Addendum to "SecureTVM: A TVM-Based Compiler Framework for Selective Privacy-Preserving Neural Inference"

Po-Hsuan Huang*[1], Chia-Heng Tu†[1], Shen-Ming Chung[2], Pei-Yuan Wu[3], Tung-Lin Tsai[3], Yi-An Lin[3], Chun-Yi Dai[3], and Tzu-Yi Liao[3]

[1]Department of Computer Science and Information Engineering, National Cheng Kung University, Taiwan
[2]Industrial Technology Research Institute, Taiwan
[3]Department of Electrical Engineering, National Taiwan University, Taiwan

March 6, 2022

A An Example of SecureTVM Workflow

The workflow of SecureTVM for converting a deep learning model into the ABY 2PC code is illustrated in Fig. 1, where the input model with a convolution layer (`Conv2D`) and a fully-connected layer (`Dense`) is illustrated in the top-left corner of Fig. 1 and the corresponding ABY code is listed in the bottom-right corner of Fig. 1. The input model defined by Keras is converted into the NN operations supported by Relay IR[1], which is a computational graph IR. At this IR level, the input model is defined by the model input (i.e., `%input_1`), the involved NN operations (e.g., `nn.conv2d`) and their parameters (e.g., `%v_param_1`), and the intermediate outputs (e.g., `%1`). The implementations of the NN operations are specified when the model is further *lowered* to the TVM IR. Note that each `produce` block is responsible for producing the results for a tensor, such as `produce conv.global` for computing the results for the tensor data `conv.global`. The `allocate` command is to create a buffer to keep temporary results; for example, `data_vec` and `kernel_vec` is to keep the preprocessed data that are required by `conv`. The loops that are able to be parallelized are labeled with `parallel` blocks.

The ABY backend produces the 2PC code, which consists of the implementation for the main function and the layer functions, generated by NN network generator and NN layer implementation generator, respectively. The main function prepares the necessary data required by the model layers, which are represented by the function calls to the involved operations. The layer functions are the implementation of the NN operations required by the main function. The SecureBoost library offers the primitive and optimized functions required by the main/layer functions. For instance, we have created the fixed-point data type, `s_float` and implemented the primitive operations (e.g., additions and subtractions) for the fixed-point numbers. The key concepts are highlighted as follows.

- There is only one copy of the ABY 2PC program to be run by the two participants, the data provider and the model provider; it is similar to the single program, multiple data (SPMD) paradigm. When the program instances are running, the flags are used to specify the *roles*, i.e., `CLIENT` for the data provider and `SERVER` for the model provider.
- Marker 1 shows that the helper function `PUB::input_from_file` is used to load the input data for the two parties. For the data provider, her/his data is loaded once. On the contrary, as for the model provider, each model layer has its own parameters, and the parameters of each layer are loaded right before they are used. For instance, `p1.txt` points to the file containing the parameter for the first layer `fused_nn_conv2d_PRI`.

*aben20807@gmail.com
†chiaheng@ncku.edu.tw
[1]It is done by invoking the TVM frontend function call, `from_keras`.

- Marker 2 illustrates the loaded data are converted into the ciphertext (by `pub_to_pri`) before they are used by the function of the first layer. It is running in the secure mode.
- Marker 3 depicts the mechanism used for the *layer-by-layer circuit generation and execution*, which is done by the *reset* functionality supported by ABY to reclaim the resources consumed by the previously constructed circuits. That is, the code segment before the `ExecCircuit` function is the construction of circuits for the `Conv2D` layer (the setup phase), and the execution of the `ExecCircuit` function actually performs the secure inference of the `Conv2D` layer (the online phase). Before the `reset` function is invoked to free the allocated resources for the layer, the `pri_to_shared` function is used to keep the intermediate outputs of the `Conv2D` layer[2] in a temporary secure store (`_input_5_reset`), which is then served as the input to the next layer via the `shared_to_pri` function. The layer-wise circuit generation/execution can be done by injecting the `ExecCircuit` and `reset` functions at the layer boundaries with the support of the temporary store mechanism.
- Marker 4 demonstrates that changing the execution from secure to native mode is done by using the public data type. That is, the outcomes calculated in the previous layer are converted into the public version via the `pri_to_pub` function, and the input parameters are stored as the public data type `PUB::s_float`. As a result, the last layer `Dense` runs in the native mode at the machine of data provider, and the model provider skip the computation of `Dense` directly to prevent aimless computing on zero data.
- Finally, as the claim in Section 3.1 of "SecureTVM: A TVM-Based Compiler Framework for Selective Privacy-Preserving Neural Inference", the `softmax` layer is evaluated by the data provider locally to get the prediction result from the 2PC inference; on the other hand, model provider only get an array be filled with zero.

