SOTERIA Proof

A SOTERIA Security Proof

We now discuss the privacy-preserving security of the SOTERIA protocol. The goal is to reduce the security of our system to the security of the underlying security mechanisms, namely the isolation guarantees of Intel SGX and the bootstrapped secure channels, and the indistinguishability properties of encryption.

The security goal consists in demonstrating that SOTERIA ensures privacy-preserving machine learning. Concretely, this means that the behavior displayed by SOTERIA in the real-world is indistinguishable from the one displayed by an idealized functionality in the ideal-world, that simply computes over the task script and provides an output via secure channel. The only information revealed during this process is the length of I/O, the number of computation steps, and the access patterns to the external storage where data is kept

Formally, this security goal is defined using the real-versus-ideal world paradigm, similarly to the Universal Composability [2] framework.

We begin with a more formal description of our security model. Then, we present an intermediate result for ensuring the security of enclaves relying on external storage. We can finally specify the behavior of the Client, Master and Workers, and present the full proof.

A.1 Formal Security Model

Our model considers external environment $\mathcal{Z}$ and internal adversary $\mathcal{A}$. Π denotes the protocol running in the real world, and $\mathcal{S}$ and $\mathcal{F}$ denote the simulator and functionality, respectively, running in the ideal-world. The real-world considers a Client C, a Master node M and 2 Worker nodes W_1 and W_2. This is for simplicity, as the definition and proof can be easily generalized to consider any number of Worker nodes. We also consider a global storage G, which is initialized by $\mathcal{Z}$ before starting the protocol. The Ideal functionality is parametrised by this external storage $\mathcal{F}$<G>, and will reveal the access patterns via leakage function $\mathcal{L}$.[1]

In the real-world, $\mathcal{Z}$ begins by providing public inputs to C in the form of (s, m), where s is the task script, and m is the manifest detailing data in G to be retrieved.[2] The Client will then execute protocol Π, sending messages to M, W_1 and W_2. When the script is concluded, output is provided to C, finally being returned to $\mathcal{Z}$. $\mathcal{A}$ can observe all communication between C, M, W_1, W_2 and G.

In the ideal world, (s, m) are provided to dummy Client C, which in turn forwards them to $\mathcal{F}$<G>. The functionality will simply run the protocol and forward the output to C, which in turn is returned to $\mathcal{Z}$. All the communication observed by $\mathcal{A}$ must be emulated by simulator $\mathcal{S}$, which receives (s, m), leakage $\mathcal{L}$ produced from the functionality interaction with storage G, and the size of the output.

Security is predicated on ensuring that $\mathcal{S}$ does not require any sensitive information (contained in G) to emulate the communication to $\mathcal{A}$. Given that we consider a semi-honest adversary, we can simplify the interaction with the system and instead discuss equality of views, as $\mathcal{Z}$ and $\mathcal{A}$ are unable to deviate the system from its expected execution. This is captured by the following definition.

Definition 1 *Let* Real *denote the view of* $\mathcal{Z}$ *in the real-world, and let* Ideal *denote the view of* $\mathcal{Z}$ *in the ideal-world. Protocol* Π *securely realises* $\mathcal{F}$ *for storage* G *if, for all environments* $\mathcal{Z}$ *and all adversaries* $\mathcal{A}$*,*

$$\mathsf{Real}_{\mathcal{Z},\mathcal{A},\Pi}(G) \approx \mathsf{Ideal}_{\mathcal{Z},\mathcal{A},\mathcal{S},\mathcal{F}}(G)$$

A.2 Intermediate Result

For convenience, SOTERIA does not require the Client to provide input data at the time of the ML processing, and instead the Workers are given access to an external storage from which to retrieve this data. When discussing the security in the context of secure outsourced computation for SGX, this is functionally equivalent to classical scenarios where the Client provides these inputs via secure channel (Theorem 3

[1] Reasoning for the security of SML-2 instead would only require this function to also reveal statistical data to the simulator, which we consider to be non-sensitive.

[2] SOTERIA Clients are trusted. As such, we assume (s, m) to both be *valid*, in the sense that they are correct ML scripts and data sets in G, and thus can be interpreted by ideal functionality $\mathcal{F}$.

