Zero Knowledge Proofs

ZKP Applications

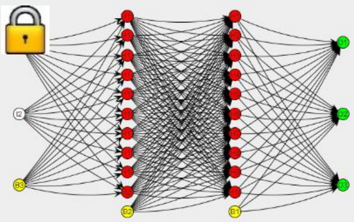
Instructors: Dan Boneh, Shafi Goldwasser, Dawn Song, Justin Thaler, **Yupeng Zhang**





Stanford University | Berkeley UNIVERSITY OF CALIFORNIA | GEORGETOWN UNIVERSITY | TEXAS A&M UNIVERSITY

ZKP for Machine Learning



ML inference

fair or not?

Credit Risk Prediction

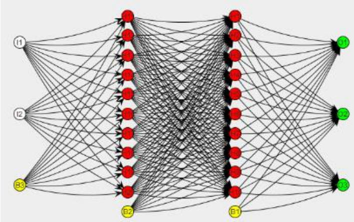
Criminal Justice

Healthcare

Proving ML Inferences using ZKP

Zero-knowledge proof without revealing the ML models

- ✓ Fairness of ML models
- ✓ Integrity of ML inferences



ML inference

Challenges

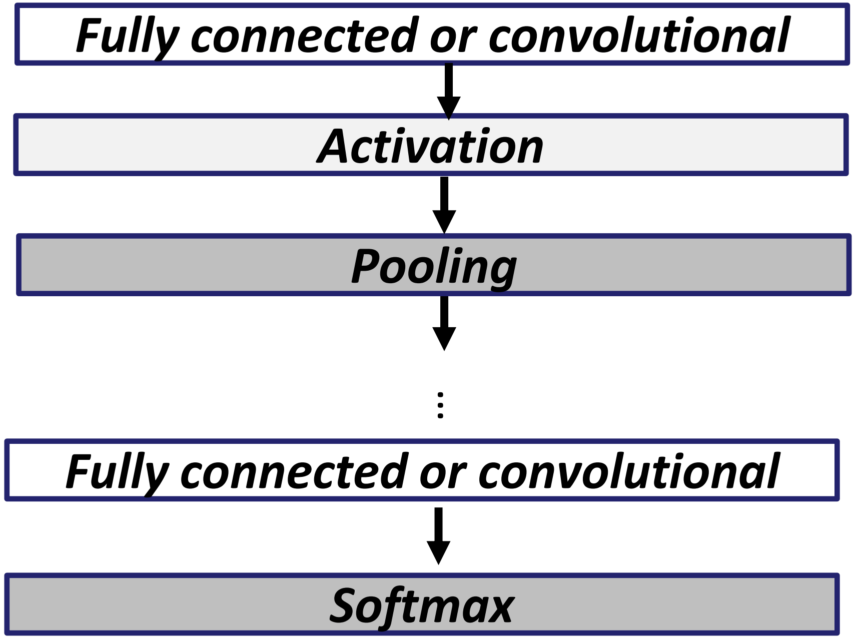
Efficiency and Scalability of general-purpose SNARKs:

scale to $<2^{30}$ = 1 billion gates (64 GB RAM), prover time minutes to hours

VGG 16 on CIFAR-10

- 15 million parameters in the model
- 1.1 billion gates for an inference

Solution: Special-Purpose ZKPs



ZKP for Matrix Multiplication [Thaler'13]

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \quad B = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{pmatrix} \quad C = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{pmatrix}$$