[2] Actually, it is the outcomes computed by the `fused_nn_relu_PRI` function.

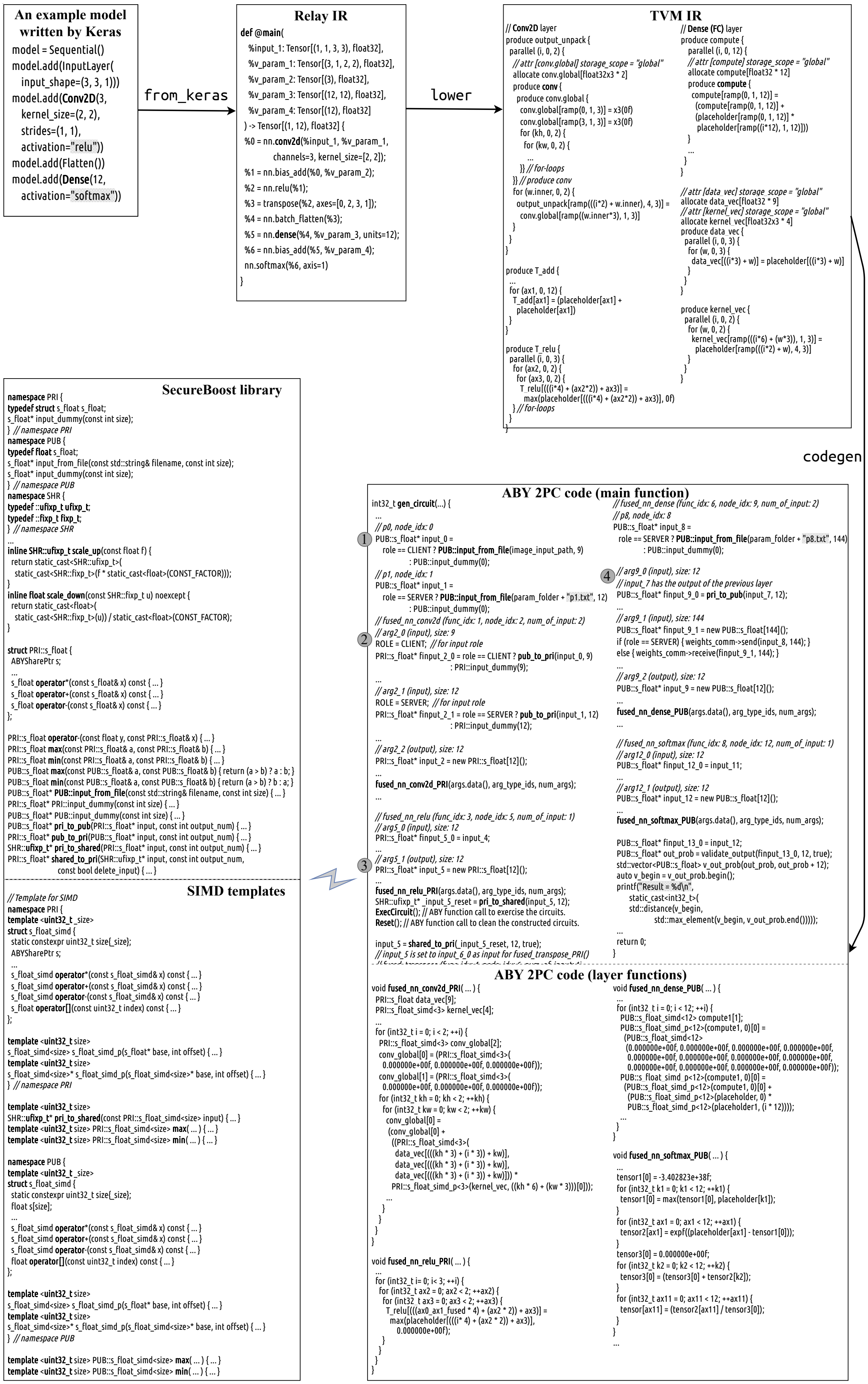

Fig. 1: An example workflow for converting an NN model to the 2PC code version.