

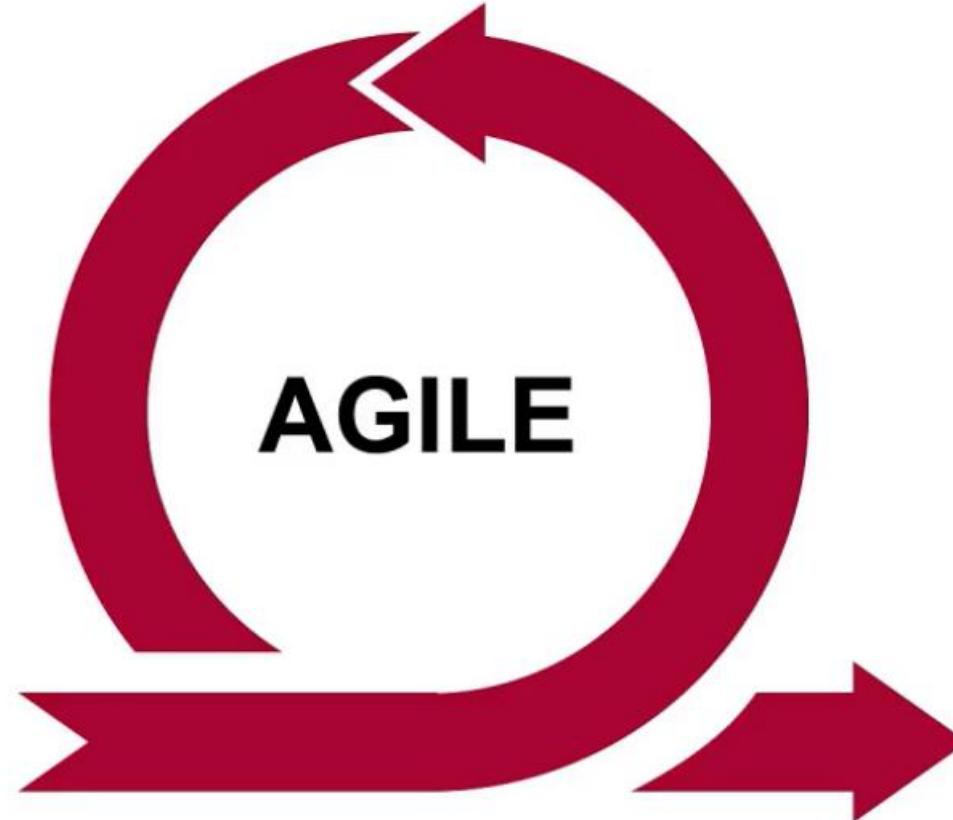


Professional Certificate in Coding: Full Stack Development with MERN: Week 10

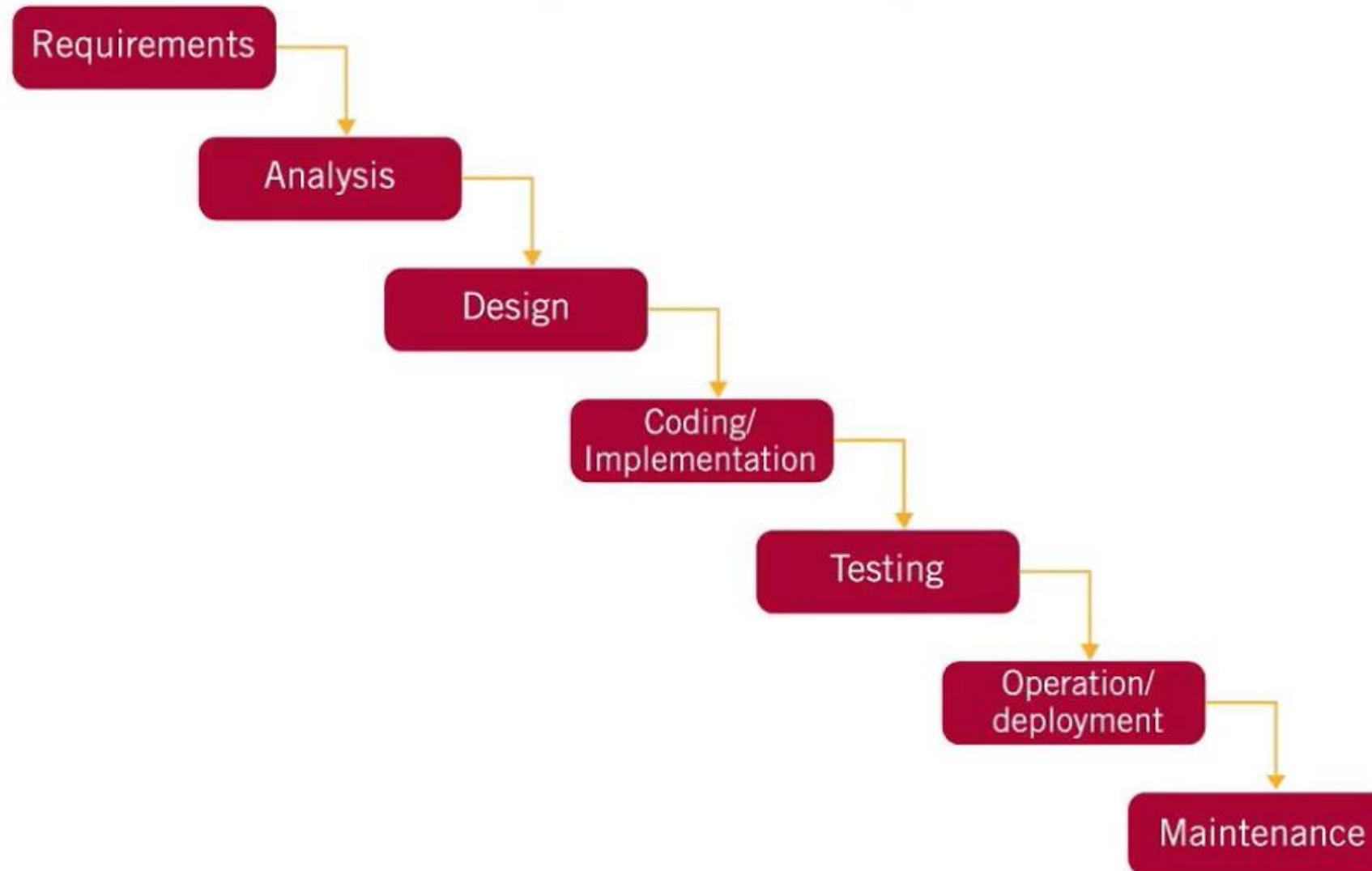
---

**Introduction To Cybersecurity And Recursion**

# Agile Methodology



# Waterfall Model

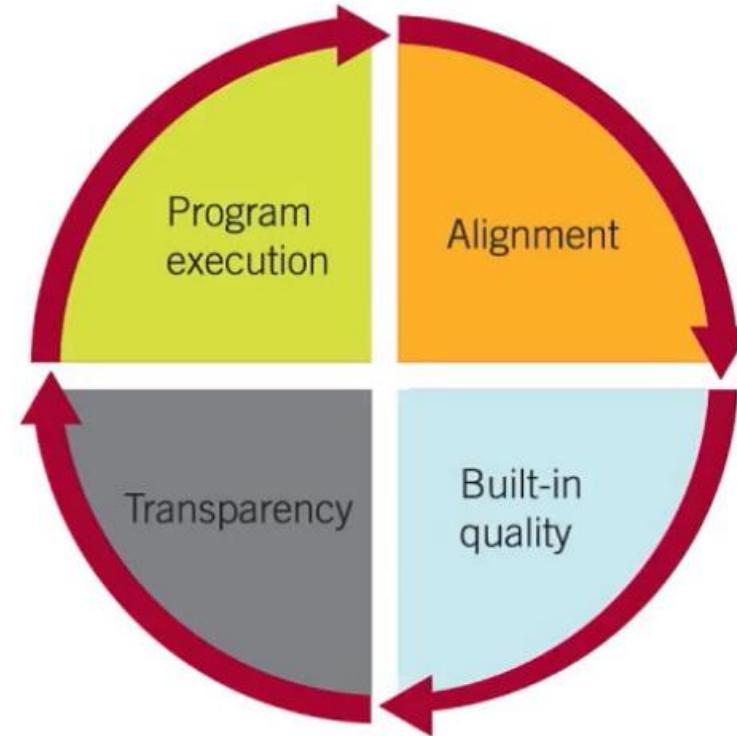


# Agile Principles

- 1 Satisfy the customer through continuous delivery
- 2 Welcome changing requirements
- 3 Deliver working software
- 4 Collaborate with business leadership
- 5 Built projects around motivated individuals
- 6 Face to face conversation
- 7 Measure progress by working software
- 8 Maintain a constant development pace
- 9 Focus on technical excellence and good design
- 10 Keep it simple with clear scope
- 11 Self organizing teams produce the best software
- 12 Reflection improves team performance

## Agile Values

- ① Individual and interactions over processes and tools
- ② Working software over comprehensive documentation
- ③ Customer collaboration over contract negotiation
- ④ Responding to change over following a plan



- User Stories
  - Add card, Conversation, Confirmation
- Define some problem in the real world e.g. booking an airplane ticket
  - Upselling, cross-selling (booking hotel, car rental, ...)
  - Updating flight times, seats ...
  - Cancelling, refunds, invoice receipts, credits, air miles, ...

