal{A}$ by itself. Moreover, when fixing g', h', $\mathcal{S}_{\mathsf{Sen}}$ picks r uniformly at random from $\mathbb{Z}_p$ and computes $g' = g^r$. It then randomly picks h' from $\mathbb{G}$. Note that the only difference between the view in this game and the simulated view is with respect to the way h' is generated since the leakage is identically simulated. Namely, in the simulation (g_0, h_0, g', h') is a Diffie-Hellman tuple whereas in $\mathbf{Hybrid}_1$, (g_0, h_0, g', h') is a non Diffie-Hellman tuple. Therefore, any distinguisher that distinguishes these views can be reduced to a distinguisher that distinguishes a Diffie-Hellman tuple from a non Diffie-Hellman tuple, which essentially breaks the k-DDH assumption.

Assume by contradiction the existence of a PPT adversary $\mathcal{A}$, a PPT distinguisher D and a polynomial $p(\cdot)$ such that for infinitely many n's it holds that

$$\Pr[D(\mathbf{Output}^{\mathrm{sm}}_{\mathrm{LOT}}(\mathcal{A}(x_0, x_1, n), \mathsf{Rec}(\sigma, n)) = 1] - \Pr[D(\mathbf{Output}^{\mathrm{Hyb}_1}_{\mathrm{LOT}}(\mathcal{A}(x_0, x_1, n), \mathsf{Rec}(\sigma, n)) = 1] \geq 1/p(n)$$

we build an adversary $\mathcal{A}_{\mathrm{k-DDH}}$ that externally interacts with a challenger in the k-DDH game, defined above, and internally interacts with adversary $\mathcal{A}$. $\mathcal{A}_{\mathrm{k-DDH}}$ is defined as follows.

1. Upon receiving (x_0, x_1, σ) and the auxiliary z input of $\mathcal{A}$, $\mathcal{A}_{\text{k-DDH}}$ invokes $\mathcal{A}((x_0, x_1), z)$ and emulates the role of the receiver on σ.

2. Whenever $\mathcal{A}$ outputs a leakage function f to be applied on σ and r, $\mathcal{A}_{\text{k-DDH}}$ defines a new function f' so that f is incorporated with the same code of f expect that σ is already hardwired within it. Instead of handing f, $\mathcal{A}_{\text{k-DDH}}$ gives f' to the challenger and forwards $\mathcal{A}$ the leakage returned from its challenger.

3. Let $(g, h, \tilde{g}, \tilde{h})$ denote the tuple T and $\tilde{f}$ the leakage returned by the challenger. $\mathcal{A}_{\text{k-DDH}}$ fixes $(g_0 = g, h_0 = h, g_1, h_1)$ for randomly picked g_1, h_1 from $\mathbb{G}$. Furthermore, $\mathcal{A}_{\text{k-DDH}}$ fixes $(g' = \tilde{g}, h' = \tilde{h})$.

4. $\mathcal{A}_{\text{k-DDH}}$ outputs whatever $\mathcal{A}$ does.

We first observe that in case $\tilde{g} = g^r$ and $\tilde{h} = h^r$ then the adversary's view is distributed as in the simulated execution. This is because the messages sent by $\mathcal{A}_{\text{k-DDH}}$ are computed as if $\sigma = 0$. In addition, the leakage is identically distributed since it is applied on the real σ and r. On the other hand, if $\tilde{g} = g^r$ but $\tilde{h} = h^{r'}$ for r' an independent value of r, then the adversary's view is distributed as in the hybrid game. This is due to the fact that the message sent by the receiver is computed as in the hybrid game where h' is picked at random from $\mathbb{G}$. Nevertheless, the leakage is still applied on r as in $\mathbf{Hybrid}_1$. This concludes the proof since it implies that

$$\Pr[\mathcal{A}_{\text{k-DDH}}(g, h, g^r, h^r, \tilde{f}) = 1] - \Pr[\mathcal{A}_{\text{k-DDH}}(g, h, g^r, h^{r'}, \tilde{f}) = 1] \geq 1/p(n).$$

We recall that this proof assumes that intermediate computations from the computations g'^r, h'^r do not leak. Therefore, this reduction holds in the presence of leaking $(1 - \omega(\log n)/\log p)$ bits from r.

We define a new game $\mathbf{Hybrid}_2$ with a non-interactive simulator $\mathcal{S}^2_{\mathsf{Sen}}$ that is identical to simulator $\mathcal{S}^1_{\mathsf{Sen}}$ except that $\mathcal{S}^2_{\mathsf{Sen}}$ computes g' to be an uniform value in $\mathbb{G}$ that is independent of r. By the k-IND assumption, this ensures that the views generated in $\mathbf{Hybrid}_1$ and $\mathbf{Hybrid}_2$ are computationally close.

We define a new game $\mathbf{Hybrid}_3$ with a non-interactive simulator $\mathcal{S}^3_{\mathsf{Sen}}$ that plays the role of the honest receiver with $\sigma = 1$. That is, the message sent to $\mathcal{A}$ is computed so that $g' = g_1^{r'}$ and $h' = h_1^{r''}$ for random and independent r', r''. We claim that the adversary's view in $\mathbf{Hybrid}_2$ and $\mathbf{Hybrid}_3$ is identically distributed. In particular, $(g_0^{r'}, h_0^{r''})$ is identically distributed to $(g_1^{r'}, h_1^{r''})$ for random and independent r', r'' even given (g_0, h_0, g_0, h_1) and leakage from r. This is because r is chosen independently of r', r''.

In order to complete the proof we define a new game $\mathbf{Hybrid}_4$ with a non-interactive simulator $\mathcal{S}^4_{\mathsf{Sen}}$ that is identical to simulator $\mathcal{S}^3_{\mathsf{Sen}}$ except that $\mathcal{S}^4_{\mathsf{Sen}}$ computes g' as in game $\mathbf{Hybrid}_1$. From the statistical indistinguishability of $g_1^{r'}$ and g_0^r the views are indistinguishable.

Finally, the proof for which the output distribution from real execution is computationally indistinguishable from the output distribution induced by game $\mathbf{Hybrid}_4$, is identical to the proof in which the simulated output distribution and the output distribution induced by game $\mathbf{Hybrid}_1$ are computationally indistinguishable. This concludes the proof.

Note that the proof holds for the memory leakage setting as well. ■

5.2 Extending Security to General Leakage Functions

5.2.1 Secure Exponentiation

In this subsection we show how to compute the exponentiations in a secure way, such that the adversary cannot leak from data that a naive implementation would touch. This removes the restriction on the class $\mathcal{H}_{\mathsf{Rec}}$ and the last restriction (Item 2 in the Definition of $\mathcal{H}_{\mathsf{Sen}}$) on $\mathcal{H}_{\mathsf{Sen}}$.

