

# CMSC818I

# Background

Yizheng Chen | University of Maryland  
[surrealyz.github.io](https://surrealyz.github.io)

September 3, 2024

# Standard Metric to Evaluate Correctness

- `pass@k`
  - Given  $k$  generations, the expected likelihood of generating correct code in any generation

# pass@k

- $n$ : # of generated samples
  - $n \geq k$
- $c$ : # of correct samples that pass unit tests

# pass@k

- $n$ : # of generated samples
  - $n \geq k$
- $c$ : # of correct samples that pass unit tests

$$\text{pass@}k := \mathbb{E}_{\text{Problems}} \left[ 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}} \right]$$

# pass@k

- $n$ : # of generated samples
  - $n \geq k$
- $c$ : # of correct samples that pass unit tests

$$\text{pass@}k := \mathbb{E}_{\text{Problems}} \left[ 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}} \right]$$

Any  $k$  generations

# pass@k

- $n$ : # of generated samples
  - $n \geq k$
- $c$ : # of correct samples that pass unit tests

Any  $k$  incorrect generation

$$\text{pass}@k := \mathbb{E}_{\text{Problems}} \left[ 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}} \right]$$

Any  $k$  generations

# pass@k

- $n$ : # of generated samples
  - $n \geq k$
- $c$ : # of correct samples that pass unit tests

$$\text{pass@}k := \mathbb{E}_{\text{Problems}} \left[ 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}} \right]$$

Chance of having incorrect generations out of  $k$  generations

# Scientific Peer Reviewer

The paper has not been published yet and is currently submitted to a top conference where you've been assigned as a peer reviewer. Complete a full review of the paper answering all prompts of the official review form of the top venue in this research area. This includes recommending whether to accept or reject the paper.



# Scientific Peer Reviewer

## Scientific Contribution

1. Independent Confirmation of Important Results with Limited Prior Research
2. Provides a New Data Set For Public Use
3. Creates a New Tool to Enable Future Science
4. Addresses a Long-Known Issue
5. Identifies an Impactful Vulnerability
6. Provides a Valuable Step Forward in an Established Field
7. Establishes a New Research Direction
8. Other

# Scientific Peer Reviewer

Show review form

# Archaeologist

You're an archeologist who must determine where this paper sits in the context of previous and subsequent work. Find and report on one older paper cited within the current paper that substantially influenced the current paper and one newer paper that cites this current paper.

# Academic Researcher

You're a researcher who is working on a new project in this area. Propose an imaginary follow-up project not just based on the current but only possible due to the existence and success of the current paper.

# Industry Practitioner

You work at a company or organization developing an application or product of your choice (that has not already been suggested in a prior session). Bring a convincing pitch for why you should be paid to implement the method in the paper, and discuss at least one positive and negative impact of this application.

# Hacker

You're a hacker who needs a demo of this paper ASAP. Modify the implementation of the paper to make it run on a small dataset or toy problem. Prepare to share the core code of the algorithm to the class and demo your implementation. Do not simply download and run an existing implementation – though you are welcome to use (and give credit to) an existing implementation for “backbone” code.

# Experiment Template

- Research Question / Problem
- Related work
- Experiment setup
  - Is there a simpler nontrivial version to try instead?
- Results
  - Stare at the outline / results / goals, does the set up make sense?
- What you learned from the result
- What is the next step

# Private Investigator

You are a detective who needs to run a background check on one of the paper's authors. Where have they worked? What did they study? What previous projects might have led to working on this one? What motivated them to work on this project?



# Social Impact Assessor

Identify how this paper self-assesses its (likely positive) impact on the world. Have any additional positive social impacts left out? What are possible negative social impacts that were overlooked or omitted?



**Soheil Feizi** ✓

@FeiziSoheil



🤔 The LLM Benchmark Conjecture: For any LLM, there exists a benchmark where it's at the top of the leaderboard!

7:51 PM · Jun 27, 2024 · **6,012** Views



**Brendan Dolan-Gavitt** ✓ ✨ @moyix · Jun 27



Proof by contradiction: assume we have heard of an LLM that is not at the top or any benchmark. But in this case observe that it would not have been published and we would not have heard of it, contradicting our assumption. Therefore such an LLM cannot exist; QED.