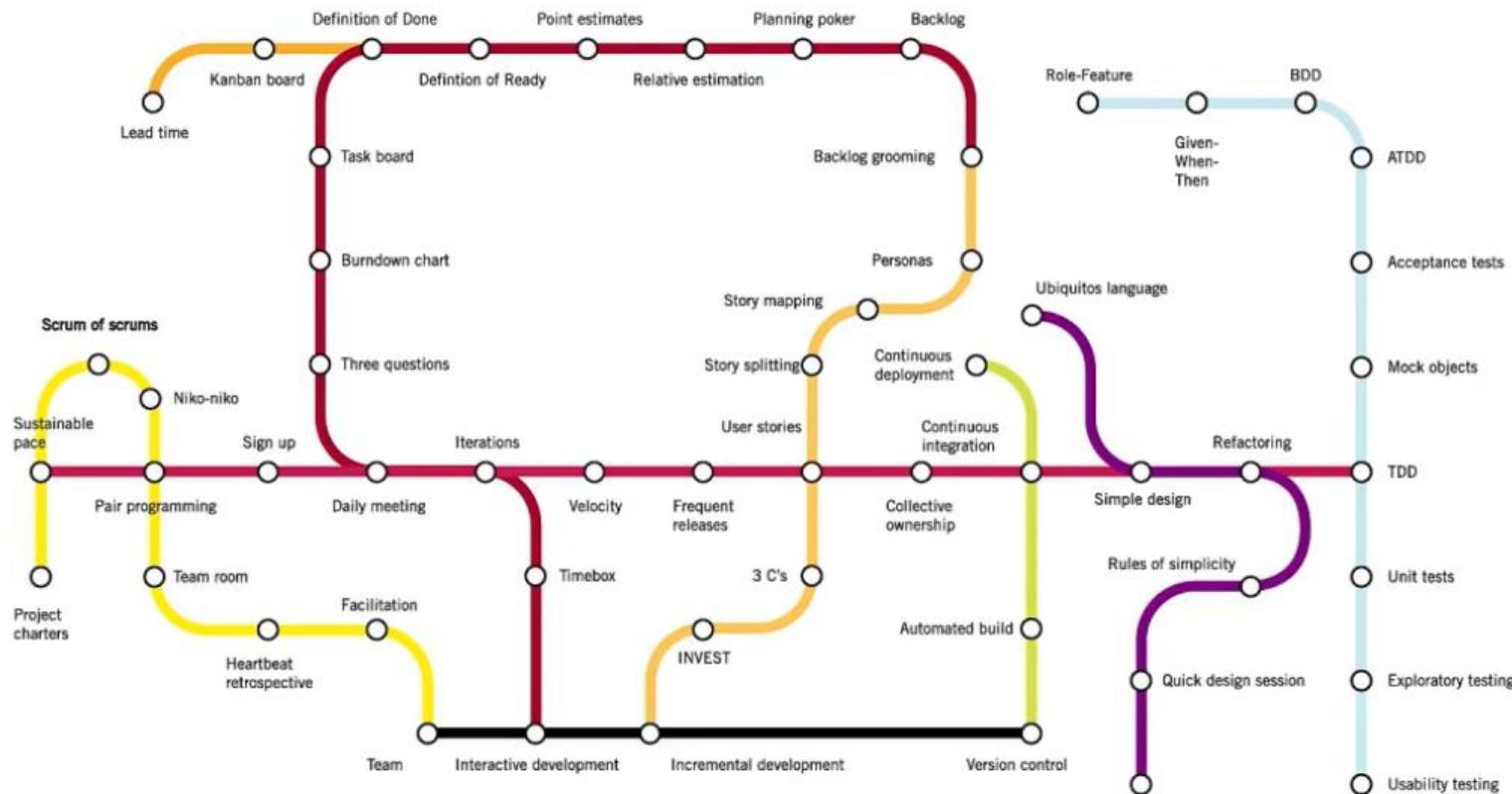
# Scrum



# Project Tradeoffs



# Various Agile “Tribes”



Lines represent practices from the various Agile “tribes” or areas of concern:

Extreme programming  
Teams  
Lean

Scrum  
Product management  
INVEST

Design  
Testing  
Fundamentals

# Authentication And Authorization (1/2)

Authentication:  
who are you?



## Authentication And Authorization (2/2)

Authorization: what are you allowed to access?