We begin with the case that the receiver is corrupted. We require that the adversary cannot leak from $g_{1-\sigma}^{v_{1-\sigma}}, h_{1-\sigma}^{u_{1-\sigma}}, g'^{v_{1-\sigma}}, h'^{u_{1-\sigma}}$ since otherwise the privacy of $x_{1-\sigma}$ is not guaranteed any more. We employ ideas used by [DF12]. In this paper Faust and Dziembowski construct (amongst other things) a leakage resilient identification protocol by adapting Okamoto's scheme [Oka92] into the leakage resilient setting. Okamoto's protocol requires from the prover to perform similar computations to what is required from the sender in our OT construction. We briefly review their technique for computing $g'^{v_0}h'^{u_0}$ (and the rest of computations) securely. Instead of storing v_0, u_0 in the memory using two group elements, sample $V_0, U_0, Z_0 \leftarrow \mathbb{Z}_p^\ell$ and compute $\left(g'^{v_0^1}h'^{u_0^1}\right)^{z_0^1}, \ldots, \left(g'^{v_0^n}h'^{u_0^\ell}\right)^{z_0^n}$ for $V_0 = (v_0^1, \ldots, v_0^\ell), U_0 = (u_0^1, \ldots, u_0^\ell)$ and $Z_0 = (z_0^1, \ldots, z_0^\ell)$. This implies that one can view $v_0 = \langle V_0, Z_0 \rangle$ and $u_0 = \langle U_0, Z_0 \rangle$. The proof from [DF12] shows that v_0, u_0 are statistically close to uniform when employing the computation as above in the only computational model, so that the fraction of tolerated leakage is at most $(\ell \log p)/2$ bits.

We now consider the case of a corrupted sender. In this case we wish to compute exponentiations g_σ^r, h_σ^r without leaking anything but $f(r)$ such that f is a leakage function with some specified leakage rate. Specifically, we wish to implement these exponentiations in a black-box manner. To do so, we rely on a recent result by Akavia et al. [AGH] who show a continual leakage PKE in the distributed setting (where the secret key is shared amongst two parties). The crux of their proof is in showing how to simulate decryptions for the CPA game in the presence of leakage. Assuming the existence of a homomorphic SKE scheme (for the full details see [AGH]), this is achieved by tricking the adversary and decrypting fake ciphertexts into desired plaintexts. We use their protocol in order to implement secure exponentiation. The idea is as follows. A generator g is stored in the memory using random $g_1, \ldots, g_\ell$ and $t_1, \ldots, t_\ell$, so that $g = \prod_i g_i^{t_i}$. Then, for computing g^r the receiver emulates the following protocol in its head as would have been executed between the two memory parts M_1 and M_2: M_1 computes first the encryptions of $g_1, \ldots, g_\ell$ with respect to the above SKE, and "sends" these ciphertexts to M_2 which keeps $t_1, \ldots, t_\ell$. Given $t_1, \ldots, t_\ell$ and r, M_2 computes the encryption of $\prod_i g_i^{rt_i}$ and returns this encryption c to M_1 which decrypts it. The result of [AGH] shows, based on the leftover hash lemma, that g is statistically close to uniform when tolerating $1 - o(1)$ fraction of leakage from both $g_1, \ldots, g_\ell$ and $t_1, \ldots, t_\ell$, as long as leakage from $g_1, \ldots, g_\ell$ and $t_1, \ldots, t_\ell$ is computed independently. This implies that the adversary does not learn anything but the computed outcome g^r. Similarly, we compute h^r using $h_1, \ldots, h_\ell$ and $t'_1, \ldots, t'_\ell$.

5.2.2 Removing the final Restriction on the Leakage Functions

In this section we focus on extending our proof from Section 5.1 into the general case by eliminating the first restriction (Item 1 of the definition of $\mathcal{H}_{\mathsf{Sen}}$) on the class $\mathcal{H}_{\mathsf{Sen}}$. We recall that the proof for Lemma 5.1 may fail only if a corrupted sender leaks from g_σ, h_σ, since then the adversary can compare its leakage to the public values $(g_0, h_0), (g_1, h_1)$ and guess σ. Consider the leakage obtained from either g_σ or h_σ via functions $f_1, \ldots, f_t$. For simplicity consider only leakage from g_σ, the argument for h_σ follows. It can be easily seen that there are two options when leaking from g_σ. Either for some i, $f_i(g_0) \neq f_i(g_1))$ or not. Note that in the former case the adversary can easily learn σ by simply comparing the leakage against g_0 and g_1. Whereas, in the later case the adversary cannot learn anything about σ from the leakage.

We continue with a formal description of $\mathcal{S}'_{\mathsf{Sen}}$. We then show that $\mathcal{S}'_{\mathsf{Sen}}$ meets definition 3.7. Fix $\mathcal{A}$ an adversary that controls the sender; define $\mathcal{S}'_{\mathsf{Sen}}$ as follows.

1. Upon receiving (x_0, x_1), $\mathcal{S}'_{\mathsf{Sen}}$ fixes the randomness $R_{\mathcal{A}}$ for $\mathcal{A}$ and invokes $\mathcal{A}$ on $(x_0, x_1, R_{\mathcal{A}})$.
2. $\mathcal{S}'_{\mathsf{Sen}}$ picks randomness for the honest receiver R_{Sen}. Upon receiving a leakage query f from $\mathcal{A}$, $\mathcal{S}'_{\mathsf{Sen}}$ records f and checks if the input to f is g_σ. If so, $\mathcal{S}'_{\mathsf{Sen}}$ verifies whether $f(g_0) \neq f(g_1)$ and defines a new function f' that leaks σ in this case. Otherwise, $\mathcal{S}'_{\mathsf{Sen}}$ invokes $\mathcal{S}_{\mathsf{Sen}}$ on f to get the leakage for $\mathcal{A}$.

3. In case f' is defined, $\mathcal{S}'_{\mathsf{Sen}}$ submits f' to the trusted party and receives σ back. $\mathcal{S}'_{\mathsf{Sen}}$ uses σ to complete the simulation and outputs whatever $\mathcal{A}$ does. Otherwise, $\mathcal{S}'_{\mathsf{Sen}}$ invokes simulator $\mathcal{S}_{\mathsf{Sen}}$ defined in Section 5.1 and outputs whatever $\mathcal{S}_{\mathsf{Sen}}$ does.

Note first that $\mathcal{S}'_{\mathsf{Sen}}$ runs in polynomial time since it only invokes $\mathcal{S}_{\mathsf{Sen}}$ and evaluates PPT leakage functions. We analyze two cases here. (**1**) $\mathcal{S}'_{\mathsf{Sen}}$ leaks σ. (**2**) $\mathcal{S}'_{\mathsf{Sen}}$ invokes simulator $\mathcal{S}_{\mathsf{Sen}}$.

The first case. We begin with the former case and show how to generate a new fake view $\mathbf{View}^{\mathrm{fk}}_{\pi,\mathcal{A}}((x_0,x_1),\sigma)$ so that

$$\mathbf{View}^{\mathrm{fk}}_{\pi,\mathcal{A}}((x_0,x_1),\sigma) \approx_c \mathbf{View}^{\mathrm{rl}}_{\pi,\mathcal{A}}((x_0,x_1),\sigma) \tag{3}$$

and σ does not lose any entropy given $\mathbf{View}^{\mathrm{fk}}_{\pi,\mathcal{A}}((x_0,x_1),\sigma)$. We then argue that leaking σ by $\mathcal{S}'_{\mathsf{Sen}}$ is allowed according to Definition 3.7 since the leakage in real world and $\mathbf{View}^{\mathrm{fk}}_{\pi,\mathcal{A}}((x_0,x_1),\sigma)$ leaks σ completely. Finally, we argue that the adversary's view is indistinguishable in both simulated and real executions i.e.

$$\mathbf{View}^{\mathrm{sm}}_{\pi,\mathcal{A}}((x_0,x_1),\sigma) \approx_c \mathbf{View}^{\mathrm{rl}}_{\pi,\mathcal{A}}((x_0,x_1),\sigma)$$

This essentially follows immediately from the fact that $\mathcal{S}'_{\mathsf{Sen}}$ leaks σ.

