Mystique: Efficient Conversions for Zero-Knowledge Proofs with Applications to Machine Learning

Chenkai Weng Kang Yang
Xiang Xie Jonathan Katz Xiao Wang

How to Demonstrate Evasion Attacks

I can crash your model, but I won't reveal the vulnerability unless you offer me bug bounty rewards.

Challenger

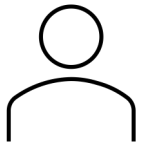
Model

()

Input

→ "Panda"

Show me in zero-knowledge!



Model Developer

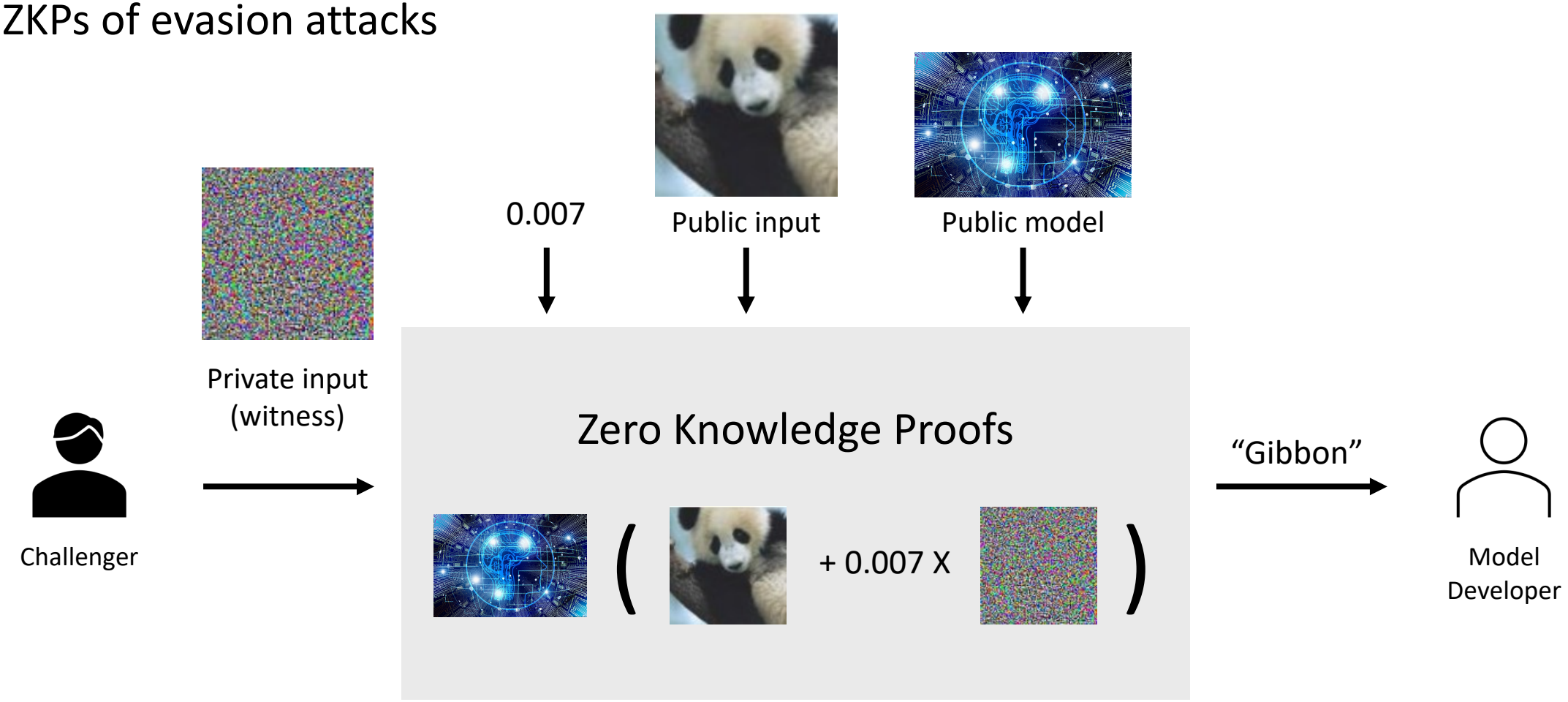
Reference: I. J. Goodfellow, J. Shlens and C. Szeged, Explaining and Harnessing Adversarial Examples, International Conference on Learning Representations, 2015.