Algorithm Setup(i, m)<G>:
$\mathsf{k} \leftarrow\$\ \Theta.\mathsf{Gen}()$
$\mathsf{c} \leftarrow\$\ \Theta.\mathsf{Enc}(\mathsf{k}, i)$
$G[m] \leftarrow \mathsf{c}$
Return (m, k)

Figure 1: Secure external storage setup.

in [1]). The reasoning is simply that, if a protocol securely realises a functionality with a given input provided via secure channel, then the same functionality can be securely realised with the same input fixed in an external storage, securely accessed by the enclave.

Consider a protocol Π_1 that securely realises some functionality $\mathcal{F}$ with simulator $\mathcal{S}_1$ according to Theorem 3 of [2]. We construct protocol Π_2 built on top of this secure protocol Π_1, where input data is pre-established and provided to the enclave via an initial Setup stage where inputs are stored in encrypted fashion (Figure 1 describes a simplified version of the process for a single entry). Inputs to Π_2 are exactly the same as those for Π_1, but instead of being transmitted via the secure channel established with Attested Computation, they are retrieved from storage using a key sent via the same channel. The Client-server communication increases by a constant (the key length), which can be trivially simulated, and the rest of the input can be simulated in a similar way using the IND-CPA properties of Θ. This protocol behavior will be key for all Soteria Workers. Our theorem is as follows.

Theorem 1 *Let Π_1 be a protocol that securely realises functionality $\mathcal{F}$ according to Theorem 3 in [1]. Then Π_2, constructed as discussed above, securely realises $\mathcal{F}$ according to Definition 1.*

Proof. To demonstrate this result, we construct simulator $\mathcal{S}_2$ using $\mathcal{S}_1$, then argue that, given that $\mathcal{S}_1$ is a valid simulator for the view of Π_1, then the simulated view must be indistinguishable from the one of the real world of Π_2.

We begin by deconstructing $\mathcal{S}_1$ in two parts: $\mathcal{S}_1.\mathsf{AC}()$ will produce the view for the establishment of secure channel, while $\mathcal{S}_1.\mathsf{Send}(l)$ will produce a simulated view of Client inputs, given their length. In turn, our simulator will share the same functions, but also include a third $\mathcal{S}_2.\mathsf{Get}(l)$ to simulate information being retrieved from G, given its length. Our simulator is depicted in Figure 2.

The view presented to $\mathcal{A}$ is composed of three different types of messages:

- Messages exchanged during the secure channel establishment are exactly the same as in Π_1. Thus they remain indistinguishable from Π_2.

Algorithm AC()	**Algorithm** Send(l)	**Algorithm** Get(l)
$\mathsf{k} \leftarrow\$\ \Theta.\mathsf{Gen}()$	Return $\mathcal{S}_1.\mathsf{Send}(l)$	$i \leftarrow \{0\}^l$
Return $\mathcal{S}_1.\mathsf{AC}()$		$\mathsf{c} \leftarrow\$\ \Theta.\mathsf{Enc}(\mathsf{k}, i)$
		Return c

Figure 2: Simulator for Π_2.

- Outputs received via the secure channel follow the exact same simulation strategy than Π_1, and thus are indistinguishable from Π_2.
- Messages produced from G in Π_2 are encryption of data in $G[m]$, while the values presented by $\mathcal{S}_2$ are dummy encryptions with the same length. We can thus reduce the advantage of $\mathcal{A}$ to distinguish these views to the advantage of the same adversary to attack the IND-CPA guarantees of encryption scheme Θ, which is negligible.

As such, if $\mathcal{S}_1$ is a valid simulator for Π_1 to $\mathcal{A}$, then the view presented by $\mathcal{S}_2$ must also be indistinguishable for Π_2 to $\mathcal{A}$.

Let

$$\mathsf{Adv}^{\mathsf{Dist}}{}_{\mathcal{Z},\mathcal{A},\Pi,\mathcal{S},\mathcal{F}}(G) = \\ |\Pr[\mathsf{Real}^{G}_{\mathcal{Z},\mathcal{A},\Pi_2} \Rightarrow \mathsf{T}] - \Pr[\mathsf{Ideal}^{G}_{\mathcal{Z},\mathcal{A},\mathcal{S}_2,\mathcal{F}} \Rightarrow \mathsf{T}]|$$

To conclude, we have that, for negligible function μ,

$$\mathsf{Adv}^{\mathsf{Dist}}_{\mathcal{Z},\mathcal{A},\Pi_2,\mathcal{S}_2,\mathcal{F}}(G) = \mathsf{Adv}^{\mathsf{Dist}}_{\mathcal{Z},\mathcal{A},\Pi_1,\mathcal{S}_1,\mathcal{F}}() + \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\Theta,\mathcal{A}}() \\ \leq \mu()$$

and Theorem 1 follows.