5



257



# Main Challenge

Test set contamination



# **Background in Computer Security**

# **Exercise: Vulnerability vs Exploit Program**

# 2023 CWE Top 25 Most Dangerous Software Weaknesses

[Top 25 Home](#)

Share via: [Twitter](#)

[View in table format](#)

[Key Insights](#)

[Methodology](#)

1

Out-of-bounds Write

[CWE-787](#) | CVEs in KEV: 70 | Rank Last Year: 1

2

Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

[CWE-79](#) | CVEs in KEV: 4 | Rank Last Year: 2

3

Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

[CWE-89](#) | CVEs in KEV: 6 | Rank Last Year: 3

4

Use After Free

[CWE-416](#) | CVEs in KEV: 44 | Rank Last Year: 7 (up 3) ▲

5

Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')

[CWE-78](#) | CVEs in KEV: 23 | Rank Last Year: 6 (up 1) ▲

6

Improper Input Validation

[CWE-20](#) | CVEs in KEV: 35 | Rank Last Year: 4 (down 2) ▼

7

Out-of-bounds Read

[CWE-125](#) | CVEs in KEV: 2 | Rank Last Year: 5 (down 2) ▼

8

Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

[CWE-22](#) | CVEs in KEV: 16 | Rank Last Year: 8

[https://cwe.mitre.org/top25/archive/2023/2023\\_top25\\_list.html](https://cwe.mitre.org/top25/archive/2023/2023_top25_list.html)

# SQL Databases

- SQL: Structured Query Language
  - Create and query data
- A database has some tables
- A table has a predefined structure

# SQL Database Example

Table

**Users**

Table name

Name	Gender	Age	Email	Password
Dee	F	28	<a href="mailto:dee@pp.com">dee@pp.com</a>	j3i8g8ha
Mac	M	7	<a href="mailto:bouncer@pp.com">bouncer@pp.com</a>	a0u23bt
Charlie	M	32	<a href="mailto:aneifjask@pp.com">aneifjask@pp.com</a>	0aergja
Dennis	M	28	<a href="mailto:imagod@pp.com">imagod@pp.com</a>	1bjb9a93

Row  
(Record)

Column



# SQL (Standard Query Language) Example

**Users**

Name	Gender	Age	Email	Password
Dee	F	28	<u>dee@pp.com</u>	j3i8g8ha
Mac	M	7	<u>bouncer@pp.com</u>	a0u23bt
Charlie	M	32	<u>readgood@pp.com</u>	0aergja
Dennis	M	28	<u>imagod@pp.com</u>	1bjb9a93

```
SELECT Age FROM Users WHERE Name='Dee'; 28
```

```
SELECT Age FROM Users WHERE Name='Dee' OR Name='Mac'; 28, 7
```

```
UPDATE Users SET email='readgood@pp.com'  
WHERE Age=32; -- this is a comment
```

```
INSERT INTO Users Values('Frank', 'M', 57, ...);
```

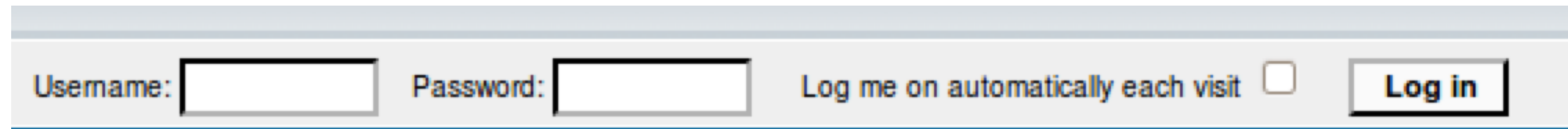
```
DROP TABLE Users;
```

# Some SQL Syntax

- `SELECT * FROM table`
  - The asterisk (\*) is shorthand for “all columns.” Select all columns from the table, keeping all rows.
- `WHERE` can be used to filter out certain rows
  - Arithmetic comparison: `<`, `<=`, `>`, `>=`, `=`, `<>`
  - Arithmetic operators: `+`, `-`, `*`, `/`
  - Boolean operators: `AND`, `OR`
  - `AND` has precedence over `OR`