# Symmetric Encryption And Notation

Key = 12345

Key {Hello World} → c935af

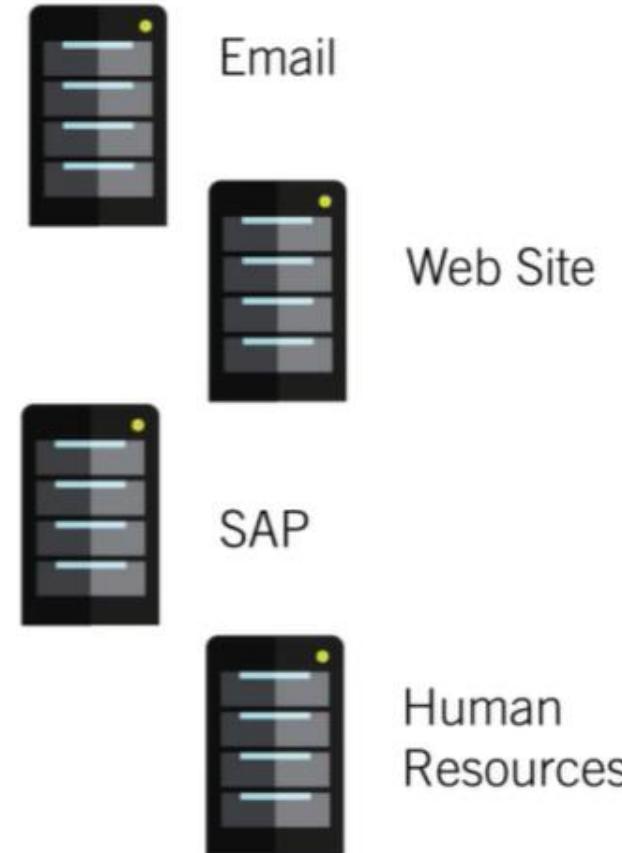
Key {c935af} → Hello World

# Multiple Users That Needs To Use Number Of Services

Users



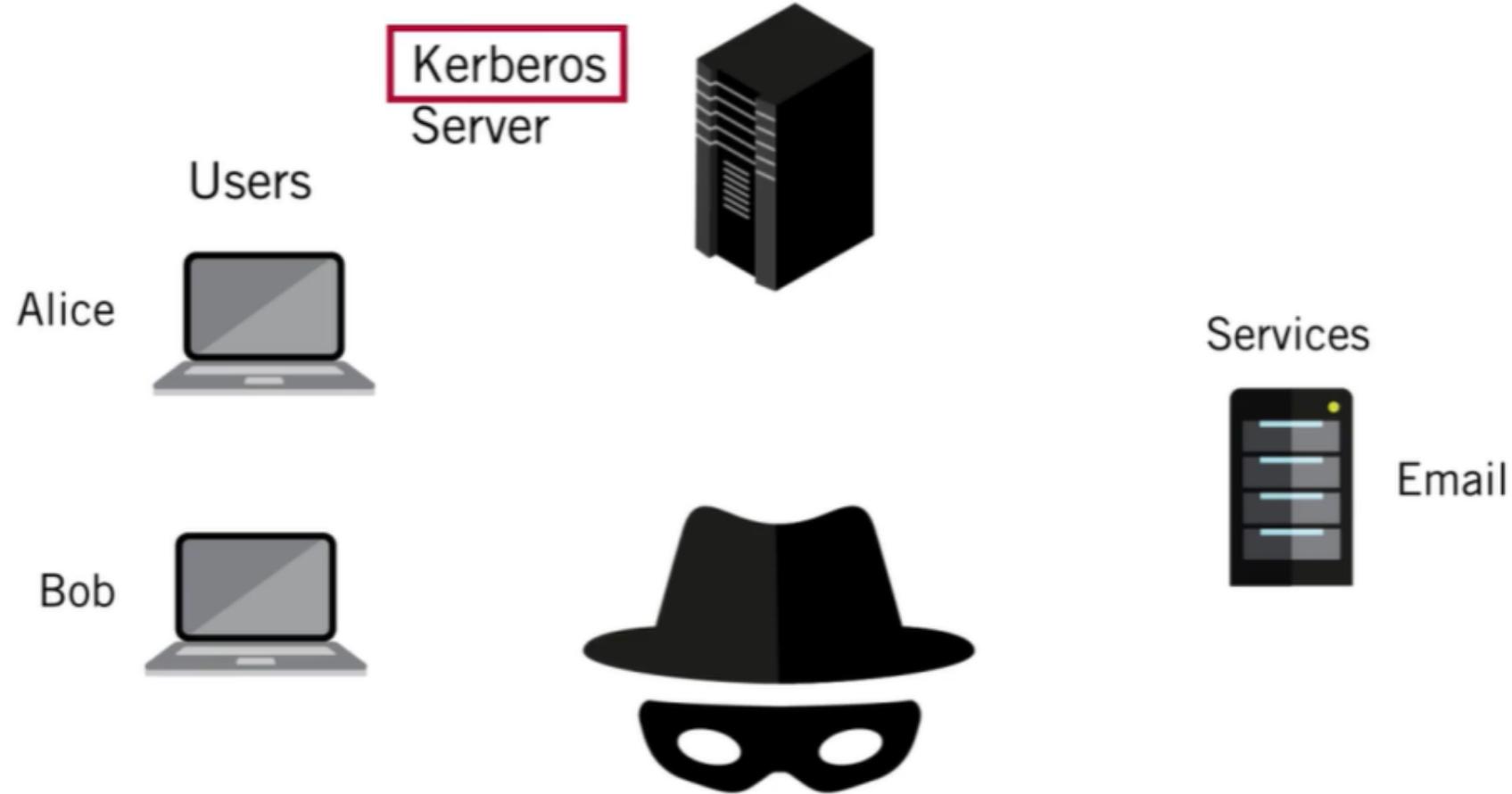
Corp Services



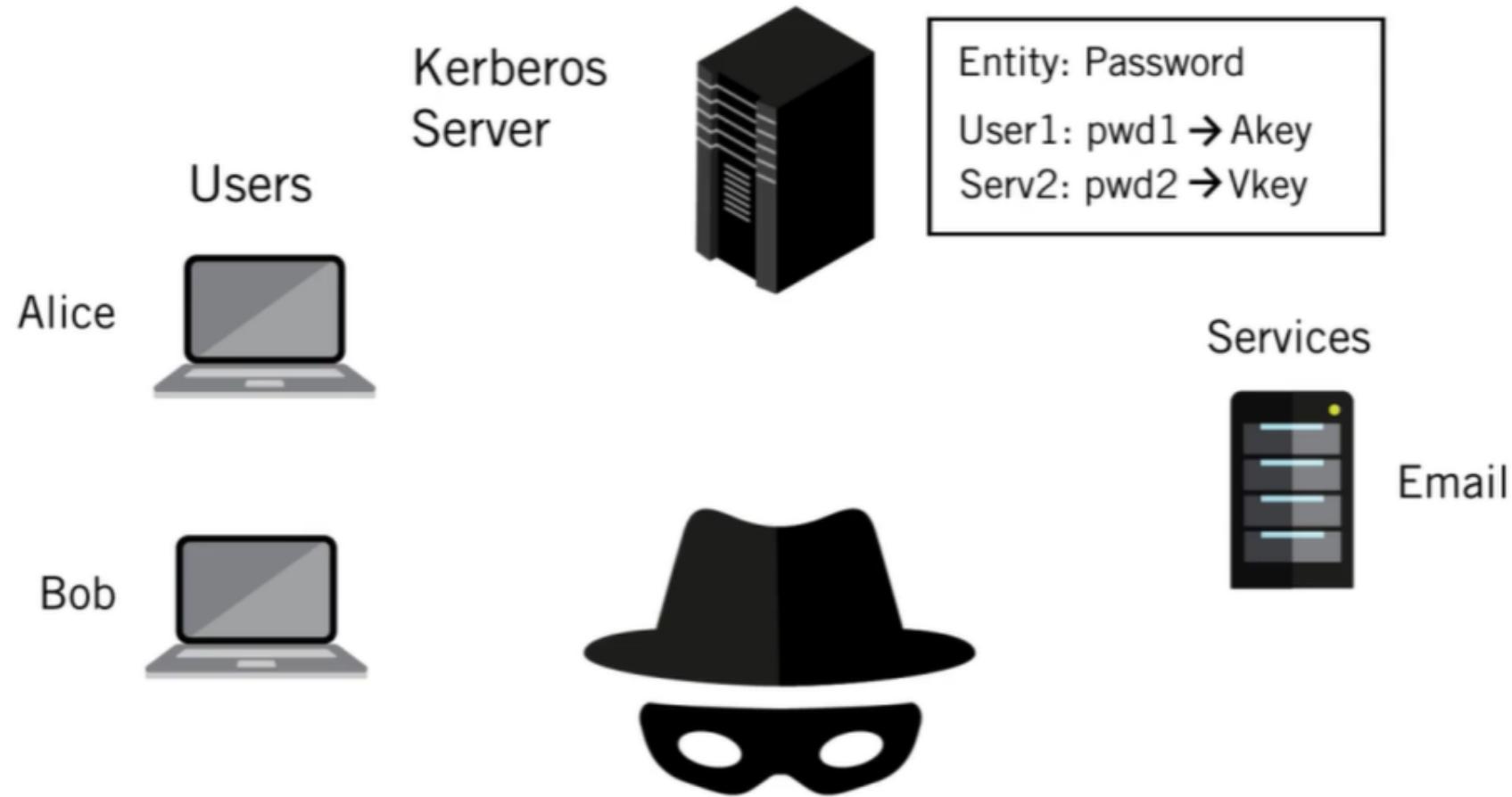
# Assumptions

- We do not trust the network to be secure.
- Attackers can see all messages and can copy and replay any message
- We would like single sign on (SSO) so users do not have to enter password multiple times
- Password should never be stored on the client machine
- Password should never be passed in cleartext on network

# Concept Of An Authentication Service That Users And Services Trust (1/2)



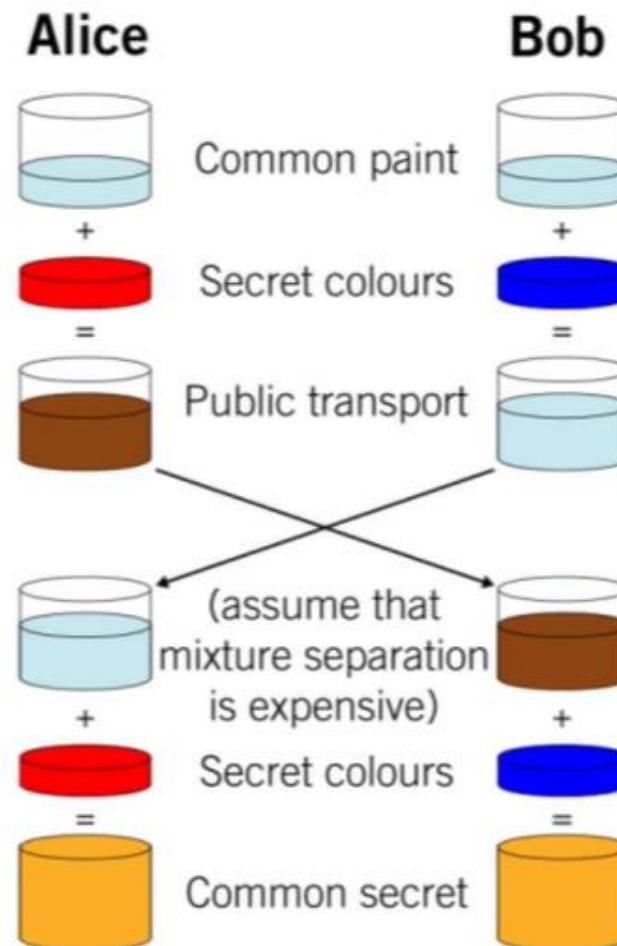
# Concept Of An Authentication Service That Users And Services Trust (2/2)



## Shared Secret Idea

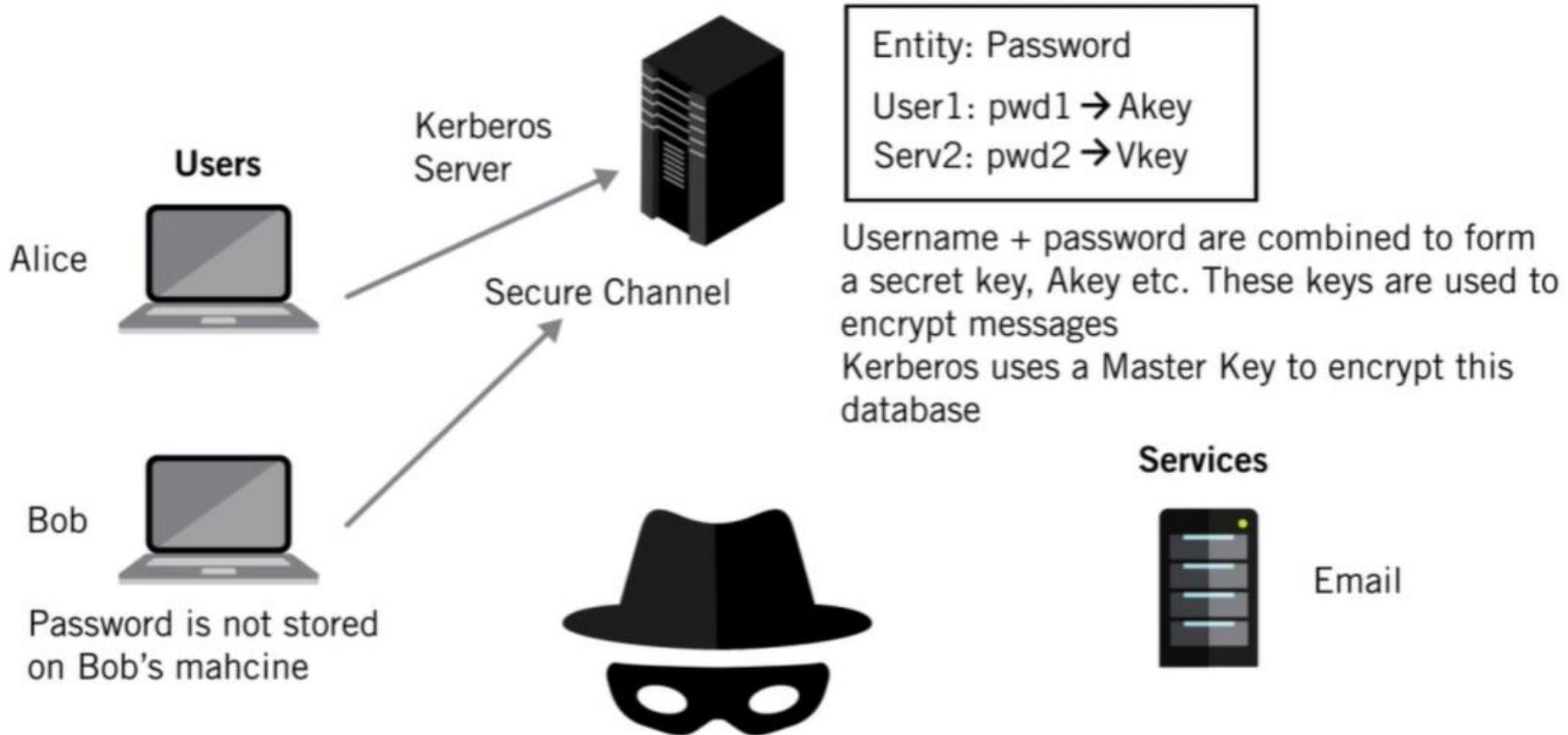
- User and Kerberos share a secret
- They use the secret to encrypt messages
- Kerberos knows a message is from Alice if Alice's secret decrypts the message
- Alice knows it's a legitimate Kerberos message if her secret decrypts the message
- Encrypted messages can be “copied” by attacker but cannot be decrypted