A.3 Soteria Client, Master and Workers

The Soteria components follow standard methodologies for ensuring secure outsourced computation using SGX. As such, and given the complexity of ML tasks described in the script, we consider the following set of functions.

Secure channels are established with enclaves. We define $\mathsf{init}(P)$ as the bootstrapping process, establishing a channel with participant P. This produces an object that can be used to send and receive data via send and $\mathsf{receive}$. Untrusted storage is not protected with secure channels, and can be accessed using the call $\mathsf{uGet}(G, m)$, which retrieves data from G considering manifest file m. Concretely, this is achieved using the open-source library Graphene-SGX, which we assume to correctly implement this mechanism. Finally, the script s defines the actual computation that must be performed by the system, and will be

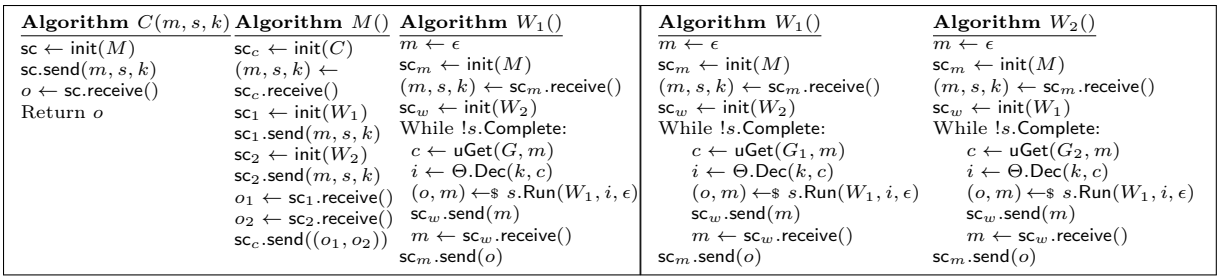

Figure 3: SOTERIA Components. Client C (left), Master node M (middle) and Worker node 1 W (right).

Figure 4: SOTERIA Workers with split storage.

executed collaboratively with both Workers. As such, we define s as a stateful object with the main method $\mathsf{Run}(\mathsf{id}, i_1, i_2)$, where input id is the identifier of the Worker, i_1 is input from storage and i_2 is intermediate input (e.g. model parameters), returning (o_1, o_2), where o_1 is the (possibly) final output, and o_2 is the (optional) intermediate output for dissemination. For simplicity, we also define method $\mathsf{Complete}$ that returns T if the task is complete, or F otherwise.

The SOTERIA components can be analysed in Figure 3 and are as follows. The Client C (left of Figure 3) simply establishes the channel with M, sends the parameters (manifest file, task script and storage key), and awaits computation output. Observe that we assume that the key k has been previously initialized, and that the actual data has been previously encrypted in G using it. The Master M (middle of Figure 3) will receive the parameters from C and establish channels with W_1 and W_2, forwarding them the same parameters and awaiting computation output. When it arrives, it is forwarded to the Client.[3] Worker W_1 (right of Figure 3) receives the parameters from M and starts processing the script: retrieves encrypted data from G, decrypts, processes and exchanges intermediate results with the other Worker. When the script is concluded, it returns its output to M. The behavior of W_2 is the same, but connection is established instead with W_1.

A.4 Full Proof

Given the description of SOTERIA components in Figure 3, the SOTERIA protocol Π_{xyz} is straightforward to describe. Considering a pre-encrypted storage G, the Client C, Master M and Workers W_1, W_2 execute following their respective specifications. Our theorem for the security of SOTERIA is as follows.

Theorem 2 Π_{xyz}*, assuming the setup of Figure 1 and constructed as discussed above, securely realises* $\mathcal{F}$ *according to Definition 1.*

The proof is presented as a sequence of four games. We begin in the real-world, and sequentially adapt our setting until we arrive in the ideal world. We then argue that all steps up to that point are of negligible advantage to $\mathcal{A}$, and thus the views must be indistinguishable to $\mathcal{Z}$.

The first is a simplification step, where, instead of using a single storage G, we slice the storage to consider G_1 and G_2. Figure 4 represents this change. This enables us to split the execution environment of W_1 and W_2 seamlessly, and can be done trivially since manifest file m by construction will never require different Workers to access the same parts of G. Since these two games are functionally equal, the adversarial advantage is exactly 0.