# Server-side code

## Website



Username:  Password:  Log me on automatically each visit

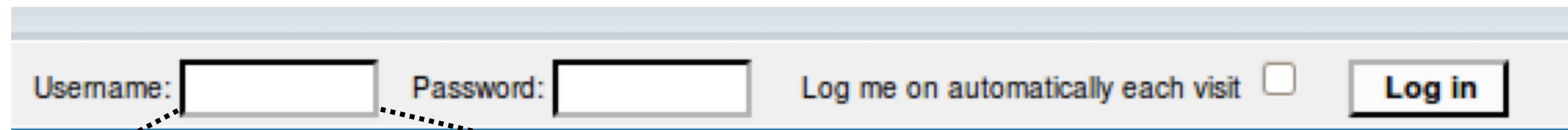
## “Login code” (php)

```
$result = mysql_query("select * from Users  
                        where(name=' $user' and password=' $pass' );");
```

Suppose you successfully log in as \$user  
if this query returns any rows whatsoever

**How could you exploit this?**

# SQL injection




A screenshot of a web application's login interface. It features a light gray header bar with a blue border. On the left, there are two input fields: 'Username:' and 'Password:'. To the right of the password field is a checkbox labeled 'Log me on automatically each visit'. Further right is a 'Log in' button. A dotted line connects the 'Log in' button to a box containing a SQL injection payload.

**frank' OR 1=1); --**

```
$result = mysql_query("select * from Users  
where(name=' $user' and password=' $pass ');");
```

```
$result = mysql_query("select * from Users where  
(name=' frank' OR 1=1); -- ' and password='x ');");
```

# SQL injection



A screenshot of a web login form. It features a 'Username:' label followed by a text input field, a 'Password:' label followed by another text input field, a checkbox labeled 'Log me on automatically each visit', and a 'Log in' button. A blue callout box with the word 'garbage' in blue text points to the password input field.

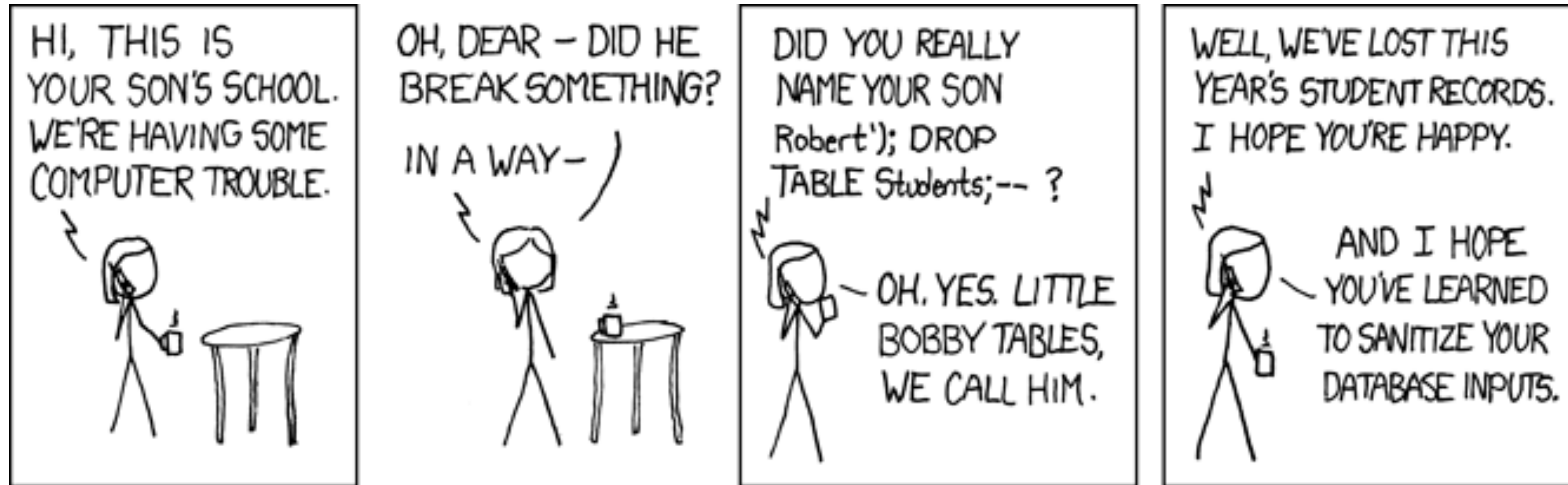
```
frank' OR 1=1); DROP TABLE Users; --
```

```
$result = mysql_query("select * from Users  
where(name='$user' and password='$pass');");
```

```
$result = mysql_query("select * from Users  
where(name='frank' OR 1=1);  
DROP TABLE Users; --  
' and password='garbage');");
```