# Diffie-Hellman Key Exchange

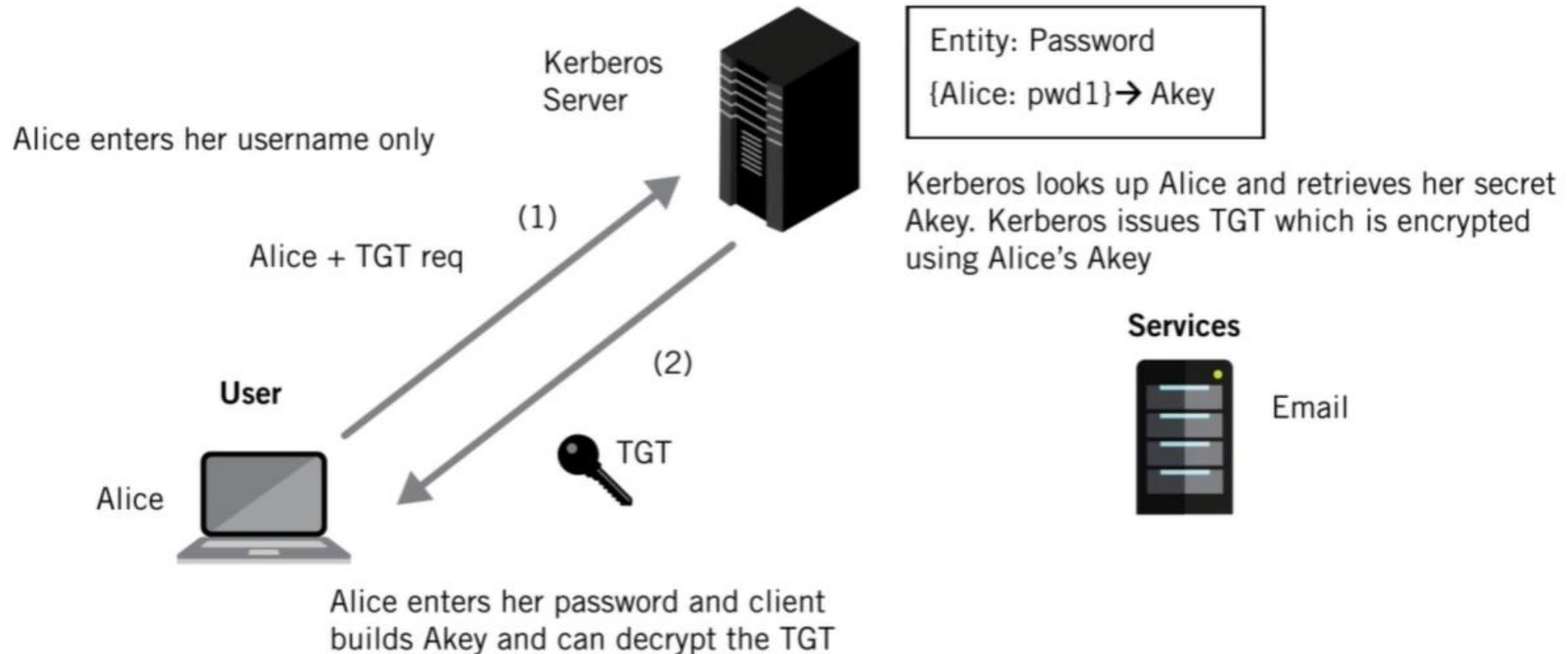


# Secure Channel To Set And Store The Password

A Secure Channel is used to initially set the password and store in Kerberos database eg Diffie-Hellman