The second step is a hybrid argument, where we sequentially replace both Workers by ideal functionalities performing partial steps of the ML script. Concretely, we argue as follows. Replace W_1 with a functionality for its part of the ML script $\mathcal{F}_{W1}$, according to Definition 1. From Theorem 1 we can establish that this adaptation entails negligible advantage to $\mathcal{A}$ provided that the protocol without external access realises the same functionality. However this is necessarily the case, as it follows the exact structure as the constructions in [1]. We can repeat this process for W_2.[4] As such, using the intermediate result, we can thus upper bound the advantage adversary to distinguish these two scenarios by applying twice the result of Theorem 1.

The third step replaces the Master by an ideal functionality $\mathcal{F}_M$ that simply forwards requests to the Worker functionalities. This one follows the same logic as the previous one, without requiring the ex-

[3] In the actual protocol, the Master has additional steps to process the output. We describe it like this for simplicity, as it does not change the proof.

[4] Again, this technique extends for an arbitrary number of Workers. N number of Workers would just require us to adapt the multiplication factor in the final formula, which would still be negligible.

ternal storage, as the protocol also follows the exact structure as the constructions in [1].

In the final step, we have 3 functionalities $(\mathcal{F}_M, \mathcal{F}_{W1}, \mathcal{F}_{W2})$ playing the roles of (M, W_1, W_2), respectively. We finalize by combining them to a single functionality $\mathcal{F}$ for ML script processing. This can be done by constructing a big simulator S that builds upon the simulators for the individual components (S_M, S_{W1}, S_{W2}). The simulator S behaves as follows:

- Run S_M to construct the communication trace that emulates the first part of $\mathcal{F}$.
- Run the initial step of S_{W1} and S_{W2} to construct the communication trace for establishing secure channels between Workers and Master.
- Call leakage function $\mathcal{L}$ to retrieve the access patterns to G. Use the result to infer which part of the storage is being accessed, and run S_{W1} or S_{W2} to emulate the computation stage.

Given that the view produced by S is exactly the same as the one provided by the combination of S_M, S_{W1} and S_{W2}, the adversarial advantage is exactly 0.

We are now exactly in the ideal world specified for Definition 1.

Let

$$\mathsf{Adv}^{\mathsf{Dist}}{}_{\mathcal{Z},\mathcal{A},\Pi,\mathcal{S},\mathcal{F}}(G) = \\ |\Pr[\mathsf{Real}^{G}_{\mathcal{Z},\mathcal{A},\Pi_{\mathsf{xyz}}} \Rightarrow \mathsf{T}] - \Pr[\mathsf{Ideal}^{G}_{\mathcal{Z},\mathcal{A},\mathcal{S},\mathcal{F}} \Rightarrow \mathsf{T}]|$$

To conclude, we have that, for negligible function μ,

$$\begin{aligned}\mathsf{Adv}^{\mathsf{Dist}}_{\mathcal{Z},\mathcal{A},\Pi,\mathcal{S},\mathcal{F}}(G) &= 2 \cdot \mathsf{Adv}^{\mathsf{Dist}}_{\mathcal{Z},\mathcal{A},\Pi_{W1},\mathcal{S}_{W1},\mathcal{F}_{W1}}(G) \\ &\quad + \mathsf{Adv}^{\mathsf{Dist}}_{\mathcal{Z},\mathcal{A},\Pi_M,\mathcal{S}_M,\mathcal{F}_M}() \\ &\leq \mu()\end{aligned}$$

and Theorem 2 follows.

B ML Workflow attacks

This section presents the attacks in Section 3.1 in further detail, and argues in which circumstances Soteria is secure against each attack. First, we will describe a general adversarial model against Soteria that follows the security restrictions justified in Appendix A. Then, we will present an experiment that captures what constitutes a valid attack under each definition, as described in Section 3.1. For each attack, we consider our protocol to be secure if we can demonstrate that one cannot rely on a valid adversary against the experiment of said attack under the constraints of Soteria. In some instances, this will depend on known attack limitations, which we detail case-by-case.