**Can chain together statements with semicolon:  
STATEMENT 1 ; STATEMENT 2**

# Exploits of a Mom



[https://www.explainkcd.com/wiki/index.php/327:\\_Exploits\\_of\\_a\\_Mom](https://www.explainkcd.com/wiki/index.php/327:_Exploits_of_a_Mom)



A “Licence plate” with an SQL injection attack as a way to fight back traffic cameras.  
[https://www.reddit.com/r/geek/comments/1j9tn3/speed\\_camera\\_sql\\_injection/](https://www.reddit.com/r/geek/comments/1j9tn3/speed_camera_sql_injection/)

# SQL Injection Defense: Input Sanitization

- Block special characters: ' -- ;
- Allow: input within range, e.g., integer values for some fields
- Escape special characters: \; \'
  - Escape the escape? \\
- Secure escaper exists in SQL libraries
- May not be an effective solution, if we run SQL queries with raw user inputs



# What else can we do?

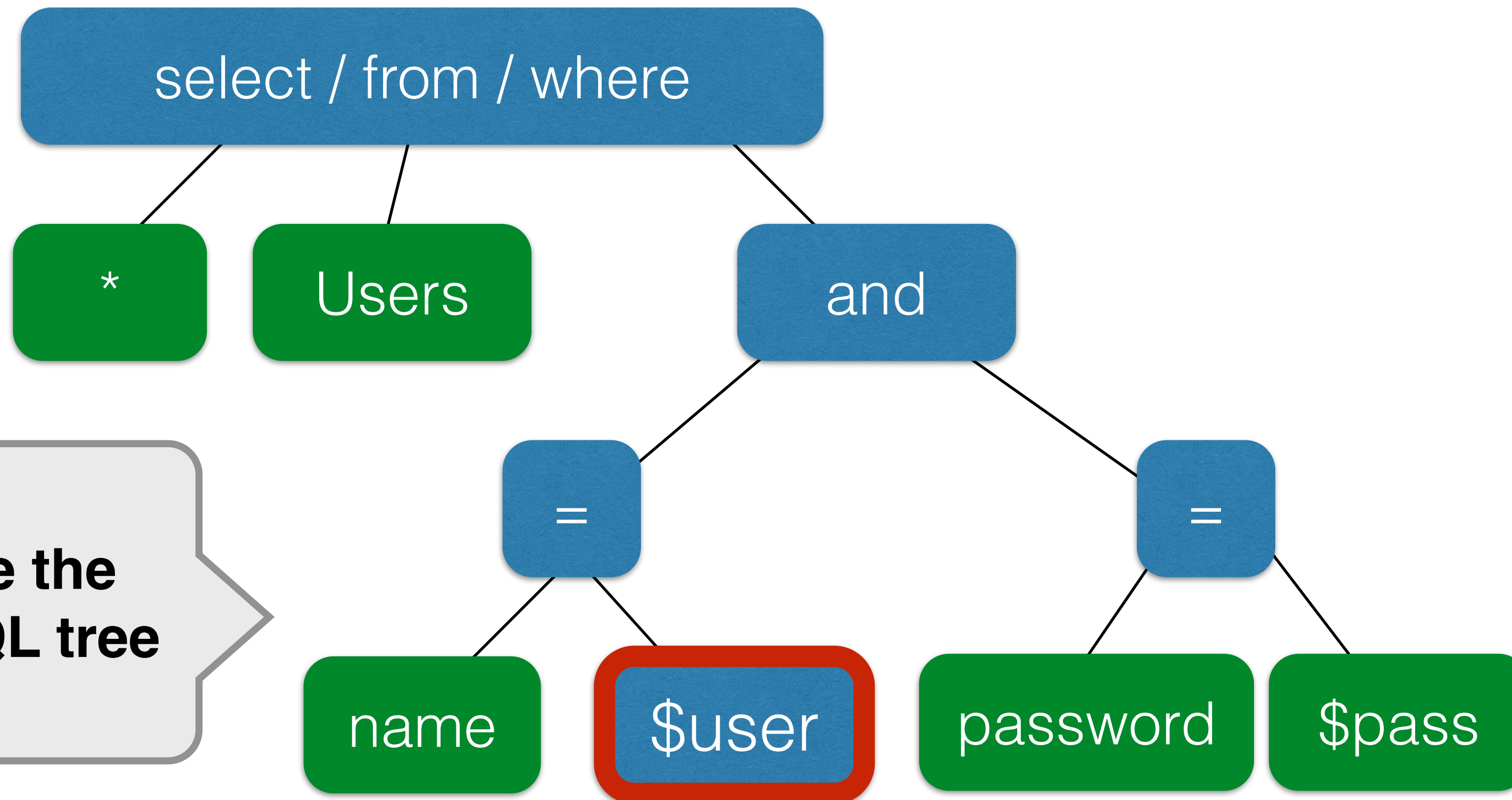
- Hint: data vs code
  - Separate data from instruction
- User input, SQL queries