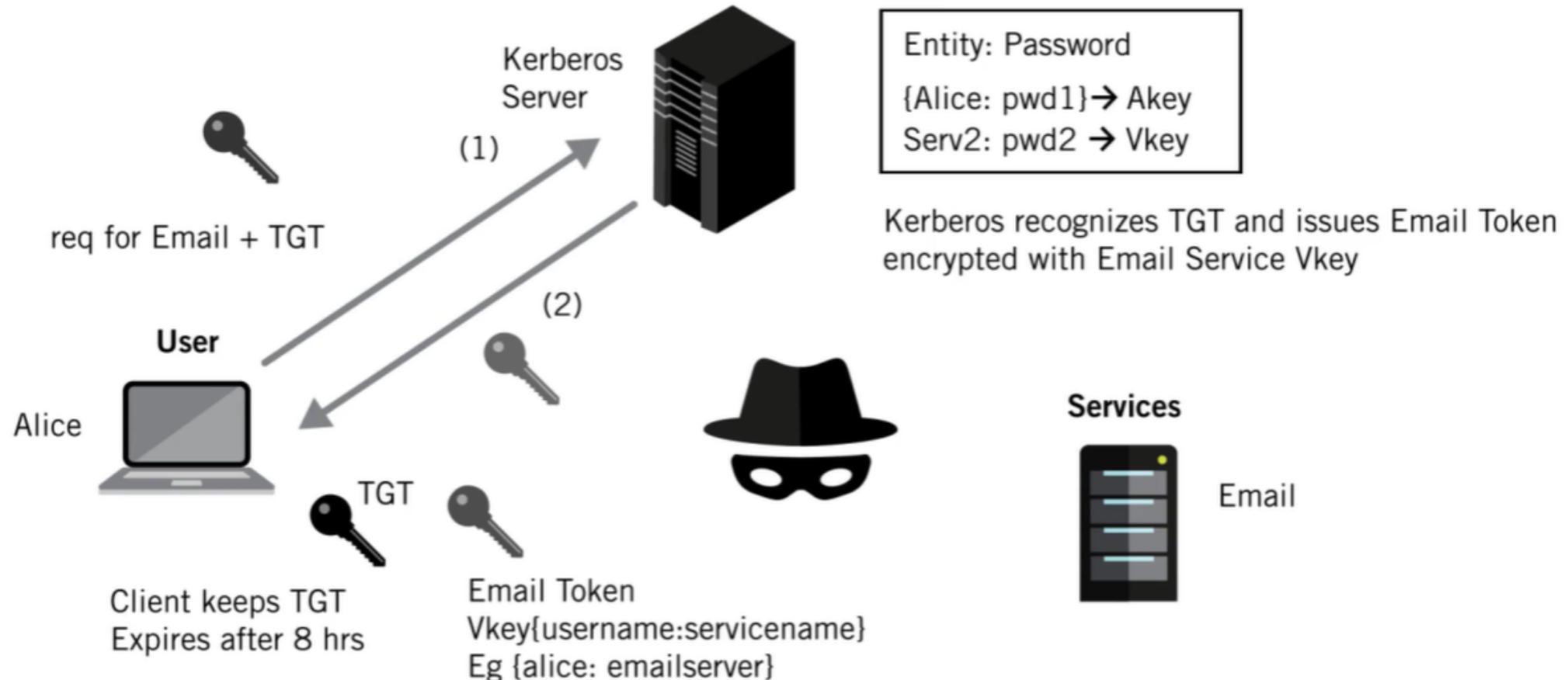


# Alice Gets A Ticket-Granting-Ticket (TGT)

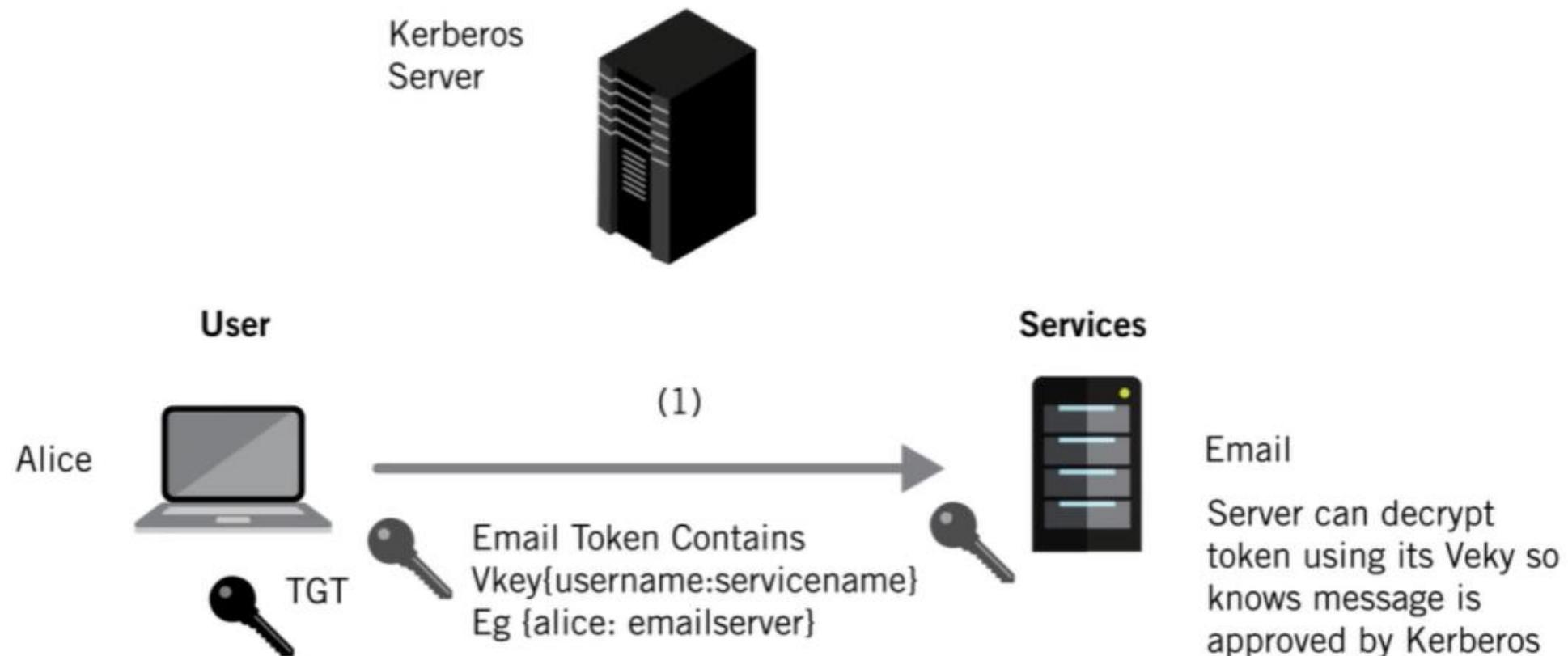


# Alice Requests Access To Email

TGT is now used rather than Password. Alice requests access to Email

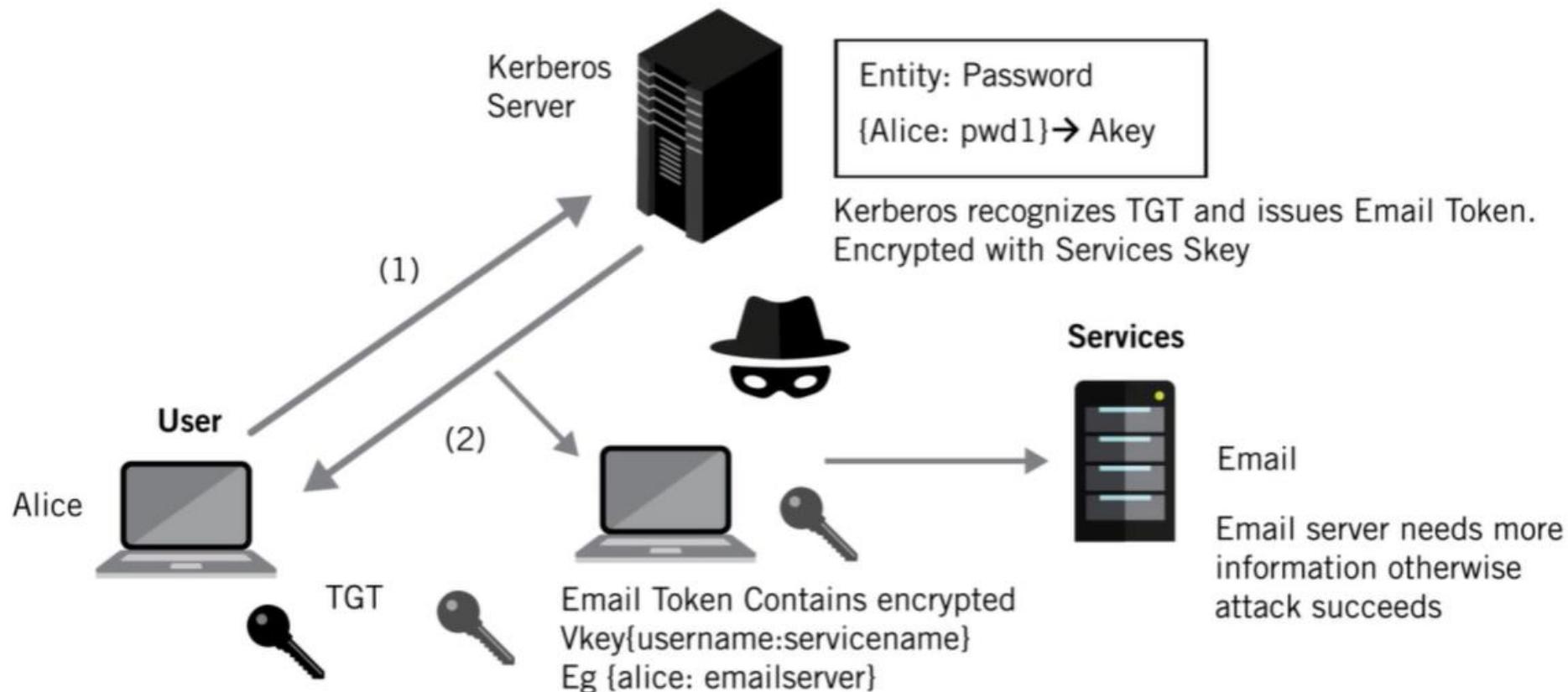


# Alice Sends Email Token To Server



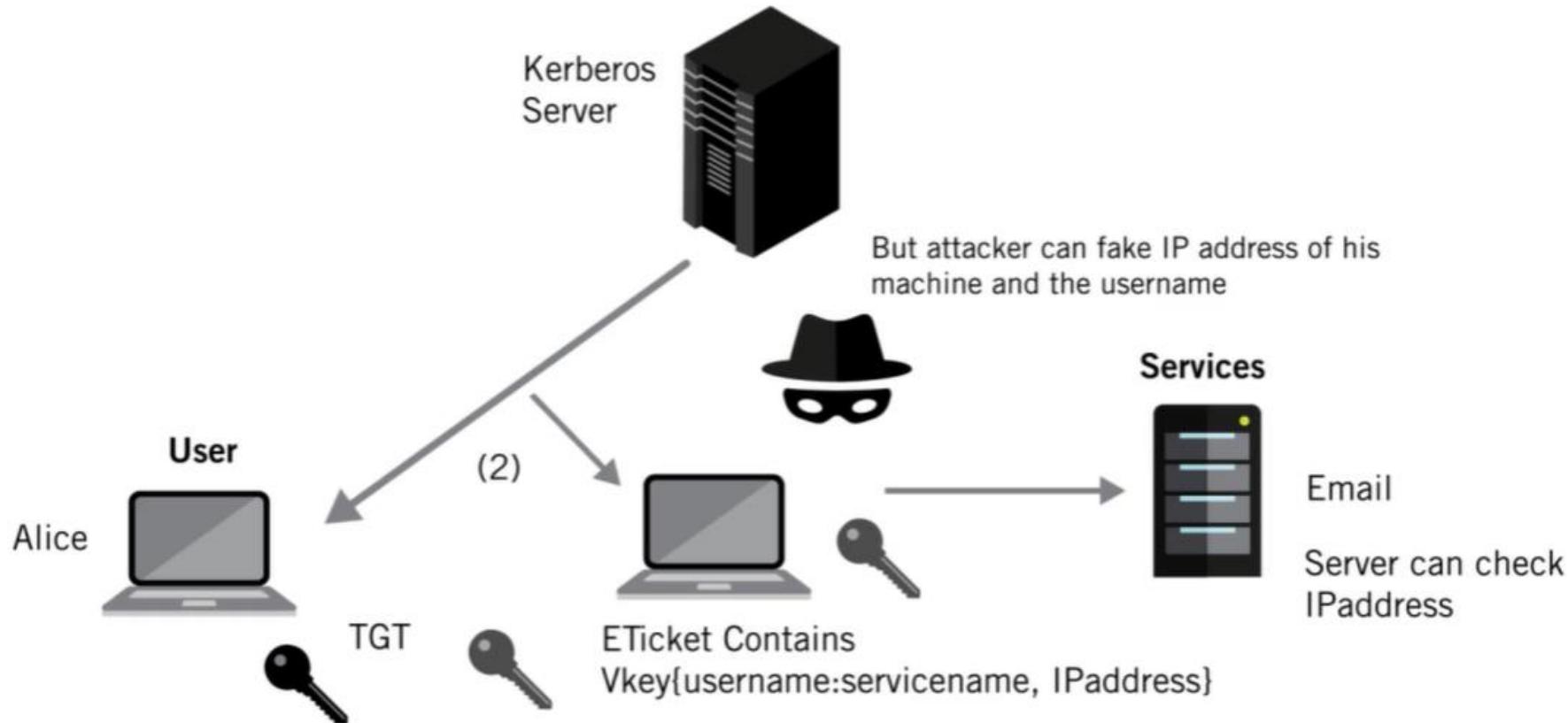
# What If The Attacker Steals The Encrypted Message?

Suppose Attacker copies Email Token then sends it on to Email Server pretending to be Alice