More formally, define simulator $\mathcal{S}_{\mathrm{fk}}$ generating the fake view as follows. Given σ, $\mathcal{S}_{\mathrm{fk}}$ first generates a Diffie-Hellman tuple $\xi = ((g_0,h_0),(g_1,h_1))$ and completes the execution of Protocol 2, playing the role of the honest receiver on his input bit. Specifically, $\mathcal{S}_{\mathrm{fk}}$ does whatever an honest receiver would do while pretending to generate ξ as in Protocol 2. (Note that in the fake game we can deviate from the instructions of the receiver as specified in the real execution.) We now claim that Equation (3) holds. This is because from the adversary's viewpoint the only difference is with respect to tuple ξ, which by the DDH assumption is computationally indistinguishable since the adversary is not given any access to the randomness used for generating ξ.

We also note that the remaining average min-entropy of σ conditioned on the fake view is 1 since the message sent by the receiver is distributed identically in both potential values of σ (note that the view does not include the leakage).

Next, we claim that leaking σ is allowed according to Definition 3.7. This is due to the fact that $\mathcal{S}'_{\mathsf{Sen}}$ asks to leak σ only in case that $f(g_0) \neq f(g_1)$. Now, since the adversary sees $f(g_\sigma), g_0, g_1$ it can easily conclude the value of σ given the fake view. Thus, the remaining (average) min-entropy of σ condition on $f(g_\sigma)$ and the fake view is still 0.

The second case. In the later case it holds that $f(g_0) = f(g_1)$ and $\mathcal{S}'_{\mathsf{Sen}}$ does not leak σ. We first argue that Equation (3) holds. This is due to the fact that $\mathcal{S}_{\mathrm{fk}}$ knows σ and can follow the role of the honest receiver on this input and answers the adversary's leakage queries. Thus, security is reduced to DDH as before. The fact that the average min-entropy of σ conditioned on the fake view is 1 follows as above. We conclude with a claim that

$$\mathbf{View}^{\mathrm{sm}}_{\pi,\mathcal{A}}((x_0,x_1),\sigma) \approx_c \mathbf{View}^{\mathrm{rl}}_{\pi,\mathcal{A}}((x_0,x_1),\sigma)$$

for $\mathbf{View}^{\mathrm{sm}}_{\pi,\mathcal{A}}((x_0,x_1),\sigma)$ the simulated view generated by $\mathcal{S}'_{\mathsf{Sen}}$. This follows immediately from the proof of Theorem 5.1.

6 Secure Two-party Computation in the Presence of Leakage

In this section we present our construction for securely computing any two-party functionality in the $\mathcal{F}_{\mathrm{OT}}$-hybrid setting with semi-honest adversaries and local leakage attacks. We begin with a brief, but essential,

overview on Yao's protocol for two party computation in the presence of semi-honest adversaries. This description is given in Section 6.1. In Section 6.2 we present our garbling technique for the local setting and the properties of the SKE used for this garbling. We present the proof for Yao's protocol in Section 6.3.

6.1 A High Level Overview of Yao's Protocol

In this section we give a brief exposition of the original protocol and its building blocks as presented by Lindell and Pinkas in [LP09]. In their paper, Lindell and Pinkas presented an in-depth study and a comprehensive exposition of Yao's protocol together with a complete detailed proof for semi-honest adversaries. Formally speaking, in Yao's protocol the desired functionality f is represented by a boolean circuit C. Assuming x_0 and x_1 to be the inputs of the parties P_0 and P_1 respectively, the circuit C with inputs x_0, x_1 is computed gate-by-gate from the input wires to the output wires. In the following, we distinguish four different types of wires in a given boolean circuit: (a) circuit-input wires (that receives x_0 and x_1 as inputs); (b) circuit-output wires (that carry the value of $C(x_0, x_1)$); (c) gate-input wires (that enter some gate g); and (d) gate-output wires (that leave some gate g).

The underlying idea is to associate every wire w with two random values, say k_w^0, k_w^1, such that k_w^0 represents the bit 0 and k_w^1 represents the bit 1. Note that even if one of the parties knows that the value k_w^σ is obtained by wire w, it does not help it to determine the exact value for σ since k_w^0, k_w^1 are identically distributed. The garbled table for each gate maps random input values to random output values, with the property that given two input values it is only possible to learn the output value that corresponds to the output bit of the gate computed based on these inputs (so that the other value must be kept a secret). This is accomplished by viewing the four potential inputs to the gate k_1^0, k_1^1 (values associated with the first input wire) and k_2^0, k_2^1 (values associated with the second input wire), as encryption keys. Then the output values k_3^0, k_3^1, which are also keys, are encrypted under the appropriate input keys. For instance, let g be a NAND gate. Then, k_3^1 (that corresponds to bit 1) is encrypted under the pair of keys associated with the values $(0,0),\ (0,1),\ (1,0)$. Whereas, k_3^0 is encrypted under the pair of keys associated with $(1,1)$. These four encryptions, denoted by

$$\mathsf{Enc}_{k_1^0}(\mathsf{Enc}_{k_2^0}(k_3^1)),\ \ \mathsf{Enc}_{k_1^0}(\mathsf{Enc}_{k_2^1}(k_3^1)),\ \ \mathsf{Enc}_{k_1^1}(\mathsf{Enc}_{k_2^0}(k_3^1))\ \text{ and }\ \mathsf{Enc}_{k_1^1}(\mathsf{Enc}_{k_2^1}(k_3^0)),$$

are randomly permuted to obtain the garbled table for gate g. Given the input wire keys k_1^α, k_2^β that correspond to the bits α and β and the garbled table containing the four encryptions, it is possible to obtain the output wire key $k_3^{g(\alpha,\beta)}$. Moreover, this should be the only information revealed.

The description of the garbled circuit is concluded with the *output decryption tables*, mapping the random values on the circuit output wires back to their corresponding boolean values. That is, every circuit output wire w is associated with the pairs $(0, k_w^0)$ and $(1, k_w^1)$ so that once a value k_w^γ is obtained on a circuit output wire, it is possible to determine actual circuit output bit by comparing the key to the values in the output decryption table. Therefore, given the keys associated with the inputs x_0 and x_1, it is possible to obliviously compute the entire circuit gate-by-gate and finally obtain the keys on the circuit-output wires; the output of the circuit can be obtained by comparing the keys to the output-decryption tables.