# Parameterized SQL / Prepared Statements

- Idea: Parse the SQL query structure first, then insert the data
- The untrusted input cannot change the SQL query structure

# Example without Prepared Statements

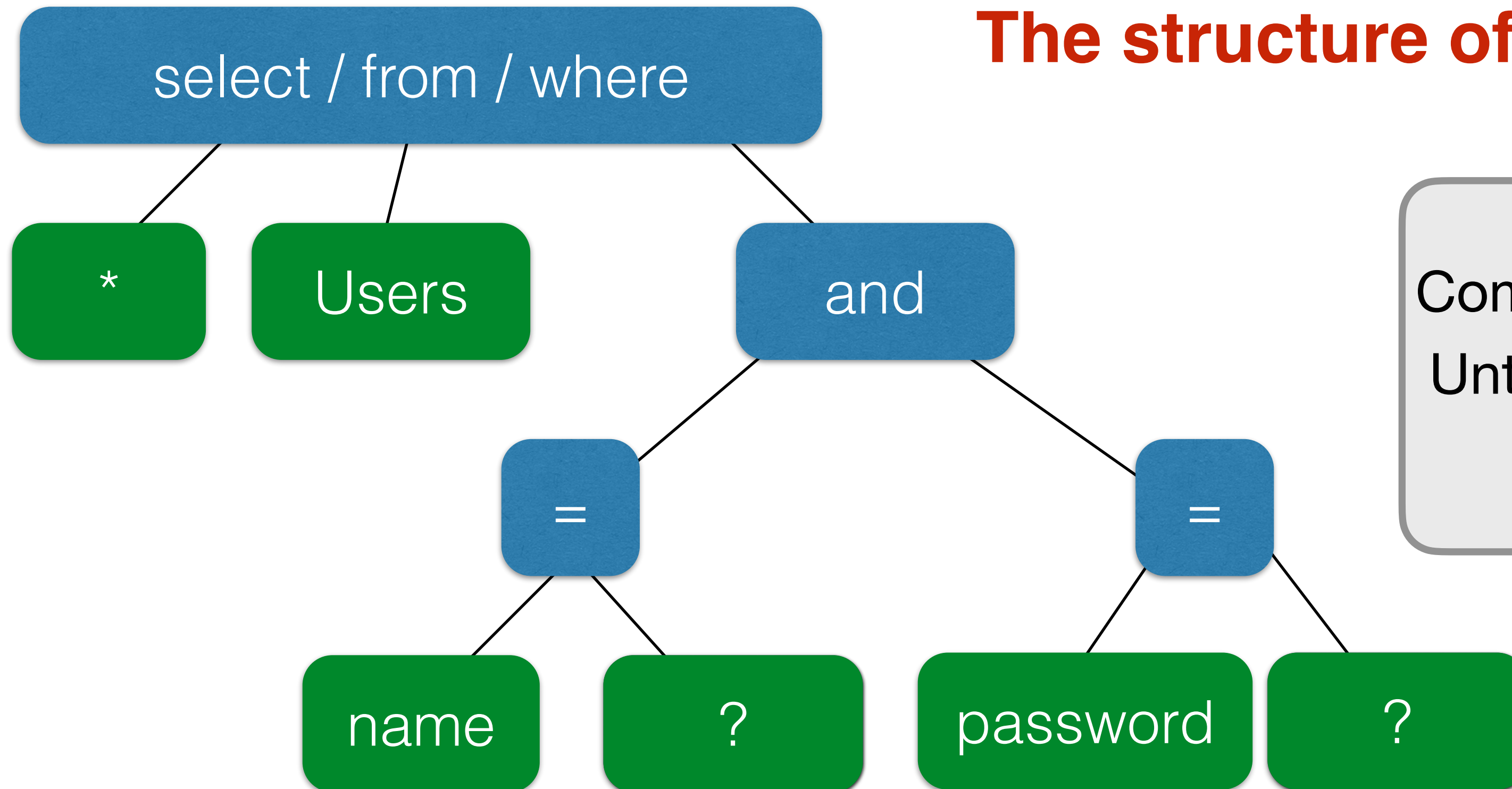
```
$result = mysql_query("select * from Users  
where(name=' $user' and password=' $pass' );");
```