Zero-Knowledge Proofs (of Knowledge)

- For a predicate C, a prover P convinces a verifier V that it knows a witness ω such that $C(\omega) = 1$, with
  - **Completeness**: if $C(\omega) = 1$, then V is convinced.
  - **Soundness**: if $C(\omega) \neq 1$, V is convinced with negligible probability.
    - Stronger properties required for proof of knowledge.
  - **Zero-knowledge**: no information is revealed except for the output of $C(\omega)$.

Applications for Machine Learning

ZKPs of evasion attacks



Applications for Machine Learning

ZKPs of evasion attacks

ZKPs of correct inference

ZKPs of private benchmarks

Current Approaches are not sufficient because…

For Large-Scale Machine Learning Tasks

Scalability

Circuits containing billions of gates

Supporting ML operations

Neural networks involving linear/non-linear layers

Efficiency

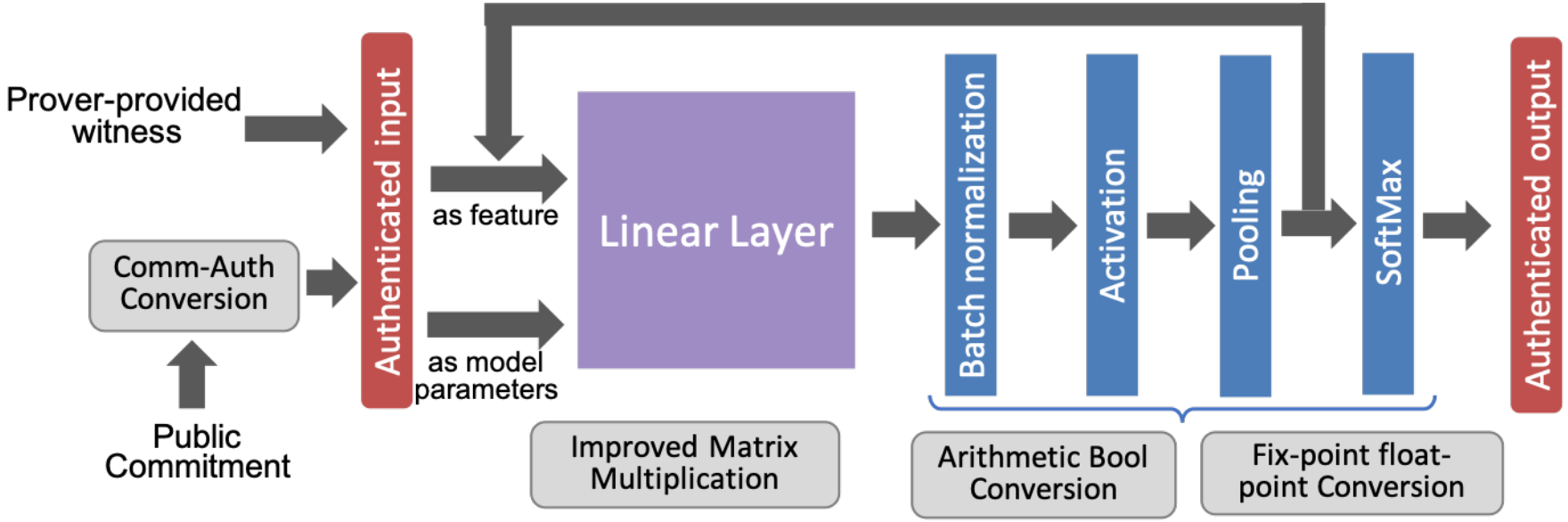
Meaningful applications run in short time