Matrix multiplication $C = A \times B$: $c_{ij} = \sum_k a_{ik} b_{kj}$

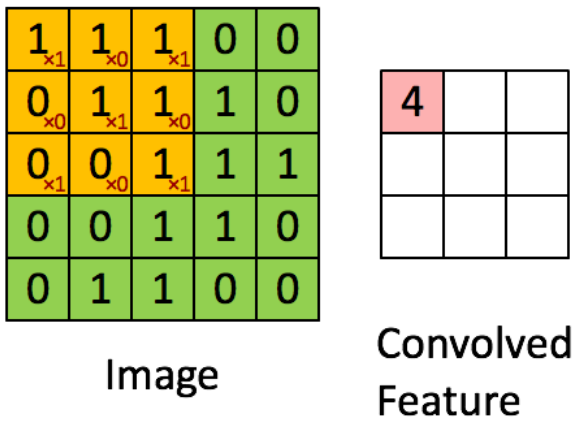
$$C(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{z}} A(\boldsymbol{x}, \boldsymbol{z}) B(\boldsymbol{z}, \boldsymbol{y}) \qquad C(\boldsymbol{i}, \boldsymbol{j}) = c_{ij} \quad A(\boldsymbol{i}, \boldsymbol{k}) = a_{ik} \quad B(\boldsymbol{k}, \boldsymbol{j}) = b_{kj}$$

- Efficient ZKP with prover time $O(n^2)$, proof size $O(\log n)$
- Faster than computing the result in $O(n^3)$
- Verifying is easier than computing

ZKP for 2-D Convolutions [LXZ'21]

2-D convolution $C = A * B$

O(NK) time to compute



Computing Convolution using FFT

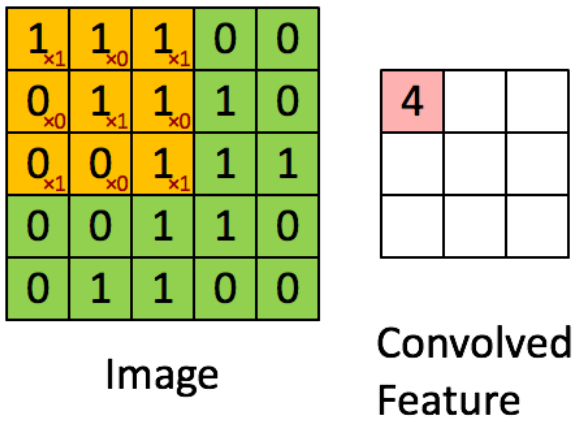
- Equivalent to 1-D convolution

$$c = a * b = \sum_i a_i \, b_{N-i}$$

- Same as polynomial multiplication

$$c(x) = a(x) \cdot b(x)$$

- Can be computed by Fast Fourier Transform (FFT)



ZKP for Fast Fourier Transform

$\bar{a} = F \times a$

$$\bar{a}(\boldsymbol{x}) = \sum_{\boldsymbol{y}} F(\boldsymbol{x}, \boldsymbol{y}) \cdot a(\boldsymbol{y})$$

$$F = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^1 & \omega^2 \\ 1 & \omega^3 & \omega^2 & \omega^1 \end{pmatrix}$$

× Size of $F(x, y)$ is N^2

✓ F consists of only N distinct values

- An efficient sumcheck protocol with prover time **O(N)**, proof size O(log N), verifier time O($\log^2$ N)
- Sublinear in the computation time O(NlogN)