With the above discussion, we are now ready to informally describe Yao's protocol. In this protocol, one of the parties, henceforth P_0, constructs a garbled circuit and sends it to the other party, henceforth P_1. P_0 and P_1 interact via a sequence of OT protocols, so that P_1 learns the keys for the circuit-input wires that are associated with x_0, x_1. Given these keys P_1 can compute the circuit as described and obtain the output and conclude the protocol.

In the following, we explore the two main tools for the protocol in light of leakage attacks: SKE used for constructing the garbled gates (cf. Section 6.2) and a leakage resilient OT (cf. Section 5).

6.2 SKE for Garbling

In this section we study the properties required from the secret key scheme used for garbling the circuit in the presence of leakage attacks. Loosely speaking, we wish our SKE to provide indistinguishability even in the presence of key leakage. Nevertheless, the notion of (computational) semantic security in the presence of leakage does not accurately capture leakage attacks occurred within the local setting, as the adversary is allowed to leak from the *each* memory part that is involved in the encryption process; this includes the secret key as well as the randomness used to generate the ciphertexts. We therefore use SKE with statistical security and bound the remaining average min-entropy of the plaintexts (that are picked at uniform) conditioned on the leakage obtained by the adversary and its view. This restricts us to compute only functions with logarithmic depth size (in the security parameter) since the key length must be decreased within the levels of the circuit. More precisely, consider two output keys k_w^0, k_w^1 that are associated with the output wire w and let k_w^0 be the output key obtained by a corrupted P_1 while evaluating the circuit. We require that the remaining entropy in k_w^1 is still high enough even given the adversary's view and leakage, so that any plaintext encrypted with k_w^1 is still statistically secure. We formalize this intuition below.

Two more important properties about the SKE range [LP09] are elusive and efficiently verifiable range. Loosely speaking, elusive range means that an encryption under one key falls in the range of an encryption under another key with at most negligible probability. Whereas, efficiently verifiable range implies that it is possible to efficiently verify if a given ciphertext is in the range of a given key k. This additional property from the secret key scheme is required to enable P_1 to compute the garbled circuit correctly. Namely, for every gate P_1 should be able to decrypt the right output key. We use the approach from [NPS99] to sort this out. Namely, by using the least significant bit of each input key to immediately conclude the row in the garbled table of the specified gate. This way P_1 only needs to decrypt a single ciphertext. This poses a restriction on the least significant bit of each key and has no effect for security considerations.

Finally, we require from our SKE that $\mathcal{K} = \mathcal{M} = \mathcal{C}$ for the key, plaintext and ciphertext spaces, respectively. This is because the output keys of some gate are encrypted under the input keys of this gate. Furthermore, the plaintext and ciphertext spaces should be identical for the obvious reason of a double encryption. Enforcing the plaintext and the ciphertext spaces to be same is immediate for our particular choice of SKE. We further show how to handle the case where the key and the plaintext spaces by allowing these spaces to be different but connected via a one-to-one mapping. We summarize our requirements in the following definition.

Definition 6.1 (Good SKE for garbling.) *We say that* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a* good SKE for garbling *if the following properties hold,*

1. Π *is one-time statistical secure even in the presence of* λ *key leakage. Namely, for each* $m_0, m_1 \in \mathcal{M}$ *and* $sk \leftarrow \mathsf{Gen}(1)$*,*

$$(\mathsf{Enc}_{sk}(m_0), m_0, m_1, f(sk)) \approx_s (\mathsf{Enc}_{sk}(m_1), m_0, m_1, f(sk))$$

   for function f *with leakage rate* λ*.*

2. $\mathcal{K} = \mathcal{M} = \mathcal{C}$*, for* $\mathcal{K}$ *the key space,* $\mathcal{M}$ *the plaintext space and* $\mathcal{C}$ *the ciphertext space.*

We now show that there exists an SKE that meets Definition 6.1. We begin with a description of the underlying SKE and then show how to use it to compute the garbled table. Let $(g, p) \leftarrow \mathcal{G}(1^n)$ be an algorithm that given a security parameter n, returns a group description $\mathbb{G} = \mathbb{G}_{g,p}$ specified by a generator g and its prime order p. Then,

Proposition 6.1 *For every parameters generating algorithm $(p, g) \leftarrow \mathcal{G}(1^n)$ and order p group $\mathbb{G}$ generated by $\mathcal{G}$, there exists an SKE, denoted by Π_{GARB} that is good for garbling in the local leakage setting.*

Proof: Consider $\Pi_{\text{GARB}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ defined as follows.

- KEY GENERATION. Given $(1^n, 1^\lambda)$ for a security parameter n, set the public parameters to be $(g, p) \leftarrow \mathcal{G}(1^n)$ and $\kappa = 1 + \frac{\lambda + 2\log(1/\epsilon)}{\log p}$ for some negligible $\epsilon = \epsilon(n)$.
  Choose uniformly random $(s_1, ...s_\kappa) \leftarrow \mathbb{Z}_p^\kappa \setminus \{0^\kappa\}$. Output the public parameters $params = (g, p)$, and the secret key $sk = (s_1, ..., s_\kappa)$.
- ENCRYPTION. On input a plaintext $m \in \mathbb{G}$, choose uniformly random $(a_1, ..., a_\kappa) \leftarrow \mathbb{G}^\kappa$. Output the ciphertext $(a_1, ..., a_\kappa, m \cdot \prod_{i\in[\kappa]} a_i^{s_i})$.
- DECRYPTION. On input a ciphertext $(c_1, ..., c_\kappa, \Phi)$, output $\Phi \cdot \left(\prod_{i\in[\kappa]} c_i^{s_i}\right)^{-1}$.

It can be shown that by the leftover hash lemma (cf. Theorem 2.3), any ciphertext $(c_1, ..., c_\kappa, \Phi)$ is ϵ-close to $(c_1, ..., c_\kappa, U_\mathbb{G})$ for a uniformly random $U_\mathbb{G} \leftarrow \mathbb{G}$, even in the presence of $\lambda = \kappa \log p - \log p - 2\log\frac{1}{\epsilon}$ secret key leakage [NPS99]. In other words, any ciphertext $(c_1, ..., c_\kappa, \Phi)$ is ϵ-close to $(c_1, ..., c_\kappa, U_\mathbb{G})$ for a uniformly random $U_\mathbb{G} \leftarrow \mathbb{G}$, as long as $H_\infty(sk) = \kappa \log p - \lambda = \log p + 2\log\frac{1}{\epsilon}$. So, one-time statistical security follows from this argument.

Next, we note that the secret key space is $\mathbb{Z}_p$ whereas, the plaintext and ciphertext spaces are the group $\mathbb{G}$. In order to maintain the second property of a good SKE for garbling we define a bijective mapping from $\mathbb{G}$ to $\mathbb{Z}_p$, denoted by $\psi : \mathbb{Z}_p \to \mathbb{G}$, where an element $j \in \mathbb{Z}_p$ is mapped to the $(j+1)$th smallest element in $\mathbb{G}$. Namely, when a secret key is initially picked (to be encrypted as part of a garbled table), it is picked from $\mathbb{G}$. Later, when it is used to encrypt the garbled table of the following gate it is mapped into $\mathbb{Z}_p$.