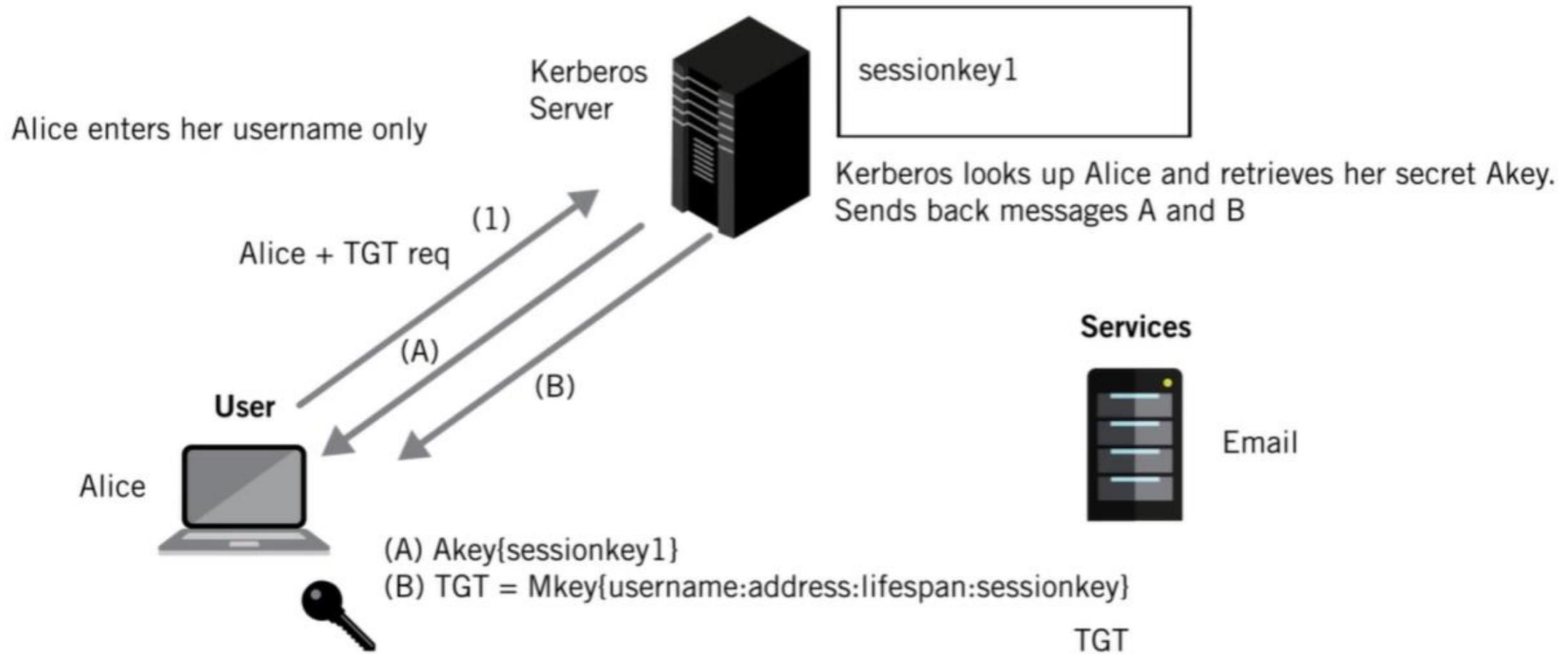


# What If The Attacker Fakes The IP Address As Well?

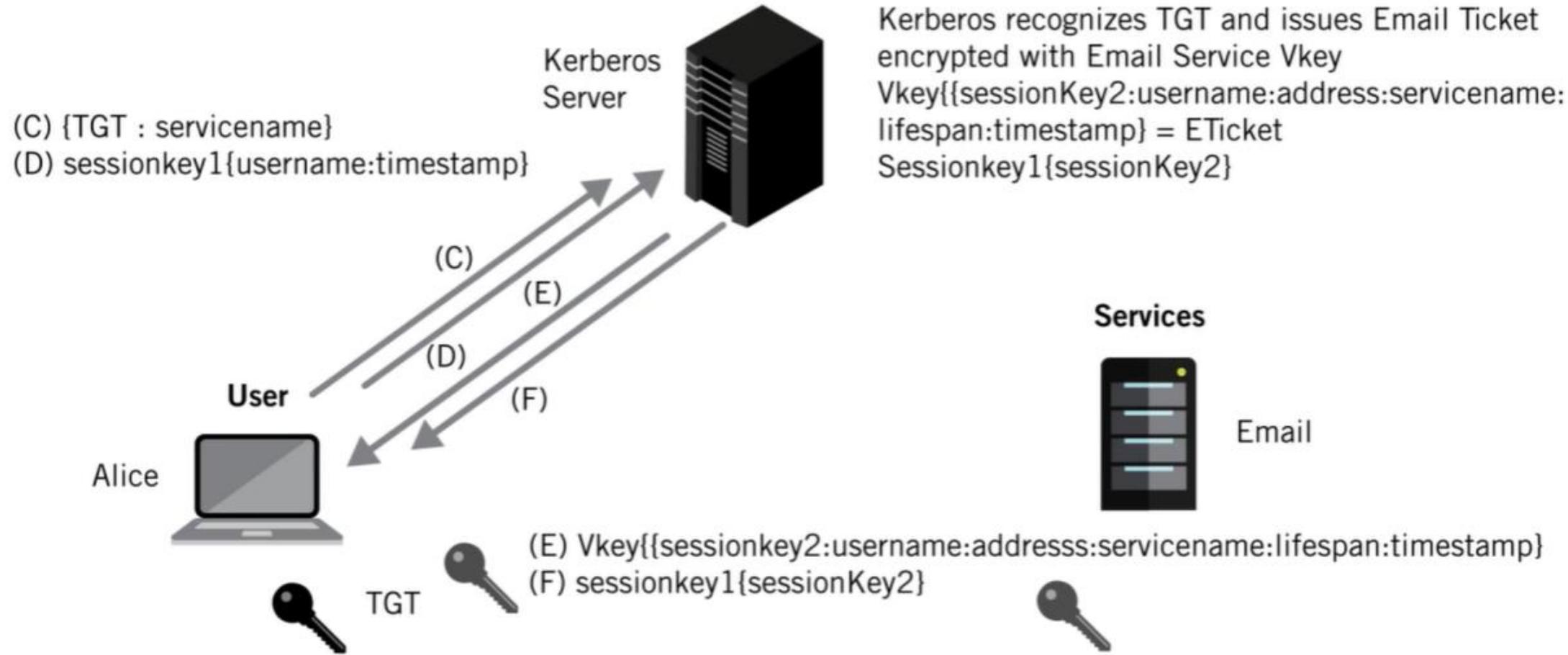
Suppose Attacker copies ETicket then sends it on to Email Server pretending to be Alice



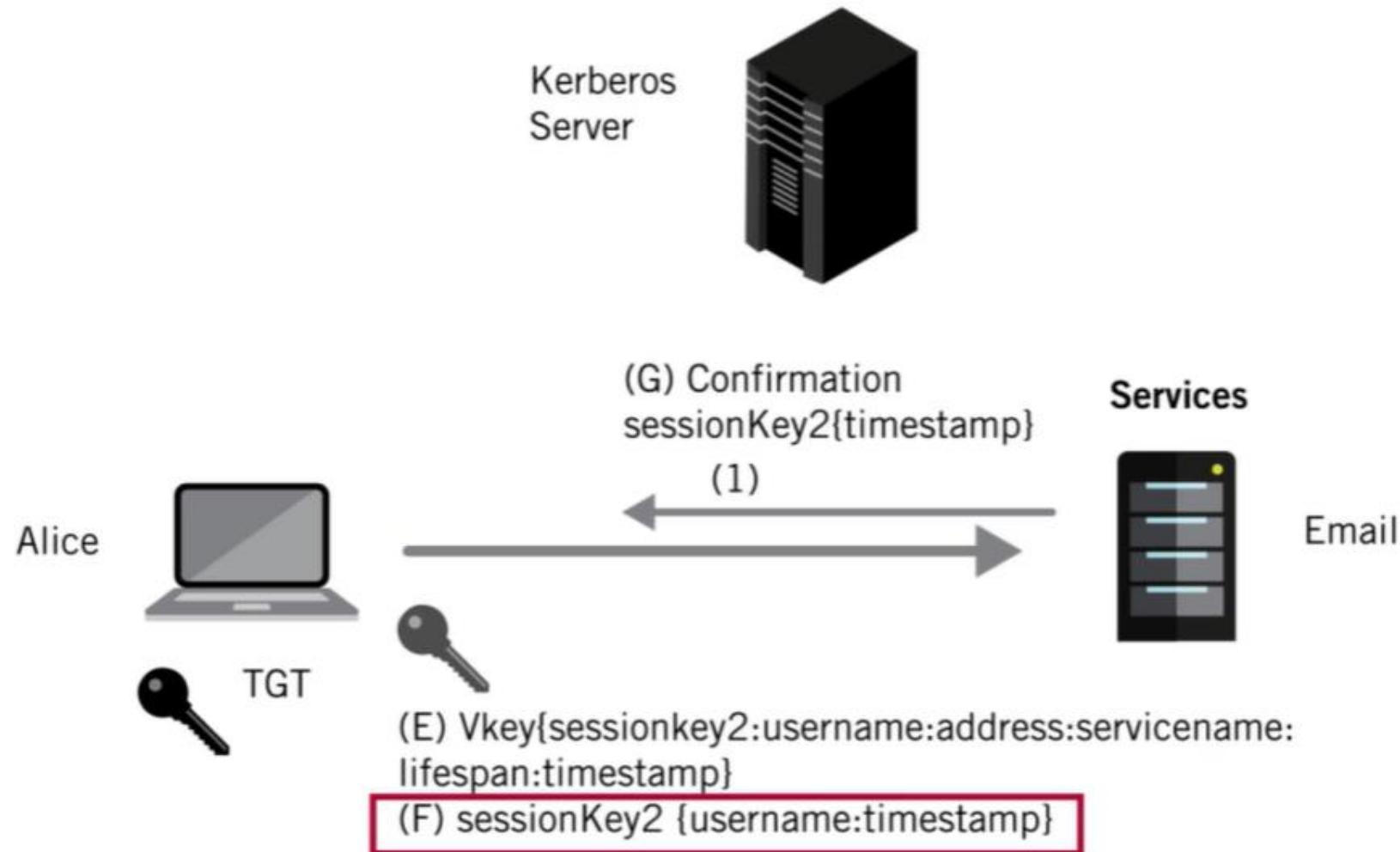
# Alice Requests A Ticket-Granting-Ticket (TGT)