Ease-of-use

Cryptographic protocols behind simple interface

Our result

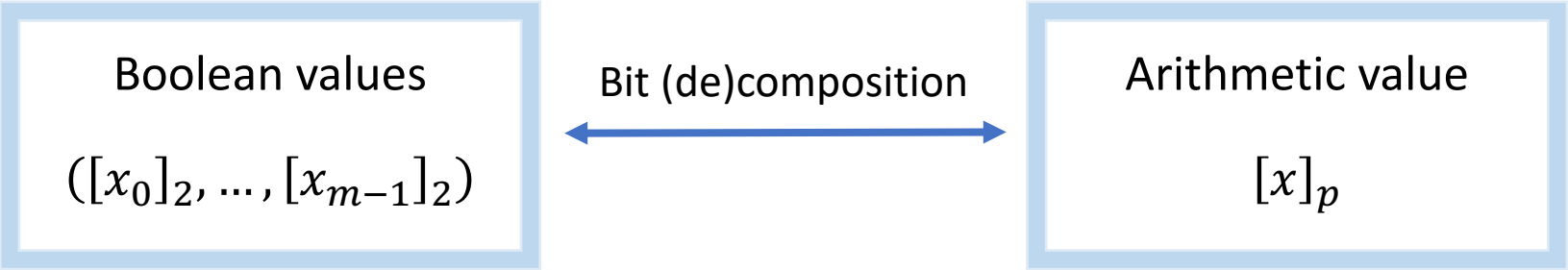
- Interactive zero-knowledge proof protocol for machine learning tasks.
- Prove private inference using ResNet-101 model (42.5 million model parameters) in 336 s.
- Optimizations for Sigmoid, Max Pooling, ReLU, SoftMax and Batch normalization.
  - Easily extend to other operations.



Key Contribution 1

Conversion Between *Boolean* and *Arithmetic* Circuits

- **Goal**: convert between $([x_0]_2, \ldots, [x_{m-1}]_2)$ and $[x]_p$



Key Contribution 1

Conversion Between *Boolean* and *Arithmetic* Circuits

- **Goal**: convert between $([x_0]_2, \ldots, [x_{m-1}]_2)$ and $[x]_p$
- **ZK-friendly extended doubly authenticated bits** (zk-edabits)
  - Adopted from *dabits*/*edabits* in MPC settings.

Reference: D. Rotaru and T. Wood. MArBled circuits: Mixing arithmetic and Boolean circuits with active security. CRYPTO 2019.
D. Escudero, S. Ghosh, M. Keller, R. Rachuri, and P. Scholl. Improved primitives for MPC over mixed arithmetic-binary circuits. CRYPTO 2019.

Key Contribution 1

Conversion Between *Boolean* and *Arithmetic* Circuits

- **Goal**: convert between $([x_0]_2, \ldots, [x_{m-1}]_2)$ and $[x]_p$
- **ZK-friendly extended doubly authenticated bits** (zk-edabits)
  - Adopted from *dabits*/*edabits* in MPC settings.
- **Performance**