We present a particular instantiation for $\mathbb{G}$ and ψ. Formally, we consider the group $\mathbb{Z}_{p'}$ where $p' = 2p+1$ is a prime, and set $\mathbb{G}$ to be the subgroup of $\mathbb{Z}_{p'}$ of quadratic residues modulo p'. Noting that $\mathbb{Z}_p \subset \mathbb{Z}_{p'}$ and that by squaring an element of $\mathbb{Z}_p$ we get an element of $\mathbb{G}$, we get that treating the secret key k as an element of $\mathbb{Z}_p$ and computing $k^2 \mod p'$ we get an element of $\mathbb{G}$. In this mapping, two elements of $\mathbb{Z}_{p'}$ share an image in $\mathbb{G}$ (one from $\mathbb{Z}_p$ and one from the other half of the group). Hence, in order to recover the correct preimage we assume that the secret key space corresponds to the set of elements from $\mathbb{Z}_p$. This concludes the proof. ■

We now show how to use Π_{GARB} for garbling. Recall first that the garbling requires to encrypt each output key k_w^α, associated with some wire w and value α, twice. So that first ciphertext $c = (c_1, ..., c_\kappa, \Phi)$ is produced with respect to one key k_1 and then c is encrypted under the second key k_2. In order to avoid the blowup in the plaintext size induced by the fact that we use k_2 to encrypt a cipherext of $\kappa + 1$ group elements, we propose to encrypt only the rightmost element Φ. Elements $c_1, ..., c_\kappa$ are given in public as part of the ciphertext anyway. This way k_2 is only used once per ciphertext c. Nevertheless, each plaintext k_w^α is used for encrypting the output wires of the following gate and thus, is comprised of κ group elements. So overall, k_1 and k_2 must be used κ times. This poses a problem since now there are too many constraints on k_1 and k_2 and we cannot hope for statistical security to hold. This implies that secret key k_w^α must be shorter than k_1, k_2.

More formally, considering the garbled circuit level-by-level so that a particular level of the circuit accommodates the maximum number of independent gates (i.e., the gates that can be computed in parallel), then we use a shorter secret key length at each level (associated with the output wires). This ensures that we do not use the input secret keys "too many times". In order to prove statistical security, we extend the statistical argument made for Π_{GARB} using the leftover hash lemma, by showing that it holds for any t ciphertexts as long as t is not too large. More formally,

Proposition 6.2 *Fix $t \geq 1$ and $\kappa = t + 1 + \frac{\lambda + 2\log(1/\epsilon)}{\log p}$. Then for $sk \leftarrow \mathsf{Gen}(1^n)$ it holds that $H_\infty(sk) \geq \log p + 2\log(1/\epsilon) + t \log p$ and*

$$SD\left((c_1, \ldots, c_t), U_{\mathbb{G}^{(\kappa+1)\times t}}\right) < t \cdot \epsilon$$

for $m_1, \ldots, m_t \in \mathcal{M}$ and $c_i \leftarrow \mathsf{Enc}_{sk}(m_i)$ for any i.

Proof: Given $\kappa = t + 1 + \frac{\lambda + 2\log(1/\epsilon)}{\log p}$, we can easily check that $H_\infty(sk) = \kappa \log p \geq \log p + 2\log(1/\epsilon) + t \log p$ holds. The proof now follows by the observation that $\tilde{H}_\infty\left(sk | \left(c_1, \ldots, c_t\right)\right) \geq \log p + 2\log(1/\epsilon)$ since each ciphertext imposes a constraint on sk of $\log p$ bits. This implies that for each i, $SD(c_i, U_{\mathbb{G}^{\kappa+1}}) \leq \epsilon$. Then, the proposition holds using a triangle inequality (cf. Fact 2.2, Item 1) and the leftover hash lemma. ■

We complete the description of our garbling using Π_{GARB}. We recall that each key corresponding to the input wire of a particular gate will be used for encryting output keys twice. In particular, if you assume κ_i to be the key length of the keys corresponding to the input wires of the gates at level i, then each input key (of length κ_i) is used to encrypt $2\kappa_{i+1}$ values for the construction of garbled gate. Therefore in order to ensure statistical indistinguishability, we require to assume $\kappa_i = 2\kappa_{i+1} + 1 + \frac{\lambda + 2\log(1/\epsilon)}{\log p}$. We fix the length of input keys associated with input wires to C (i.e., the first level) to be $\kappa_1 \log p$. For each level $i \geq 2$, the length of the input keys associated with input wires for this level is decreased into $1/3$ of this size, so that the keys for the following level are randomly picked from $\mathbb{G}^{(\kappa_{i-1})/3}$. Implying that $\kappa_1 \geq 3^{\mathsf{Depth}(C)}$. As mentioned before, this degradation is necessary to ensure that statistical indistinguishability holds even if the some key of length κ_{i-1} has been used to encrypt $2\kappa_i$ plaintexts. Taking $\lambda = \kappa_i \log p$, the leakage rate at each level is $(1/3 - o(1))$ of the key length for this level.

6.3 Yao's Two party Protocol in Local Leakage Model

In this section, we adapt the original construction of Yao's two party protocol in the two-party setting into the local leakage model. Our proof is presented in the $\mathcal{F}_{\mathrm{OT}}$-hybrid setting (see Theorem 4.4), where the parties have ideal calls to a trusted party computing $\mathcal{F}_{\mathrm{OT}}$. For simplicity we assume that these ideals calls are made sequentially. Our protocol follows the exact same description verbatim as in [LP09] with the following modifications: First, we use leakage resilient OT (cf. Section 5). Second, we use our SKE Π_{GARB} for garbling (cf. Section 6.2). For keeping the proof simple, we assume the functionality to be deterministic and same-output. This can be easily generalized to all functionalities [Gol04].

Let us begin with the formal description of the protocol.

Protocol 3 (π – Local Leakage Semi-Honest Yao's two-party protocol.)

- **Inputs:** *P_0 has $x_0 \in \{0,1\}^n$ and P_1 has $x_1 \in \{0,1\}^n$ such that $(x_0, x_1) \leftarrow M$, for some k-source sampler M and some specific k. Let $x_0 = x_0^1, \ldots, x_0^n$ and $x_1 = x_1^1, \ldots, x_1^n$.*
- **Auxiliary Input:** *A boolean circuit C such that for every $x_0, x_1 \in \{0,1\}^n$, $C(x, y) = f(x, y)$ where $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$. Furthermore, we assume that C is such that if a circuit-output wire leaves some gate, then the gate has no other wires leading from it into other gates (i.e. no circuit-output wire is also a gate-output wire). Likewise, a circuit-input wire that is also a circuit-output wire enters no gates.*
- **The Protocol:**
  1. *P_0 constructs $G(C)$ as described in Sections 6.1, 6.2 using Π_{GARB}, and sends it to P_1.*

2. *Let $w_1, \ldots, w_n$ be the circuit-input wires corresponding to x_0, and let $w_{n+1}, \ldots, w_{2n}$ be the circuit-input wires corresponding to x_1. Then,*
   (a) *P_0 sends P_1 the keys $k_1^{x_0^1}, \ldots, k_n^{x_0^n}$.*
   (b) *For every i, P_0 and P_1 execute 1-out-of-2 leakage resilient oblivious transfer protocol π_{LOT} (cf. Section 5), in which P_0's input equals (k_{n+i}^0, k_{n+i}^1) and P_1's input equals x_1^i.*
3. *Following the above, P_1 has obtained the garbled circuit and $2n$ keys corresponding to the $2n$ input wires of C. P_1 then evaluates $G(C)$, obtaining $f(x_0, x_1)$. P_1 sends $f(x_0, x_1)$ to P_0 and they both output this value.*