Performance of zkCNN

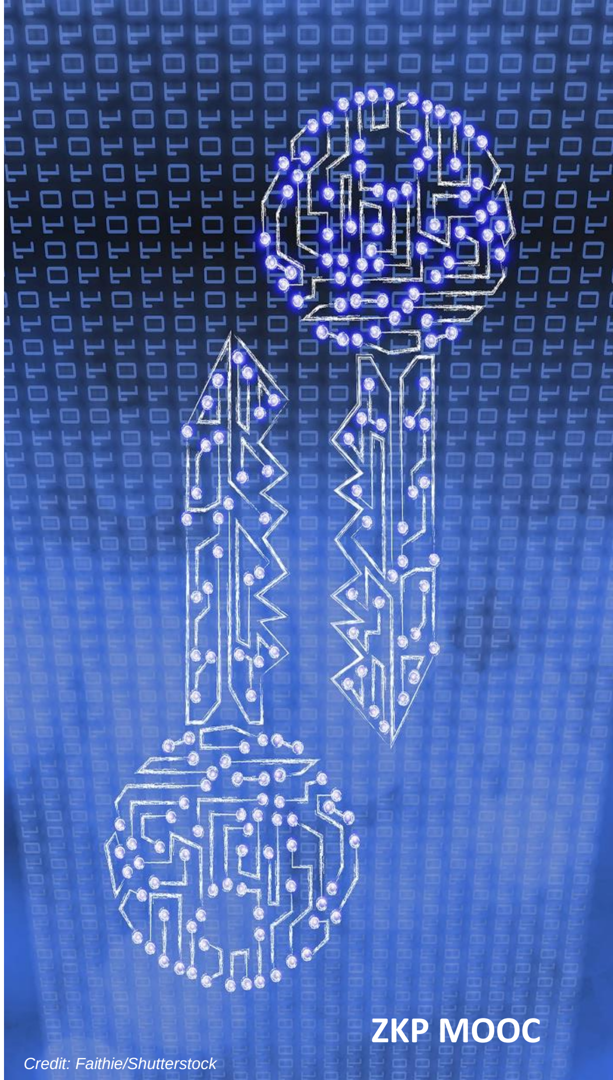
	1 inference	Accuracy on 100 images
Prover time	88 seconds	680 seconds
Proof size	341 KB	673 KB
Verifier time	59 ms	121 ms

VGG16 on CIFAR10 dataset, 15 million parameters (120MB)

Other Related Works on ZKML

ZKDT[ZFZD20], vCNN [LKKO20], ZEN [FQZ+21], Mystique [WYX+21], pvCNN [WWT+22], [KHSS22], …

ZKP for Program Analysis



Credit: Faithie/Shutterstock

Zero-knowledge Program Analysis

public function: static analysis algorithm

secret program *P*

```
#include <stdio.h>
#include <string.h>
void main(void)
{
   char str1[10];
   char str2[10];

   strcpy(str1, "Testing");
   printf("Length is %d\n", strlen(str1));
   strcpy(str2, str1);
   printf("String2 = %s\n", str2);

   if (strcmp(str1, str2) == 0)
      printf("Both strings are the same\n");

   str1[4] = '\0';
   strncat(str1, str2, 3);
   printf("String1 is now: %s\n");

   if (strncmp(str1, str2, 4) == 0)
      printf("The strings are still equal\n");
}
```

safety properties of *P*

Zero-knowledge Vulnerability Disclosure

public program

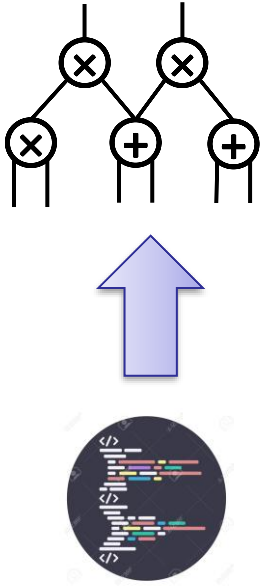
```
#include <stdio.h>
#include <string.h>
void main(void)
{
   char str1[10];
   char str2[10];

   strcpy(str1, "Testing");
   printf("Length is %d\n", strlen(str1));
   strcpy(str2, str1);
   printf("String2 = %s\n", str2);

   if (strcmp(str1, str2) == 0)
      printf("Both strings are the same\n");

   str1[4] = '\0';
   strncat(str1, str2, 3);
   printf("String1 is now: %s\n");

   if (strncmp(str1, str2, 4) == 0)
      printf("The strings are still equal\n");
}
```

secret vulnerability

Running the program leads to crash

Challenges

- ZKP schemes support circuits.
- Program analysis is usually RAM computation



Solution: Auxiliary Inputs

Ask the prover to provide additional data as the input of ZKP

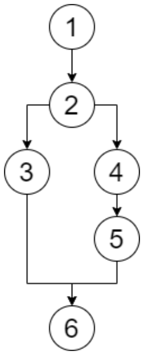
- Not trusted
- Not sent to the verifier
- Significantly improves the efficiency of ZKP