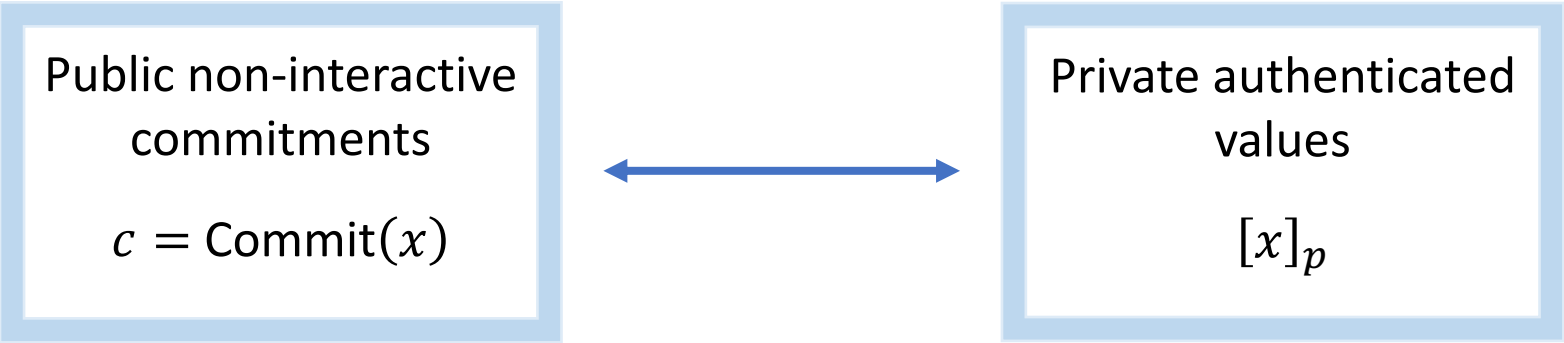
Average time per conversion

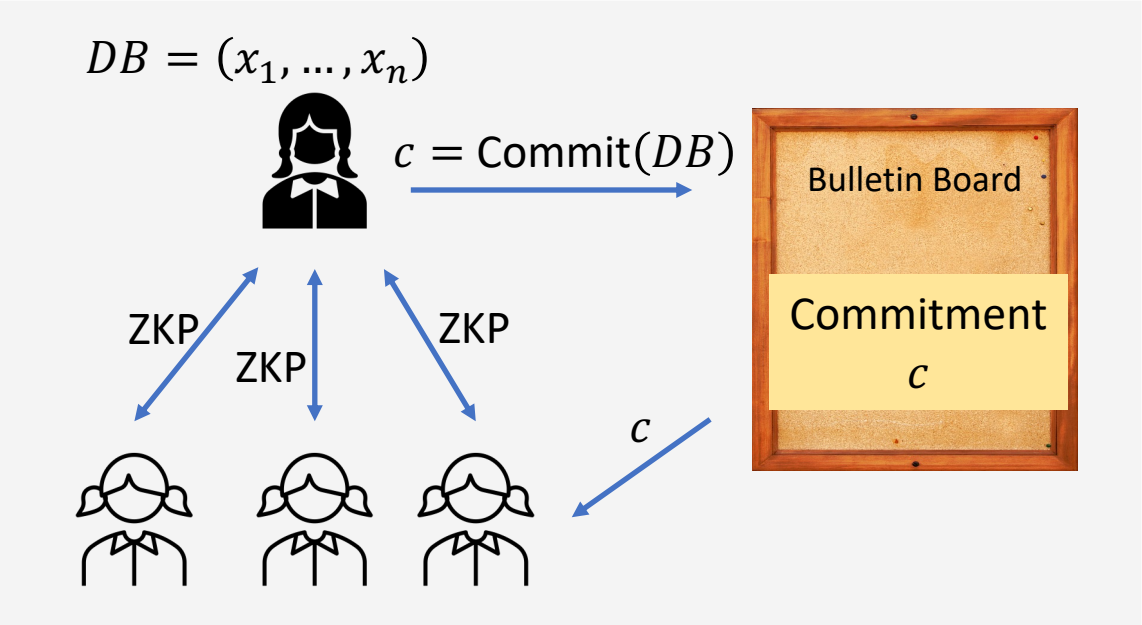
	50 Mbps	200 Mbps	1 Gbps
A2B	107 μs	45 μs	29 μs
B2A	109 μs	49 μs	33 μs

Reference: D. Rotaru and T. Wood. MArBled circuits: Mixing arithmetic and Boolean circuits with active security. CRYPTO 2019.
D. Escudero, S. Ghosh, M. Keller, R. Rachuri, and P. Scholl. Improved primitives for MPC over mixed arithmetic-binary circuits. CRYPTO 2019.