**\$user** can change the structure of the SQL tree

# Prepared Statement Example

```
$statement = $db->prepare("select * from Users  
where(name=? and password=?);");
```



**The structure of the tree is *fixed***

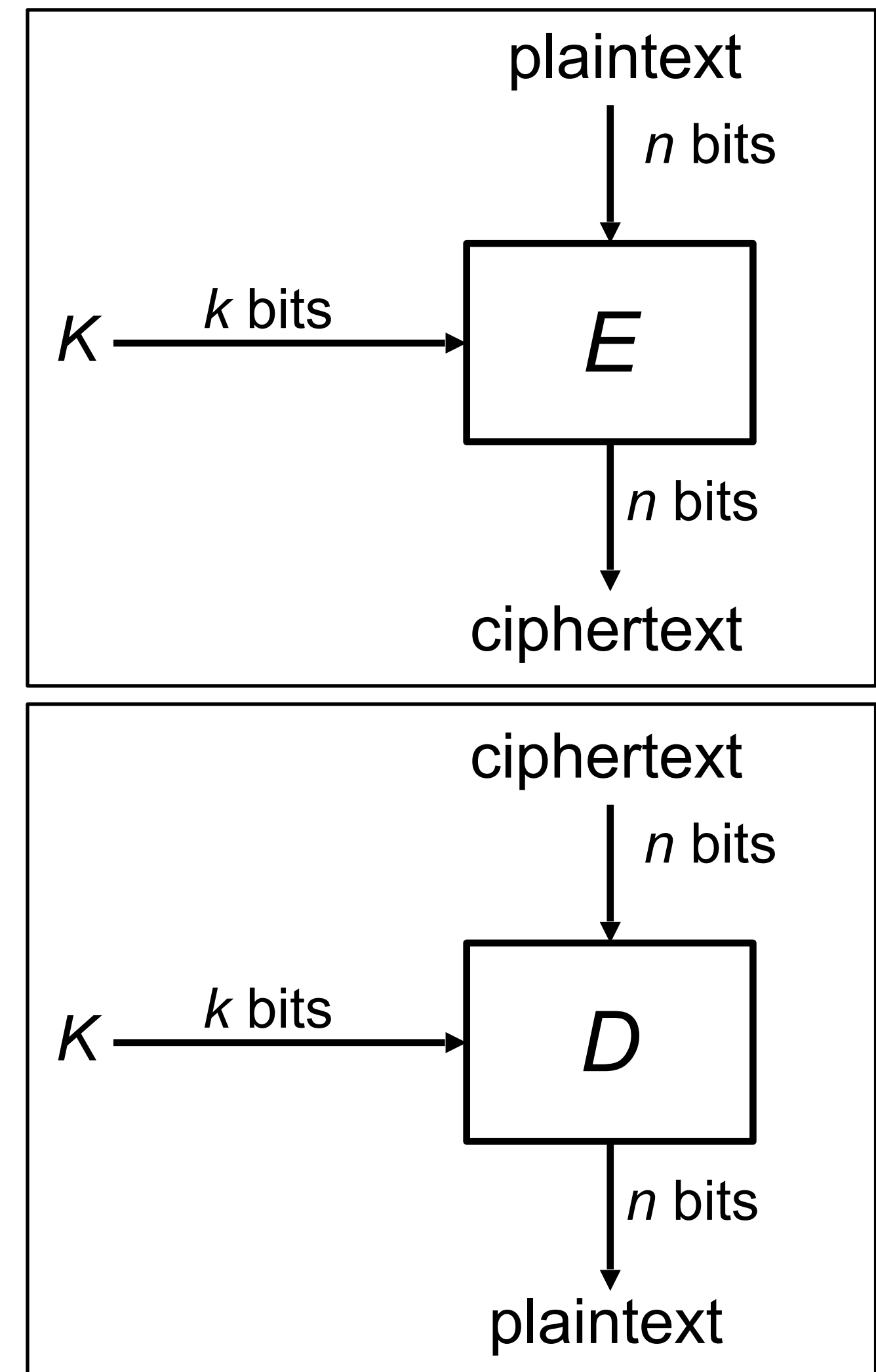
Compile first, bind data later  
Untrusted input will only be  
treated as data