Correctness follows easily from the correctness of $G(C)$. We now formally prove that the above protocol securely computes functionality f in the presence of local leakage by introducing a simulator for each corruption case. In case that P_0 is corrupted, security follows immediately from the security of the OT's. (Namely, no other message is exchanged between the parties other than the OT messages). The more challenging corruption case is when P_1 is corrupted. Similarly to our OT proof in Section 5, we present our proof in two steps. First, we describe a simulator that obtains security in the presence of restricted class of (local) leakage attacks. Then, we show how to extend this proof into the general case of tolerating security in the presence of any PPT leakage function. Specifically, the restricted class of leakage functions are those whose outputs combined with the view of the adversary does not reveal much about the input of the honest party in comparison to what the outputs of the leakage functions alone does. For this case, we design a simulator that generates the view of the adversary by itself and answers the leakage queries of the adversary with the help of trusted third party. One of the main difficulties with it is to design a simulator for which the adversary cannot detect changes in the way the garbled circuit is computed. A typical way to prove security for this case [LP09] is to design a fake garbled circuit that always outputs $f(x_0, x_1)$ irrespective of the input keys set used, and then let the adversary an arbitrary set of keys for the evaluation.

We recall the way the fake garbled circuit $\tilde{G}(C)$ is constructed. At first, the simulator chooses two random keys k_i^0 and k_i^1 for every wire w_i from the circuit C, with the restriction that the least significant bit of the keys is different. Denote the input wires of gate g by w_i and w_j. Then, only a single key (chosen arbitrarily) is encrypted under *all four combinations* of the keys $k_i^0, k_i^1, k_j^0, k_j^1$. The other output key is not encrypted at all. This computation is carried out for all the gates in the circuit. Now, since the adversary can observe the way the simulator prepared the garbled circuit, it can easily detect the changes and conclude that this is a simulated execution. This problem can be dealt using similar techniques considered in Section 5.2.1.

Finally, the important subtlety emerges here is how to extend the proof into the general case so that security is obtained against any PPT leakage function. We resolve this issue by restricting the class of functions that we can compute securely. More specifically, we require that given x_1 and $f(x_0, x_1)$ the existence of an efficient sampler that returns x_0' consistent with x_1, $f(x_0, x_1)$. In this case we say that f is efficiently domain samplable. Our main theorem holds for this restricted class of functions.

Theorem 6.2 (Main) *Let f be a deterministic same-output functionality with logarithmic depth and efficiently domain samplable. Furthermore, assume that π_{LOT} $(\lambda_0^{\text{OT}}, \lambda_1^{\text{OT}})$-securely computes the oblivious transfer functionality in the presence of local leakage. Then, Protocol 3 (λ_0, λ_1)-securely computes f in the presence of local leakage (cf. Definition 3.7), for $\lambda_1 = \lambda_1^{\text{OT}}$ and $\lambda_0 = (1/3 - o(1))$.*

For simplicity our theorem applies for the case that the parties' inputs are picked at uniform. This can be easily generalized for any min-entropy source distribution.

Informally, in case that P_0 is corrupted it leaks from the input of P_1 (which is set as input to the OT's) and the randomness used by P_1 for the OT's. Since the proof is in the local setting the leakage from this randomness can be tolerated for all n OT executions independently. When P_1 is corrupted, it may leak from

all keys that were not revealed during the circuit evaluations, and all internal computations. In order to prove statistical security we show below that P_1 cannot leak more than $1/3$ from each secret key.

Proof: We consider two corruption cases separately. Our proof is shown in the $\mathcal{F}_{\text{OT}}$-hybrid model.

The case of a corrupted P_0. Let $\mathcal{A}$ be an adversary controlling party P_0. Then, construct a simulator $\mathcal{S}_0$ as follows. Note first that P_0's view in an execution of Protocol 3 only consists of the leakage it obtains during the execution since it does not receive any message (apart from the messages within the OT's in the real setting). Thus, simulator $\mathcal{S}_0$ simulates the view of $\mathcal{A}$ by simply playing the role of a trusted party for $\mathcal{F}_{\text{OT}}$. This includes receiving the inputs of $\mathcal{A}$ for the OT ideal calls and the leakage functions to be applied on the honest party's input for the OT's, and answering the leakage queries.

More formally, let f be a function to be applied on input bit x_1^i. Then, $\mathcal{S}_0$ designs a new leakage function f' submitted to the trusted party. f' is hardwired with the code of f so that when applied on input x_1 it ignores all bits but the ith bit. $\mathcal{S}_0$ forwards $\mathcal{A}$ the trusted party's reply.

Clearly, $\mathcal{S}_0$ runs in polynomial-time. Moreover, statistical indistinguishability of the simulated and hybrid views holds trivially for any leakage rate tolerated by the underlying leakage resilient OT construction. This is because $\mathcal{S}_0$ perfectly simulates the leakage given to $\mathcal{A}$, leaking the exact same amount of leakage in the ideal as much as in the hybrid so the entropy loss of x_1 in both cases is identical.

The case of a corrupted P_1. Let $\mathcal{A}$ be an adversary controlling party P_1. Recall first that P_1's view in an execution of Protocol 3 consists of leakage on the input and randomness used by the honest party. More specifically, $\mathcal{A}$ first obtains leakage on the honest party's inputs used for the OT ideal calls. Then, it obtains leakage from the memory parts used to generate the garbled circuit.

We now design a new simulator $\mathcal{S}_1$ that only leaks from the trusted party the replies to the adversary's leakage queries and generates the rest of the view by itself. Recalling that our proof follows in two steps, we begin with a proof of security against a restricted class of leakage functions $\mathcal{H}$ defined by,

$$\Pr[\mathcal{A}\left(\tilde{f}_1,\ldots,\tilde{f}_t,\mathbf{View}_{\pi,\mathcal{A}}(x_0,x_1)\right)=x_0]\leq \Pr[\mathcal{A}\left(\tilde{f}_1,\ldots,\tilde{f}_t\right)=x_0]+\mathsf{negl}(n).$$

for $\mathbf{View}_{\pi,\mathcal{A}}(x_0,x_1)$ the view of $\mathcal{A}$ within π when corrupting party P_1, $\tilde{f}_i$ the outcome of f_i and a negligible function $\mathsf{negl}(\cdot)$. Namely, the probability of guessing x_0 given the leakage is essentially the same when given the adversary's view as well. Then, we prove the following,

Lemma 6.3 *Let f be a deterministic same-output functionality with logarithmic depth and efficiently domain samplable. Furthermore, assume that π_{LOT} $(\lambda_0^{\text{OT}},\lambda_1^{\text{OT}})$-securely computes the oblivious transfer functionality in the presence of local leakage. Then, Protocol 3 $(\lambda_0^{\text{OT}},\lambda_1^{\text{OT}})$-securely computes f in the presence of local leakage (cf. Definition 3.7), for leakage functions taken from $\mathcal{H}$ and $\lambda_0^{\text{OT}}=(1/3-o(1))$.*

Proof: More formally, construct a simulator $\mathcal{S}_1$ as follows.