Key Contribution 2

Conversion from Publicly Committed Values to Privately Authenticated Values





Key Contribution 2

Conversion from Publicly Committed Values to Privately Authenticated Values

- Hybrid commitment
  - Security reduced to random oracle and PRF.
- Performance

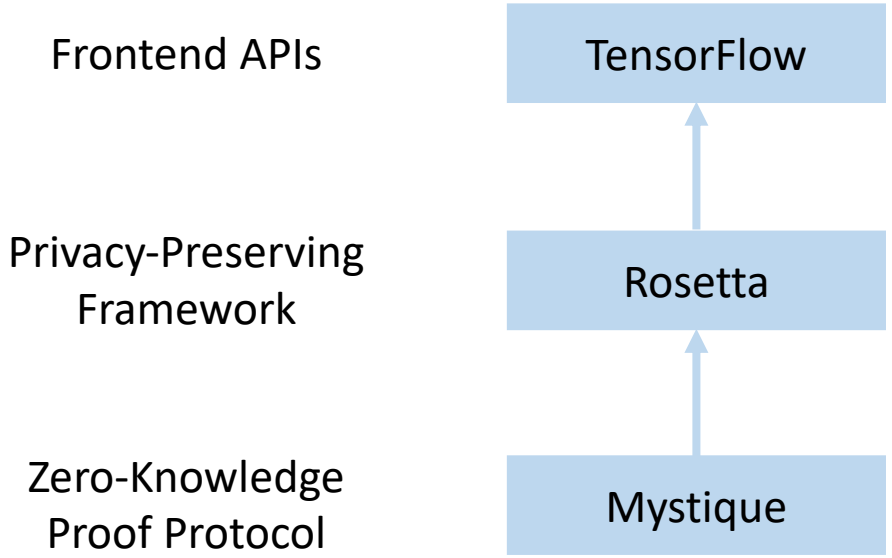
Average time per conversion

	50 Mbps	200 Mbps	1 Gbps
C2A	56 μs	55 μs	55 μs

Key Contribution 3

Integration with TensorFlow

- Simple interface
  - *Rosetta* framework based on TensorFlow.




```
##########################################################################
# Generator
##########################################################################

def network(self, x, is_training=True, reuse=False):
    with tf.variable_scope("network", reuse=reuse):

        if self.res_n < 50:
            residual_block = resblock
        else:
            residual_block = bottle_resblock2

        residual_list = get_residual_layer(self.res_n)

        ##########################################################################

        ch = 64
        x = conv(x, channels=ch, kernel=3, stride=1, scope='conv')  # 32
        # x = conv(x, channels=ch, kernel=7, stride=2, scope='conv')
        x = batch_norm(x, is_training, scope='batch_norm0')
        x = relu(x)
        x = tf.nn.max_pool(
            x, ksize=(1, 3, 3, 1), strides=(1, 2, 2, 1), padding='VALID')

        ##########################################################################
```

Other Contributions

- Matrix multiplication.
  - Freivalds algorithm.

	50 Mbps	200 Mbps	1 Gbps
2048×2048	15.19 s	11.30 s	10.39 s

- Conversion of fixed-point and floating-point numbers.

	50 Mbps	200 Mbps	1 Gbps
Fix2Float	50 μs	46 μs	46 μs
Float2Fix	49 μs	46 μs	46 μs

Benchmarking Private Inference

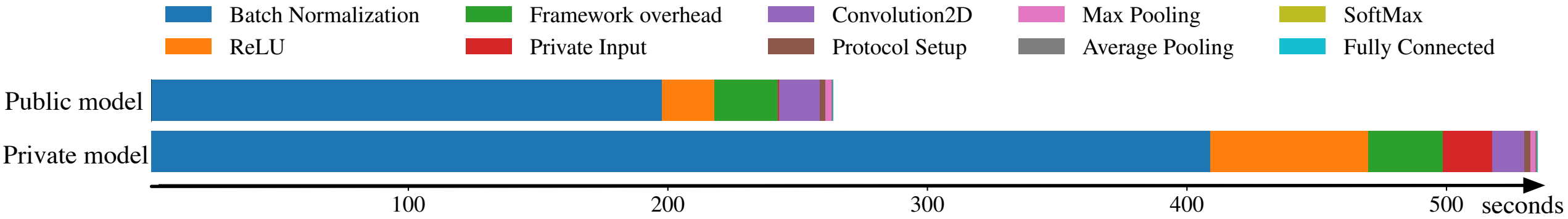
- Prove inference of large neural network.

101 layers, 42.5 million parameters

Model	Image		LeNet-5	ResNet-50	ResNet-101
Private	Private	Exec. time	5.9 s	333 s	535 s
		Comm.	16.5 MB	1.27 GB	1.98 GB
Private	Public	Exec. time	5.5 s	336 s	541 s
		Comm.	16.5 MB	1.27 GB	1.98 GB
Public	Private	Exec. time	4.9 s	158 s	262 s
		Comm.	16.4 MB	0.53 GB	0.99 GB

Network Bandwidth: 200 Mbps
Image from CIFAR-10 dataset

Microbenchmark of ResNet-101



Thank You

Full version: https://eprint.iacr.org/2021/730

Implementation:

Contact information:
Chenkai Weng, *Northwestern University, ckweng@u.northwestern.edu*