# Example

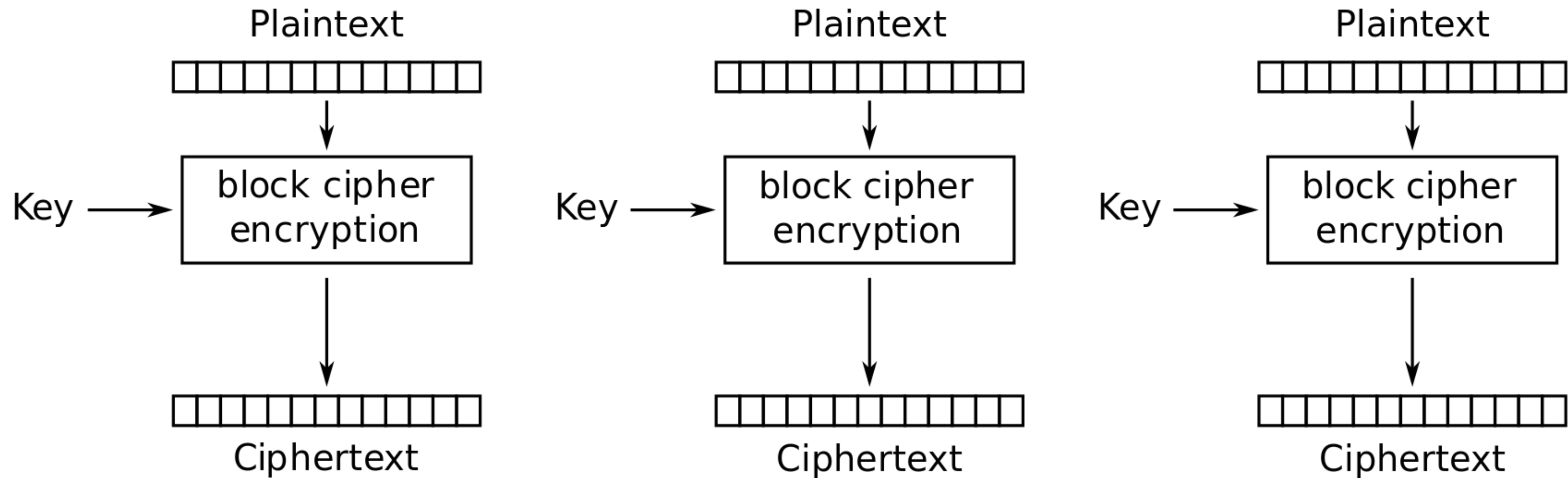
```
1 from flask import Flask, request, redirect
2 import os
3 import subprocess
4 import sys
5 import sqlite3
6
7 app = Flask(__name__)
8
9 # check if the email (GET variable) is in the database users and if so, unsubscribe by deleting it
10 @app.route('/unsubscribe')
11 def unsubscribe():
12     # get the email (GET variable)
13     email = request.args.get('email')
14     # get the database connection
15     db = sqlite3.connect("users.db")
16     # get the cursor
17     cursor = db.cursor()
18     # get the user from the database
19     cursor.execute("SELECT * FROM users WHERE email = '" + email + "'")
20     # get the user from the database
21     row = cursor.fetchone()
22     # if the email (GET variable) is in the database
23     if row:
24         # get the database connection
25         db = sqlite3.connect("users.db")
26         # get the cursor
27         cursor = db.cursor()
28         # delete the user from the database
29         cursor.execute("DELETE FROM users WHERE email = '" + email + "'")
```