Game $\mathsf{AdvXYZ}_{\mathcal{A},\Pi_{\mathsf{xyz}}}(G, s)$:
$(G') \leftarrow\$\ \mathcal{A}_1(G, s)$
$(m, l) \leftarrow \Pi_{\mathsf{xyz}}(G', s)$
$r \leftarrow\$\ \mathcal{A}_2^m(l)$
Return $\mathsf{Success}(G, s, m, r)$

Figure 5: Adversary interacting with Soteria.
ff

B.1 Attacker against Soteria

Our goal is to present a model that details the conditions in which these attacks are possible. As such, it must be both generic, to capture the multiple success conditions of attacks, as well as expressive, so that it can be easy to relate to each specific attack.

In this definition, we will also consider an adversary that can play the role of an honest client, and thus will have black-box access to the produced model. We stress that, in practice, this will not be the case in many circumstances. In those scenarios, since queries to the model are made via secure channel, an external adversary is unable to arbitrarily request queries to the model without causing it to abort. This means that any attack that requires black-box access to the model is not possible if Soteria assumes external adversaries.

Let Π_{xyz} denote the full training protocol of Soteria. It receives external storage G and task script s as inputs, and produces a model m, which can then be queried. Based on the security result of Appendix A, the interaction of an adversary with our system can be described in Figure 5. The adversary $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2\}$ can first try and manipulate the input dataset G to G'. This is then used for Π_{xyz}, which will produce the model m and the additional leakage l (SML-1 has no additional leakage, so $l = \epsilon$). Finally, the adversary can interact black-box with the model until a conclusion r is produced. This will be provided to a $\mathsf{Success}$ predicate, which will state if the attack was successful. This predicate is specific to the attack, and allows us to generally describe attacks such as *adversarial samples*, where the goal is to make the resulting model deviate, as well as *membership inference*, where the goal is to retrieve information from the original dataset.

Remark Observe that $\mathcal{A}_1$ and $\mathcal{A}_2$ do not share state. This is because they play different roles within this experiment: the first influences the system by at-

Game $\mathsf{DSetMan}_{\mathcal{A},\Pi}(G, s)$:
$G' \leftarrow\$\ \mathcal{A}(G, s)$
$m \leftarrow \Pi(G', s)$
Return $\mathsf{Success}(G, s, m)$

Figure 6: Model for dataset manipulation attack.

tempting to manipulate the training dataset G, while the second interacts with the model m and leakage l to try and extract information. Indeed, our first step will be to show that $\mathcal{A}_1$ is unable to rely on G' to meaningfully convey any additional knowledge gained by observing (G, s).

B.2 Dataset manipulation

Dataset manipulation attacks are defined by an adversary with the capability of inserting, removing or manipulating dataset information. These align with the setting considered for attacks via *adversarial samples*. Figure 6 is an experiment that describes what constitutes a successful attack for dataset manipulation. The adversary $\mathcal{A}$ is given full knowledge of G[5], and must produce an alternative input dataset G'. We then train the model (protocol Π) over that data to produce model m, and the adversary is successful if said model satisfies some attack success criteria $\mathsf{T/F} \leftarrow \mathsf{Success}$.

We now argue that the integrity guarantees of the authenticated encryption used by our external storage G ensure that these attacks do not occur for Π_{xyz}. We do this by showing that any adversary that performs an *adversarial samples* attack on Π_{xyz} can be used to construct a successful attack on the the security of the authenticated encryption scheme. First, observe that no attack can be successful if the adversary makes no changes on the input dataset, so if $G = G'$ then $\mathsf{F} \leftarrow \mathsf{Success}$. Furthermore, if Π_{xyz} aborts, then no model is produced, so it naturally follows that the attack is unsuccessful $\mathsf{F} \leftarrow \mathsf{Success}$.[6]

As such, the only cases in which $\mathsf{T} \leftarrow \mathsf{Success}$ are those in which $G' \neq G$ and Π_{xyz} does not abort. But this means that the adversary was able to forge an input that correctly decrypts, breaking the integrity of the underlying encryption scheme. Since the security guarantees of authenticated encryption ensure that the probability of existing such an adversary is negligible, the probability of such an attacker in Soteria will also been negligible.

[5]Realistically, an attacker would have less information, but for our purposes we can go for the worst case and give him all the information regarding the computation and its input.

[6]The only circumstance in which this could be considered a successful attack was if the goal was to perform a denial-of-service attack, which we consider to fall outside the scope of an adversarial sample attack.