1. On input x_1, $\mathcal{S}_1$ obtains $f(x_0,x_1)$ from the trusted party and computes potential input x_0'. It then generates a garbled circuit $G(C)$.
2. Let $(k_{n+1}^0,k_{n+1}^1,\ldots,k_{2n}^0,k_{2n}^1)$ denote the keys associated with the input wires of P_1. $\mathcal{S}_1$ receives $\mathcal{A}$'s input for the OT's and hands it the set $(k_{n+1}^{x_1^1},\ldots,k_{2n}^{x_1^n})$.
3. Let $(k_1^1,k_1^n,\ldots,k_n^1,k_n^n)$ denote the keys associated with the input wires of P_0. $\mathcal{S}_1$ sends $\mathcal{A}$ the set of input keys $(k_1^{x_0'^1},\ldots,k_n^{x_0'^n})$ and $G(C)$.

4. $\mathcal{S}_1$ emulates the answers to the leakage queries submitted by $\mathcal{A}$ as follows. Whenever $\mathcal{A}$ submits leakage queries to be applied on the honest party's input to one of the OT's ideal calls, $\mathcal{S}_1$ answers these by applying the leakage function on the specified key from the set $(k_1^{1-x_0'^1}, \ldots, k_n^{1-x_0'^n})$.

   whenever $\mathcal{A}$ submits a leakage query f to be applied on the honest party's input and randomness used for generating $G(C)$, $\mathcal{S}_1$ constructs a new function f' submitted to the trusted party as follows. f' is hardwired with the code of f and appropriate randomness used by $\mathcal{S}_1$ for generating $G(C)$.

5. $\mathcal{S}_1$ hands $\mathcal{A}$ this leakage and outputs whatever $\mathcal{A}$ does.

This concludes the description of $\mathcal{S}_1$. Clearly, $\mathcal{S}_1$ runs in polynomial-time and

$$\left(\mathbf{View}^{\mathrm{sm}}_{\pi,\mathcal{A}}(x_0,x_1)\right)_{(x_0,x_1)\leftarrow M} \approx_s \left(\mathbf{View}^{\mathrm{Hyb}}_{\pi,\mathcal{A}}(x_0,x_1)\right)_{(x_0,x_1)\leftarrow M}$$

where the left hand side view denotes the adversary view in the simulated execution whereas, the right hand side view is its view in the hybrid execution. Before proceeding with the proof we sort the gates $g_1, \ldots, g_{|C|}$ of circuite C in a *topological order*: if the input wires of a gate g_ℓ come from gates g_i and g_j, then $i \leq \ell$ and $j \leq \ell$.

Our proof follows by induction. For any $1 \leq i \leq |C|$, assume that the simulated and hybrid views, generated for gates $g_1, \ldots, g_i$, are statistically close to uniform (and thus statistically close). We then prove that the remaining (average) min-entropy of the inactive output key $k_o'^i$, that is associated with the output wire of gate g_i is greater than or equal to $\log p + 2\log(1/\epsilon)$. This will allow us to claim that the garbled table associated with gate g_i is statistically close in both executions.

We continue with our inductive argument and prove first that for $i = 1$ the simulated and hybrid views are statistically close. Note that this view of the adversary includes the following: outputs from OT's ideal calls $(k_1', \ldots, k_n')$, input keys for P_0's input $(k_1'', \ldots, k_n'')$, the garbling $\tilde{g}_1$ of the first gate g_1 and leakage on the following keys: (1) $(\tilde{k}_1'', \ldots, \tilde{k}_n'')$ inactive keys associated with input wires of P_0's input,(2) $(\tilde{k}_1', \ldots, \tilde{k}_n')$ inactive keys associated with input wires of P_0's input and (3) k_o' the inactive output key associated with output wire leaving gate g_1. We elaborate on the leakage parameters. Recall that the length of the inactive keys for the first level is $\kappa_1 \log p$ and that the length of the inactive keys for the second level is $\kappa_2 \log p$, where κ_1 and κ_2 are related as described in section 6.2. In order to apply a statistical argument the leakage rates from all keys must be less than one third the length of the specified key.

Assume w.l.o.g. that inactive keys $(\tilde{k}_1', \tilde{k}_2')$ are associated with the input wires of g_1. Then we claim that the remaining entropy of each of these keys given the leakage and the adversary's view is at least $\log p + 2\log(1/\epsilon)$. This is justified as follows. Recall first that the garbling of g_1 includes three sets of ciphertexts $(\bar{c}_1, \bar{c}_2, \bar{c}_3)$ such that each set is computed with at least one of these inactive input keys. Focusing on $\tilde{k}_1'$ of length $\kappa_1 \log p$ (the argument for $\tilde{k}_2'$ follows similarly), it holds that $\tilde{k}_1'$ is only used to compute $2\kappa_2 = 2\kappa_1/3$ ciphertexts (namely, $\tilde{k}_1'$ is used for computing the inactive/active output keys for g_1 twice, such that the length of these output keys is $\kappa_1/3$).

Let $\kappa_1 = 2\kappa_2 + 1 + \frac{\lambda + 2\log(1/\epsilon)}{\log p}$ as specified in Section 6.2. By Lemma 2.3 Item 2, it holds that

$$\tilde{H}_\infty(\tilde{k}_1'|\bar{c}_1, \bar{c}_2, \bar{c}_3, f) \geq \tilde{H}_\infty(\tilde{k}_1') - 2\kappa_2 \log p - \lambda \geq \log p + 2\log(1/\epsilon)$$

for f the leakage from $\tilde{k}_1'$ computed by leakage functions with accumulated leakage rate λ (cf. Definition 3.5). By Proposition 6.2 it holds that $(\bar{c}_1, \bar{c}_2) \approx_s (U_{\mathbb{G}^{((\kappa+1)\times\kappa_1}}, U_{\mathbb{G}^{((\kappa+1)\times\kappa_1}})$, assuming w.l.o.g. that $\tilde{k}_1'$ encrypts $\bar{c}_1$ and $\bar{c}_2$. This implies that the ciphertexts in the view of $\mathcal{A}$ are statistically close to uniform. Finally, since the rest of the view is generated uniformly, the entire view is statistically close to uniform.

Assume that for some i, the view generated with garbling for gates $g_1, \ldots, g_i$ is statistically close to uniform. We prove our claim for $i + 1$. This proof follows similarly to the proof in the base case. Loosely

speaking, we first claim that the remaining entropy of each inactive key $\tilde{k}_1'^i$ associated with input wire of gate g_i is at least $\log p + 2\log(1/\epsilon)$. This is derived by the fact that view for i is independent of $\tilde{k}_1'^i$ by the induction assumption. The above implies that the ciphertexts computed by $\tilde{k}_1'^i$ are statistically close to uniform. ■

We now discuss how to extend our proof for the general case of security against any leakage function. Our proof follows by a sequence of hybrid games. More specifically, we recall first that simulator $\mathcal{S}_1$ defined above uses input x_0' for the simulated execution with the fake garbled circuit and let x_0 be the input of P_0 for the real execution. Then, consider a hybrid input $\tilde{x}_0^i$ defined as follows. The first i bits of $\tilde{x}_0^i$ are taken from the prefix of x_0' of length i (i.e., bits $x_0'^1, \ldots, x_0'^i$) whereas, bits $i+1, \ldots, n$ are taken from the suffix of x_0 of length $n-i$ (i.e., bits $x_0^{i+1}, \ldots, x_0^n$).