# Block Ciphers: Definition

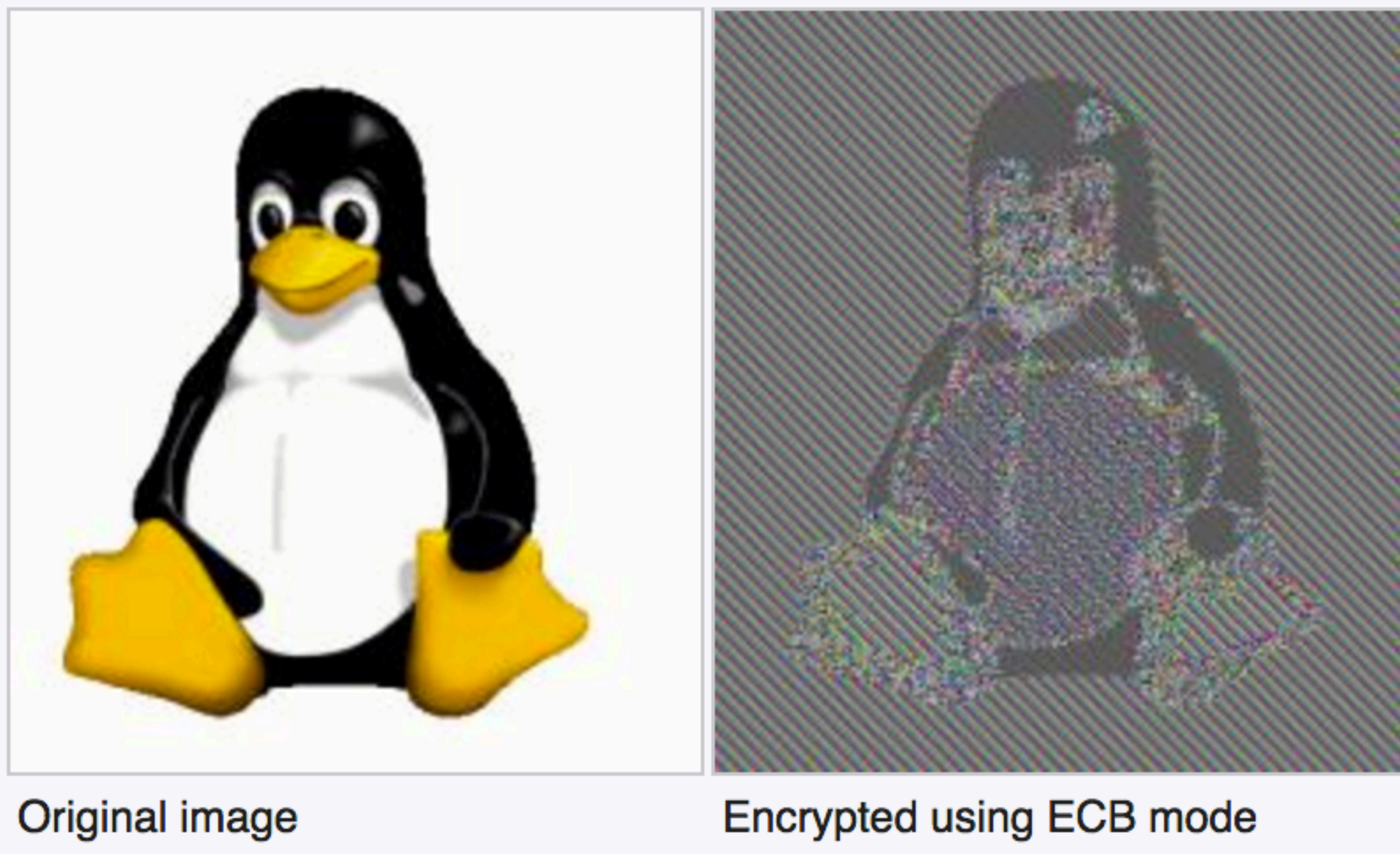
- **Block cipher:** A cryptographic scheme consisting of encode/decode algorithms for a fixed-sized block of bits:
- $E_K(M) \rightarrow C$ : Encode
  - Inputs:  $k$ -bit key  $K$  and an  $n$ -bit plaintext  $M$
  - Output: An  $n$ -bit ciphertext  $C$
  - Sometimes written as:  $\{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$
- $D_K(C) \rightarrow M$ : Decode
  - Inputs: a  $k$ -bit key, and an  $n$ -bit ciphertext  $C$
  - Output: An  $n$ -bit plaintext
  - Sometimes written as:  $\{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$
  - The inverse of the encryption function



# ECB Mode



Electronic Codebook (ECB) mode encryption



**NEVER use ECB**  
(but over 50% of Android apps do)