Example: worklist algorithm

Program:

```
1  x1 = source()
2  if (x2 > 5):
3      x3 = x1
4  else
5      x3 = 9
6  sink(x3)
```

CFG:



Worklist:

(1, 2)								

State:

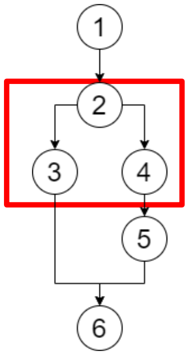
Line No.	1	2	3	4	5	6
(x1, x2, x3)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)

Worklist algorithms: update

Program:

```
x1 = source()
if (x2 > 5):
    x3 = x1
else
    x3 = 9
sink(x3)
```

CFG:



Worklist:

(1, 2)	(2, 3)	(2, 4)						

State:

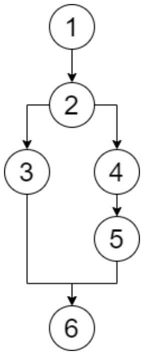
Line No.	1	2	3	4	5	6
(x1, x2, x3)	(0, 0, 0)	(1, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)

Worklist algorithms: update

Program:

```
1   x1 = source()
2   if (x2 > 5):
3       x3 = x1
4   else
5       x3 = 9
6   sink(x3)
```

CFG:



Worklist:

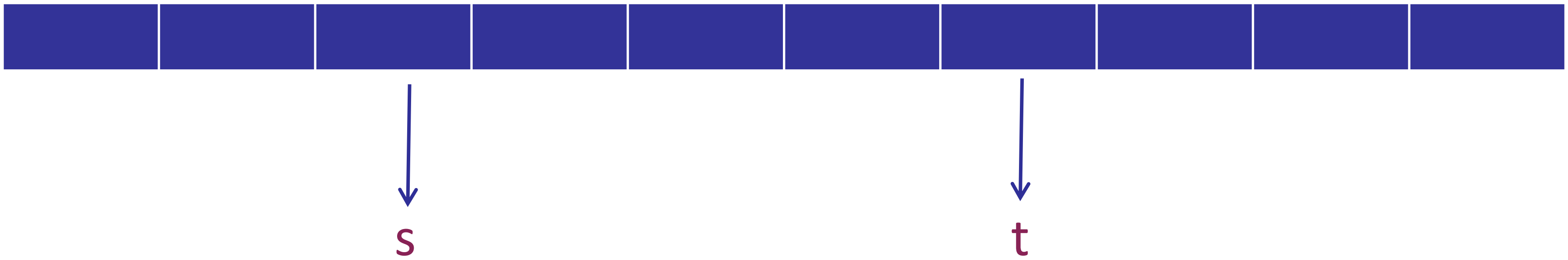
(1, 2)	(2, 3)	(2, 4)	(3, 6)	(4, 5)				

State:

Line No.	1	2	3	4	5	6
(x1, x2, x3)	(0, 0, 0)	(1, 0, 0)	(1, 0, 0)	(1, 0, 0)	(0, 0, 0)	(1, 0, 1)

Auxiliary inputs

- Prover provides final state of the list
- Prover provides head and tail of each step
- The circuit checks the correctness (offline memory checking [BEGKN’91,Setty’20, …])