# Alice Requests Email Server Access (1/2)



## Alice Requests Email Server Access (2/2)



# Authenticator

**Sent from to Alice to kerberos**

{username: TGT request} clear text

**Sent** from Kerberos Ticket Server to Alice at Login

Akey { sessionKey1 : TGT } Alice enters her password and client generates Akey

**Sent** from Alice to Kerberos requesting Email Ticket

sessioniKey1 {user: address : servicename}

**Sent** from to Kerberos to Alice

sessionKey1 {sessionKey2: Vkey{ETicket} }

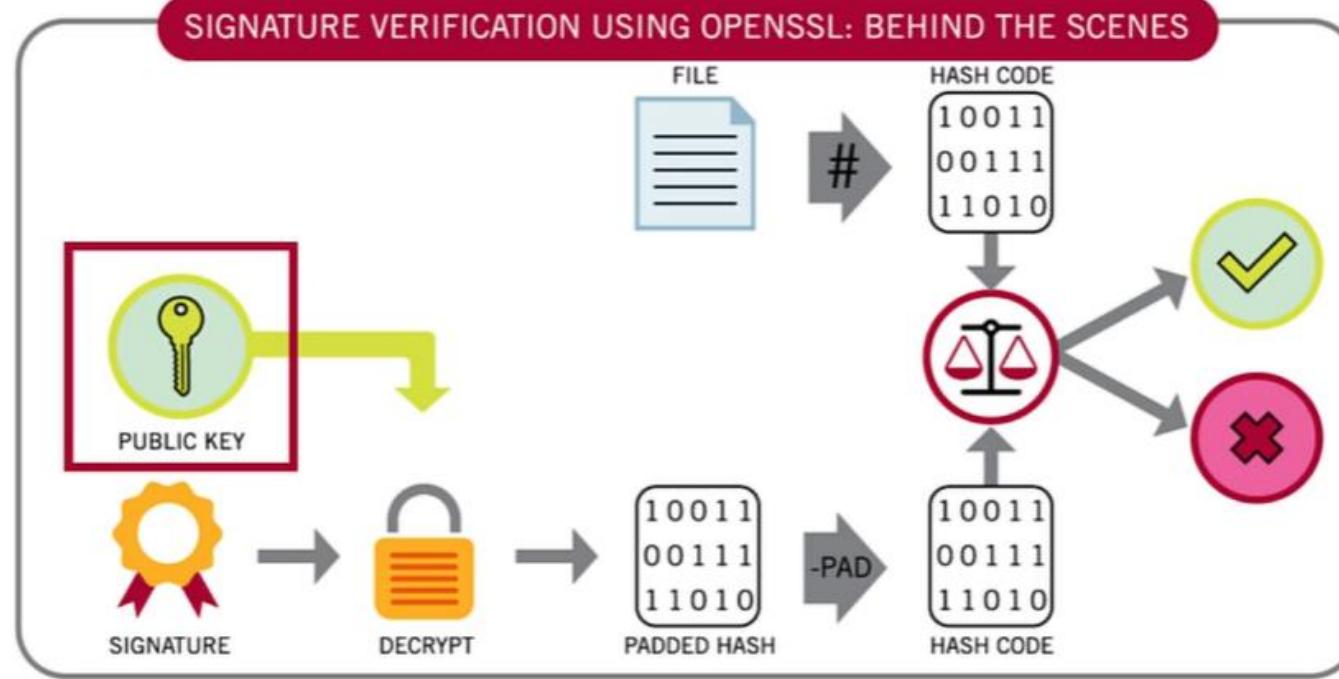
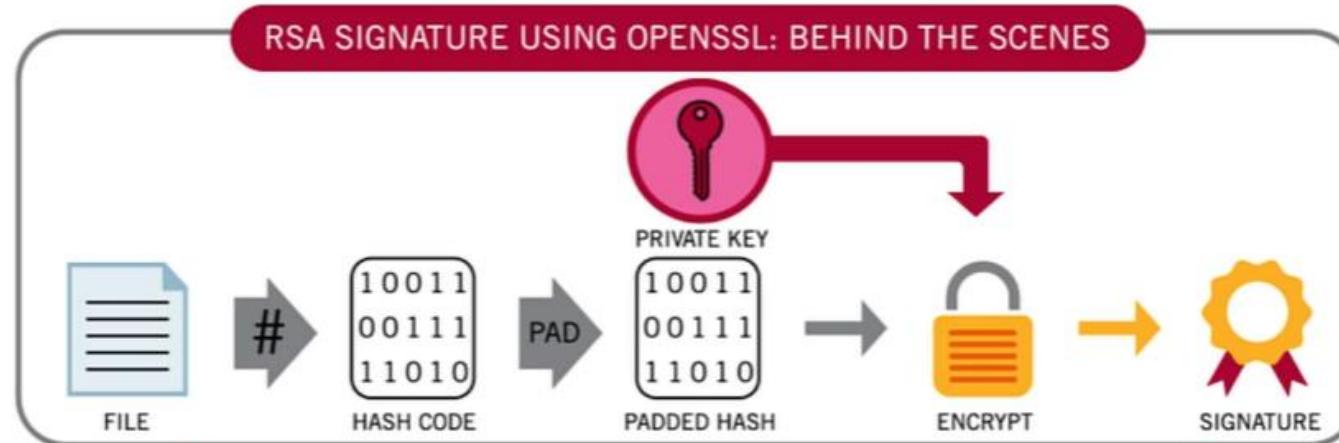
ETicket = {sessionkey2:username:address:servicename:lifespan:timestamp}

**Sent** from Alice to Email Server

sessionKey2{ username:address } + Vkey{ETicket}



# Generate A Key (1/3)



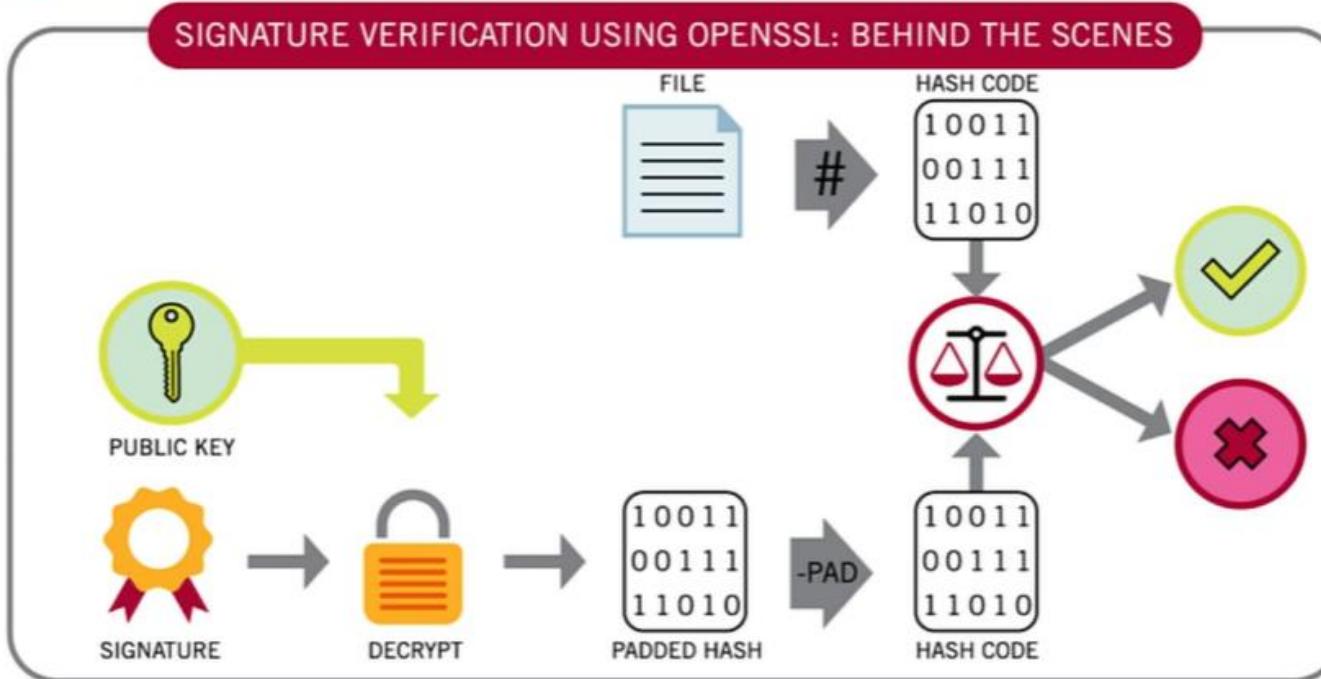
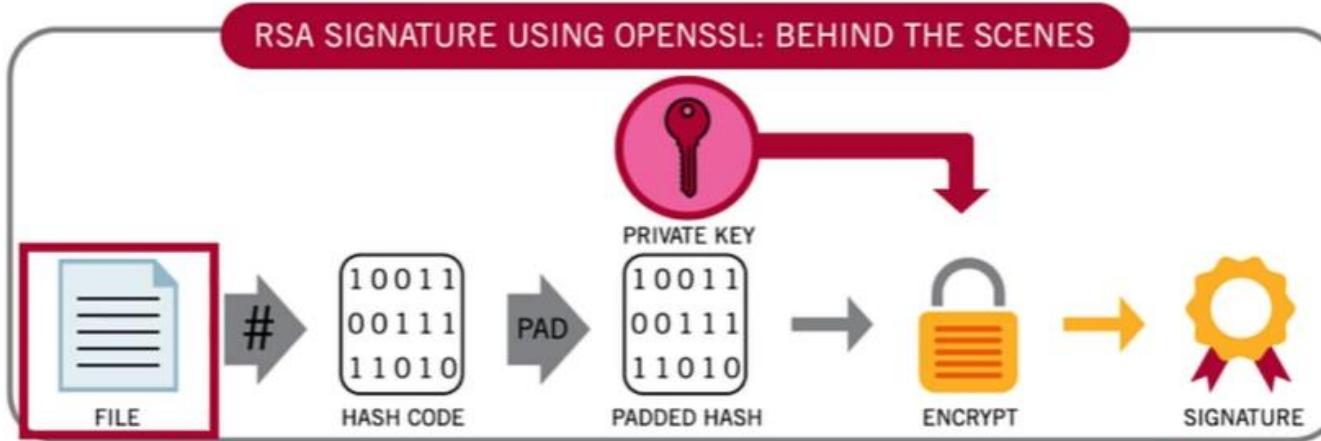
## Generate A Key (2/3)

```
[~] ~/MIT/Courses/OAuth2/Keys — johnwilliams@Johns-MBP — ..s/OAuth2/Keys — -zsh — 88x25
(base) Keys > openssl genrsa -out myprivate.pem 512
Generating RSA private key, 512 bit long modulus (2 primes)
.....+++++
..+++++
e is 65537 (0x010001)
(base) Keys > cat myprivate.pem
-----BEGIN RSA PRIVATE KEY-----
MIIB0gIBAAJBALyD4ker3ZzLbm5v1F6pgF+ZmrDzY/gAI+F3/JeHg9Ffw9qiNgJL
l5iLrMTxoIiwBaFu2MYDTliTHAAw/r4pkZUCAwEAAQJAhrh/T566ZqrgPC8NNL/6
o5msk9mWGPws3Llo+GilGqFw9sZtyyfI2IG5KbH6ZyJyCp901gA24Szrp104MD0k
AQIhAN3hPkHjP53bKcLEyUlf2w7wu7otE4MxBfxRJQ2zgcNhAiEA2YEr2Z7RM3Pg
b5NarNmEx5Zla/D60TEMXLvQUNOvLrUCICCKxKLNPatdVYanbg5A7NQIIIsBvlAkN
y53+Cr0zsriBAiEAwVVuMtxWvFON9d+XQ/l9ay0mN1JwOSv9/xf6zT28g9kCIHzA
SvXN1uuS2bAJ5Zow/aIeMQ60+kguTXj60682n3tg
-----END RSA PRIVATE KEY-----
(base) Keys > [REDACTED]
```