B.3 Black-box Attacks

All the remaining attacks with the exception of some *reconstruction attacks* follow a similar setting, where the adversary leverages a black-box access to the trained model, depicted in the experiment of Figure 7. We begin by running Π to produce our model and leakage, and then run an additional procedure $\mathsf{Extract}$ to obtain additional information from the original dataset, which cannot be retrieved by simply querying the model. This procedure captures whatever knowledge regarding the underlying ML training might be necessary for the attack to be successful (e.g. information about data features). We then provide this additional information to the adversary, and give it black-box access to the model. The success criteria depends on the specific attack, and is validated with respect to the original dataset, model, and the task script being run. E.g. for *model extraction* attack, the goal might be to present a model m' that is similar to m, evaluated by the $\mathsf{Success}$ predicate.

For simplicity, we first exclude all attacks for an external adversary, which does not have black-box access to the model of Soteria. This is true if we can show that one cannot emulate black-box access to the model using confidence values and class probabilities. Albeit an interesting research topic, current attacks are still unable to do this in an efficient way [3]. We now go case-by-case, assuming an adversary can play the role of a genuine client to our system.

Our arguments for Π_{xyz} depend on being able to rely on a successful adversary $\mathcal{A}$ of Figure 7 to perform the same attack in Figure 5. As such, the security of our system will depend on the amount of additional information z, on how it can be extracted from the view of the adversary of Soteria.

- For *membership inference*, *reconstruction attacks* based on black-box access to the model, *model inversion*, and *model extraction* via *data-free knowledge distillation*, no additional information z is required. This means that any successful adversary in Figure 7 will also be successful in Figure 5, meaning that both for SML-1 and SML-2 are vulnerable. Preventing these attacks requires restricting access to the model to untrusted participants.
- *Class-only* attacks for *model extraction* require additional knowledge from the dataset. Specifically, z must contain concrete training dataset samples. This means that, to leverage such an adversary $\mathcal{A}$, one must first be able to use $\mathcal{A}_2^m(l)$

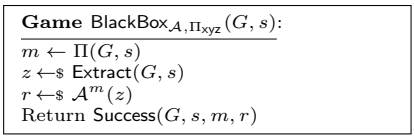

Figure 7: Model for black-box attacks.

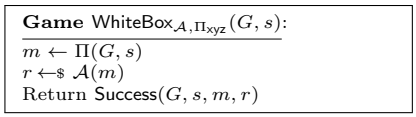

Figure 8: Model for white-box attacks.

to extract such a z. This exactly matches the setting of *model inversion* attacks. This means that SOTERIA is vulnerable to *class-only* attacks for *model extraction* in SML-1 or SML-2 if there is also an efficient attack for *model inversion* in SML-1 or SML-2, respectively.

- *Equation-solving model extraction* requires knowledge of the dimension of the training dataset G. This is additional information z that is not revealed by querying the model, which means no adversary $\mathcal{A}_2^m(\epsilon)$ can retrieve z, and thus SML-1 is secure against said attacks. However, combining public data with confidence values might allow for $\mathcal{A}_2^m(l)$ to extract a sufficient z to perform the attack, which makes SML-2 vulnerable to *equation-solving model extraction*.

- *Path-finding model extraction* attacks require information regarding leaf count, tree depth and leaf ID. As such, all this must be encapsulated in z. [5] suggests that such information is not retrievable from only black-box access to the model [5], which means no adversary $\mathcal{A}_2^m(\epsilon)$ can produce z and thus SML-1 is secure. However, this is information that can be extracted from confidence values, which suggests that an efficient adversary $z \leftarrow\$ \mathcal{A}_2^m(l)$ is likely to exist, and thus SML-2 is vulnerable to such attacks under these assumptions.

We can generalize the security of our system to these types of attacks as follows. If no additional information z is required, then Π_{xyz} is vulnerable to an adversary that can play the role of an honest client. If z can be extracted from black-box access to the model, then we can still rely on said adversary to attack Π_{xyz}. Otherwise, SML-1 is secure, as no additional information is leaked. Furthermore, the security of SML-2 will depend whether one can infer z from l and from the black-box access to the model. Concretely, if we can show that no (efficient) function F exists, such that $z \leftarrow\$ \mathsf{F}^m(l)$, then Π_{xyz} for leakage l is secure against attacks requiring additional data z.

B.4 White-box Attacks

White-box attacks capture a scenario where an adversary requires white-box access to the model. These align with the setting of *reconstruction attacks* that explicitly requires white-box access to the model. Figure 8 is an experiment that describes what constitutes a successful *reconstruction attack* in this context. We begin by training the model (protocol Π) over the original dataset to produce the model. We then provide the trained model directly to the adversary, which will reconstruct raw data r. Finally, the success of the attack is validated with respect to the original dataset.