Performance

Program with T steps and v variables

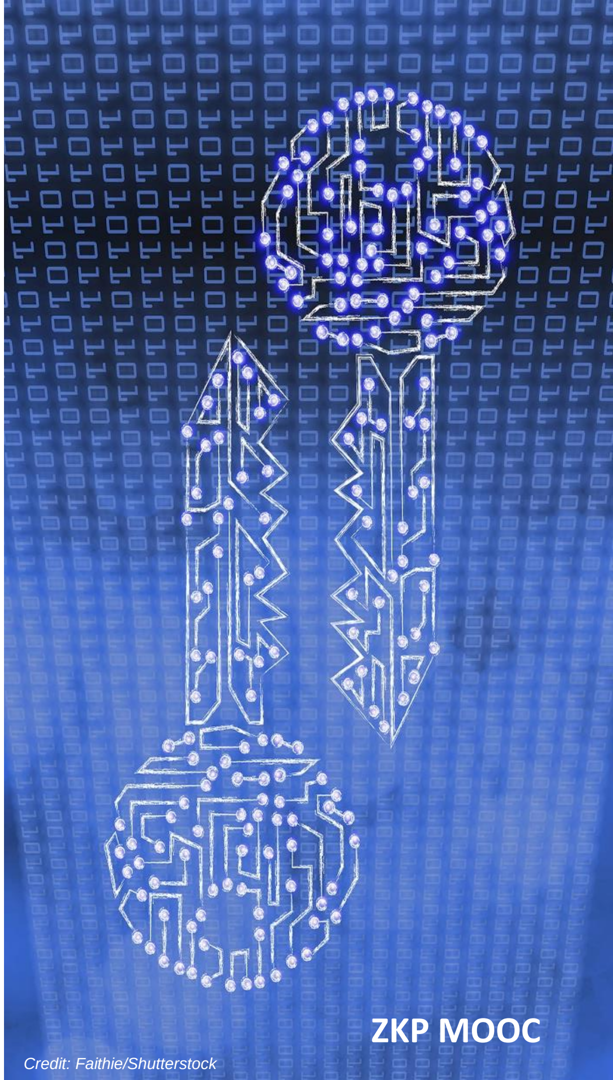
Worklist algorithm: $O(T \cdot v)$

$\longrightarrow$ circuit of size $O(T \cdot v + T \log T)$

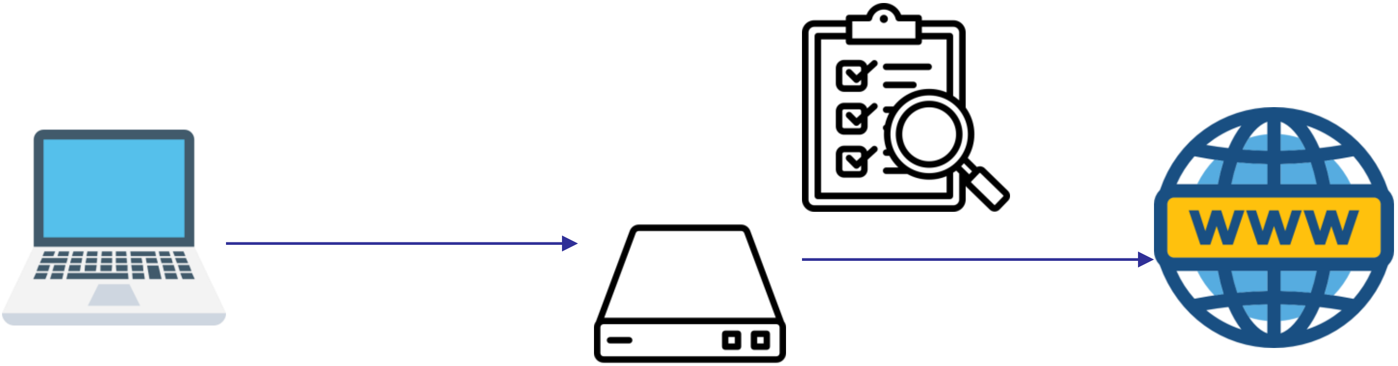
Related works

- Static analysis: [FDNZ’21, LAHPTW’22, …]
- Vulnerabilities: [GHHKPV’22, CHPPT’23, …]