We claim that if there exists a gap in guessing the input of the honest party's between the simulated and hybrid executions, then this gap can be reduced into a bit level. More specifically, define a hybrid game Game_i as follows. Given x_0, define simulator $\mathcal{S}_1^i$ that assembles $\tilde{x}_0^i$ as specified above. It then builds a garbled circuit using input $\tilde{x}_0^i$. Define a simulator $\mathcal{S}_1^i$ that uses $\tilde{x}_0^i$ as input and emulates the role of the honest P_0 on this input. Clearly, the only difference between every two consecutive games is the input bit used for location $i+1$.

We now claim that for every leakage function f applied on the secret keys k, k', associated with the ith bit of $\tilde{x}_0^{i+1}$ there are two possible cases. Either $f(k) = f(k')$ or not. The analysis follows similarly to the analysis in Section 5 since given the output of f, $\tilde{f}$, the adversary is able to compare it with the input key, associated with the ith input bit of P_0, from its view and concludes whether this bit is zero or one.

■

Acknowledgments. We wish to thank Krzysztof Pietrzak for the observation that hardness of k-DDH assumption implies the hardness of k-IND assumption.

References

[ADN+10] Joël Alwen, Yevgeniy Dodis, Moni Naor, Gil Segev, Shabsi Walfish, and Daniel Wichs. Public-key encryption in the bounded-retrieval model. In *EUROCRYPT*, pages 113–134, 2010.

[ADW09] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *CRYPTO*, pages 36–54, 2009.

[AGH] Adi Akavia, Shafi Goldwasser, and Carmit Hazay. Distributed public key schemes secure against continual leakage. In *Manuscript*.

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC*, pages 474–495, 2009.

[BCH11] Nir Bitansky, Ran Canetti, and Shai Halevi. Leakage tolerant interactive protocols. Cryptology ePrint Archive, Report 2011/204, 2011. http://eprint.iacr.org/.

[BDL01] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of eliminating errors in cryptographic computations. *J. Cryptology*, 14(2):101–119, 2001.

[BG10] Zvika Brakerski and Shafi Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In *CRYPTO*, pages 1–20, 2010.

[BKKV10] Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *FOCS*, pages 501–510, 2010.

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

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In *CRYPTO*, pages 513–525, 1997.

[BSW11] Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. 2011.

[Can97] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *CRYPTO*, pages 455–469, 1997.

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

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, Las Vegas, Nevada, October 2001.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *STOC*, pages 11–19, 1988.

[DF12] Stefan Dziembowski and Sebastian Faust. Leakage-resilient cryptography from the inner-product extractor. In *EUROCRYPT*, 2012.

[DGK+10] Yevgeniy Dodis, Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In *TCC*, pages 361–381, 2010.

[DHLAW10a] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *FOCS*, pages 511–520, 2010.

[DHLAW10b] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In *ASIACRYPT*, pages 613–631, 2010.

[DKL09] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In *STOC*, pages 621–630, 2009.

[DLWW11] Yevgeniy Dodis, Allison Lewko, Brent Waters, and Daniel Wichs. Storing secrets on continually leaky devices. In *To Apear FOCS*, 2011.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302, 2008.

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *EUROCRYPT*, pages 523–540, 2004.

[EGL85] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

[FKPR10] Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, and Guy N. Rothblum. Leakage-resilient signatures. In *TCC*, pages 343–360, 2010.

[GJS11] Sanjam Garg, Abhishek Jain, and Amit Sahai. Leakage-resilient zero knowledge. In *CRYPTO*, 2011.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-time programs. In *CRYPTO*, pages 39–56, 2008.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In *CHES*, number Generators, pages 251–261, 2001.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

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

[HL11] Shai Halevi and Huijia Lin. After-the-fact leakage in public-key encryption. In *TCC*, pages 107–124, 2011.

[HSH+09] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: cold-boot attacks on encryption keys. *Commun. ACM*, 52(5):91–98, 2009.

[IPS09] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Secure arithmetic computation with no honest majority. In *TCC*, pages 294–314, 2009.

[JS07] S. Jarecki and V. Shmatikov. Efficient two-party secure computation on committed inputs. *EUROCRYPT*, 4515:97–114, 2007.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO*, pages 388–397, 1999.

[Koc96] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *CRYPTO*, pages 104–113, 1996.

[Kus92] Eyal Kushilevitz. Privacy and communication complexity. *SIAM J. Discrete Math.*, 5(2):273–284, 1992.

[KV09] Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In *ASIACRYPT*, pages 703–720, 2009.

[Lab] Reliable Computing Laboratory. In *Boston University, Side channel attacks database, http://www.sidechannelattacks.com.*

[LLW11] Allison Lewko, Mark Lewko, and Brent Waters. How to leak on key updates. In *STOC*, 2011.

[LP09] Y. Lindell and B. Pinkas. A proof of security of yaos protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, 2009.

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

[LRW11] Allison B. Lewko, Yannis Rouselakis, and Brent Waters. Achieving leakage resilience through dual system encryption. In *TCC*, pages 70–88, 2011.

[MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In *TCC*, pages 278–296, 2004.

[MTVY11] Tal Malkin, Isamu Teranishi, Yevgeniy Vahlis, and Moti Yung. Signatures resilient to continual leakage on memory and computation. In *TCC*, pages 89–106, 2011.

[NP05] Moni Naor and Benny Pinkas. Computationally secure oblivious transfer. *J. Cryptology*, 18(1):1–35, 2005.

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

[NS09] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In *CRYPTO*, pages 18–35, 2009.

[Oka92] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *CRYPTO*, pages 31–53, 1992.

[Pie09] Krzysztof Pietrzak. A leakage-resilient mode of operation. In *EUROCRYPT*, pages 462–482, 2009.

[PSP+08] Christophe Petit, François-Xavier Standaert, Olivier Pereira, Tal Malkin, and Moti Yung. A block cipher based pseudo random number generator secure against side-channel key recovery. In *ASIACCS*, pages 56–65, 2008.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, pages 554–571, 2008.

[QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Measures and countermeasures for smart cards. In *E-smart*, pages 200–210, 2001.

[Rab81] M. Rabin. How to exchange secrets by oblivious transfer. Tech. Memo TR-81, Aiken Computation Laboratory, Harvard U., 1981.

[Riv97] Ronald L. Rivest. All-or-nothing encryption and the package transform. In *FSE*, pages 210–218, 1997.

[SV03] Amit Sahai and Salil P. Vadhan. A complete problem for statistical zero knowledge. *J. ACM*, 50(2):196–249, 2003.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164, 1982.