We now argue that these attacks do not occur for Π_{xyz}, as long as it is not possible to extract the model from the confidence values and from black-box access to the model. This is because the attacker of Figure 8 receives explicitly the model m, whereas the adversary $\mathcal{A}_2$ in Figure 5 receives the confidence values in l, and black-box access to the model. To rely on such an attacker, $\mathcal{A}_2$ must therefore be able to produce input m from its own view of the system. As such, relying on such an adversary implies there is an efficient way $m \leftarrow\$ \mathcal{A}_{\mathsf{bb}}^m(l)$ to retrieve the model m from confidence values l and black-box access to the model m, which is exactly the setting of *model extraction* attacks in the previous section. This adversary $\mathcal{A}_{\mathsf{bb}}$ can then be called by $\mathcal{A}_2$ to produce input m, which is then forwarded to the adversary of Figure 8 to produce a successful attack r. As such, SOTERIA is vulnerable to white-box *reconstruction attacks* if there exists an efficient adversary $\mathcal{A}'$ that successfully wins the experiment of Figure 7 for the *model extraction* attack.

B.5 Summary

Table 1 summarizes the attacks discussed. These are divided between all the identified classes of attacks, as well as whether the adversary is external or if it can query the model as a client. In many instances, the security of our system hinges on another argument over specific restrictions assumed for the adversary. We now list the arguments that propose the security of our system in the different contexts.

{1}: an adversary is unable to retrieve the (white-

		External		Client	
		SML-1	SML-2	SML-1	SML-2
Adversarial samples		✓	✓	✓	✓
Reconstruction	WB	✓	{1a}	{1b}	{1c}
	BB	✓	{2}	✗	✗
Membership inference		✓	{2}	✗	✗
Model inversion		✓	{2}	✗	✗
Model extr	Equation	✓	{2}	{3a}	{3b}
	Path	✓	{2}	{4a}	{4b}
	Class	✓	{2}	{5a}	{5b}
	DF KD	✓	{2}	✗	✗

Table 1: Summary of attacks against SOTERIA. ✓ means SOTERIA is resilient to the attacks, ✗ means SOTERIA is vulnerable to the attacks, and {X} means SOTERIA is secure if argument {X} is also true.

box) model from confidence values {1a}, black-box access to the model {1b}, or both {1c}.

{2}: an adversary is unable to emulate black-box access to the model from confidence values.

{3}: an adversary is unable to retrieve the dimension of the dataset from black-box access to the model {3a} and confidence values {3b}.

{4}: an adversary is unable to retrieve information of leaf count, depth and ID from black-box access to the model {4a} and confidence values {4b}.

{5}: no *model inversion* attack exists for retrieving dataset samples for SML-1 {5a} and SML-2 {5b}.

White-box based attacks explicitly require additional information, such as feature vectors, over black-box access to the model [4]. This is something that supports {1b} directly, and since confidence values are not computed from feature vectors, so would be {1a} and {1c}. Extracting model access from only confidence values is an active area of research, but current attacks [3] are still unable to do this in an efficient way {2}. Typically, one cannot infer dimension from simply querying the model, which suggests {3a} is true, but this is unclear for confidence values, and thus one might consider {3b} is false. [5] suggests {4a} is true, but {4b} is not. {5} will fundamentally depend on the application [3, 5].

References

[1] Raad Bahmani, Manuel Barbosa, Ferdinand Brasser, Bernardo Portela, Ahmad-Reza Sadeghi, Guillaume Scerri, and Bogdan Warinschi. Secure multiparty computation from sgx. In Aggelos Kiayias, editor, *Financial Cryptography and Data Security*, pages 477–497, Cham, 2017. Springer International Publishing.

[2] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145, 2001.

[3] Varun Chandrasekaran, Kamalika Chaudhuri, Irene Giacomelli, Somesh Jha, and Songbai Yan. Exploring connections between active learning and model extraction. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pages 1309–1326, 2020.

[4] Ahmed Salem, Apratim Bhattacharya, Michael Backes, Mario Fritz, and Yang Zhang. Updates-leak: Data set inference and reconstruction attacks in online learning. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pages 1291–1308, 2020.

[5] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 601–618, 2016.