ZKP for Middlebox

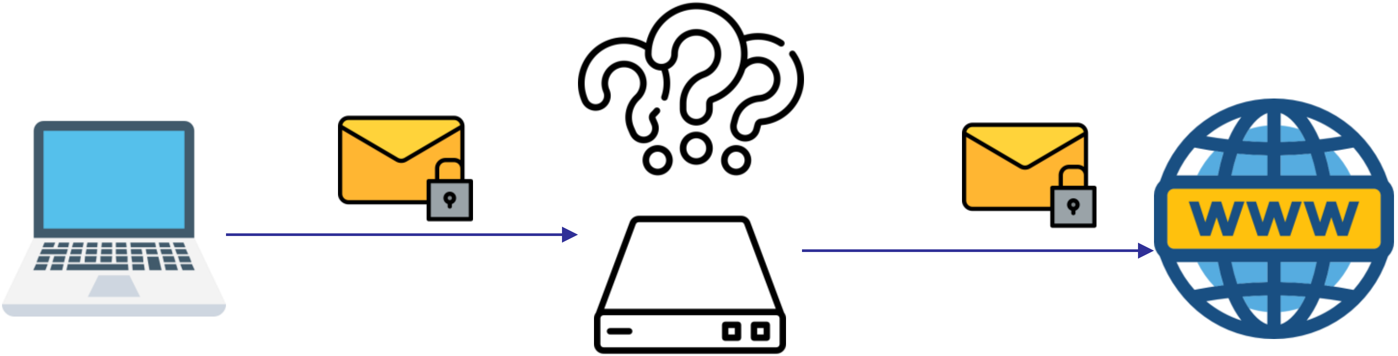


Middleboxes

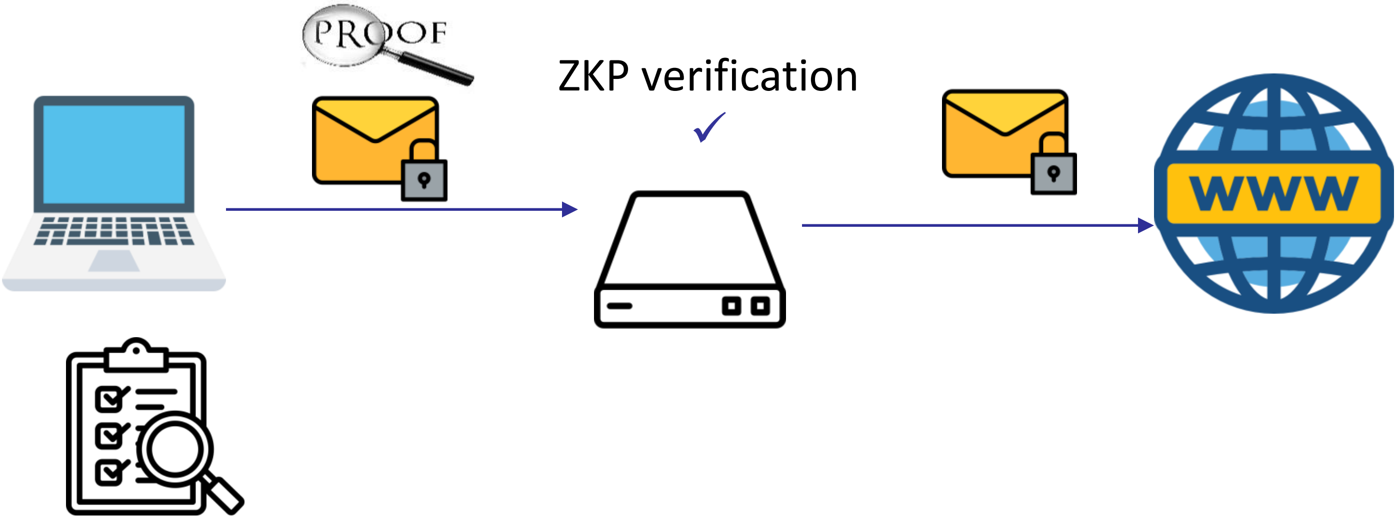


Middleboxes inspect traffic to ensure security policy

Encrypted Traffic



Zero-Knowledge Middleboxes [GAZBW'22]



Challenges

- Work with TLS 1.3
- Legacy cryptographic functions such as AES, SHA

End of Lecture