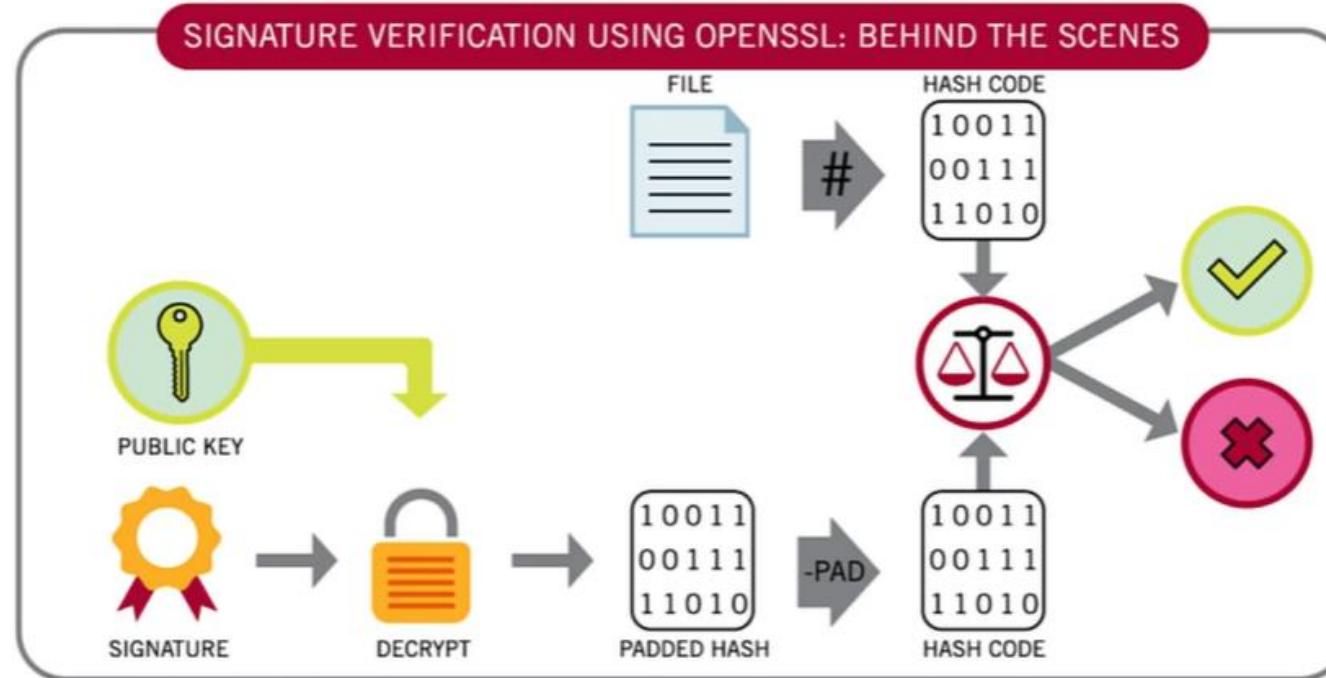
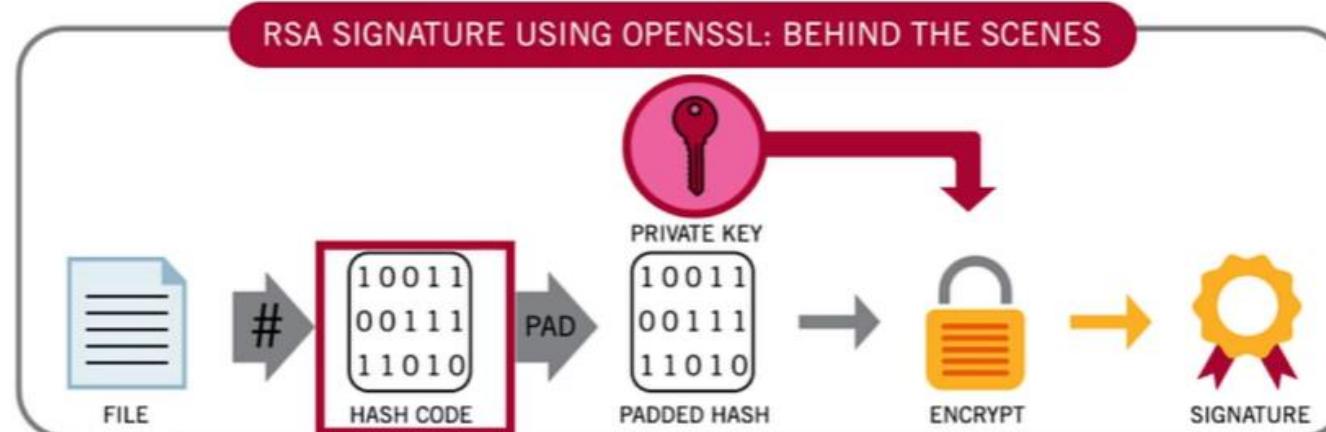
## Generate A Key (3/3)

```
~ MIT/Courses/OAuth2/Keys — johnwilliams@Johns-MBP — ..s/OAuth2/Keys — -zsh — 88x25
(base) Keys > openssl rsa -in myprivate.pem -pubout > mypublic.pem
writing RSA key
(base) Keys > ls
hello.txt      myprivate.pem mypublic.pem
(base) Keys > cat mypublic.pem
-----BEGIN PUBLIC KEY-----
MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBALyD4ker3ZzLbm5v1F6pgF+ZmrDzY/gA
I+F3/JeHg9Ffw9qiNgJLl5iLrMTxoIiwBaFu2MYDTliTHAAw/r4pkZUCAwEAAQ==
-----END PUBLIC KEY-----
(base) Keys > ls
hello.txt      myprivate.pem mypublic.pem
(base) Keys > cat hello.txt
Hello World
(base) Keys >
```

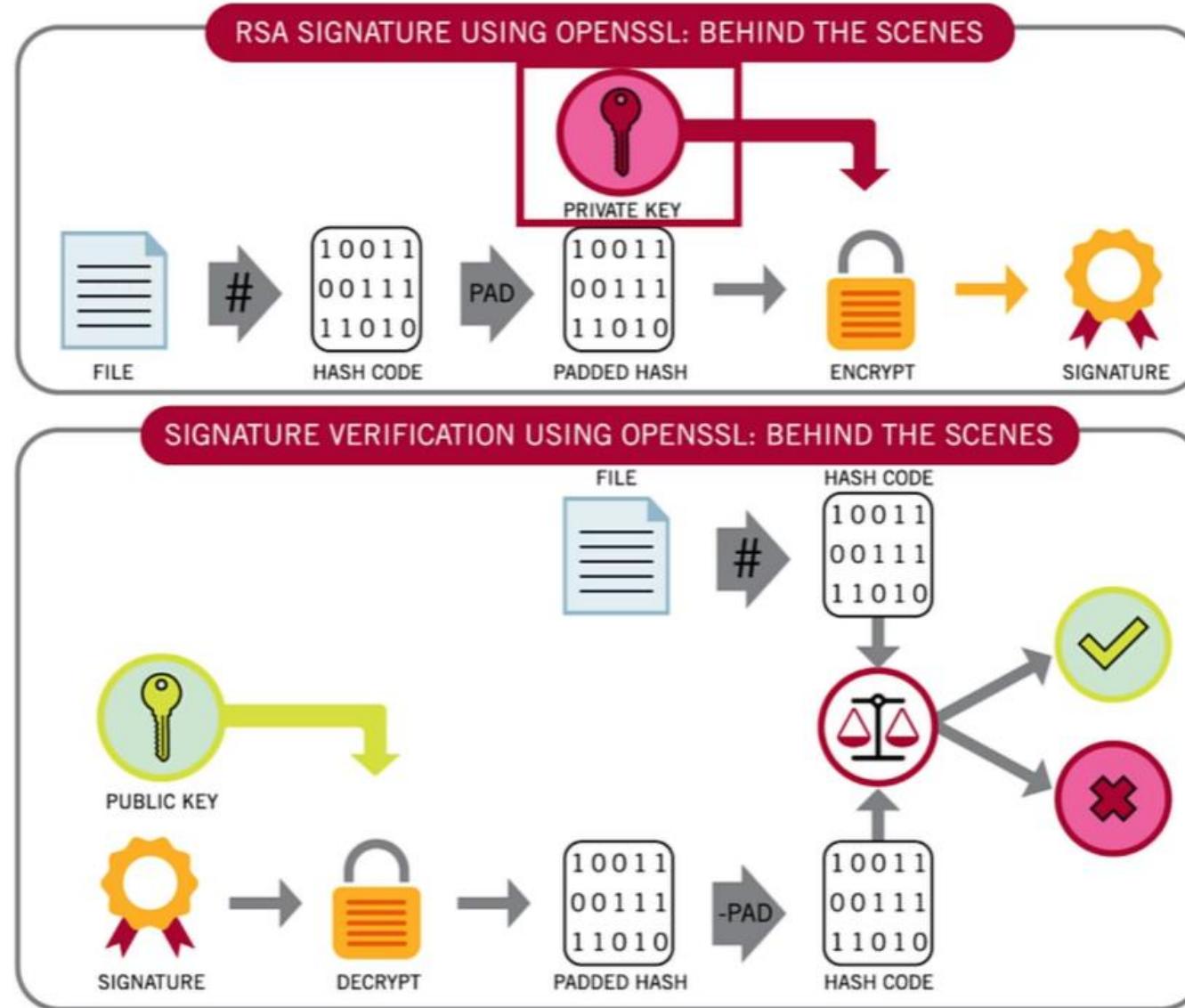
# RSA Signature Using OPENSSL (1/5)



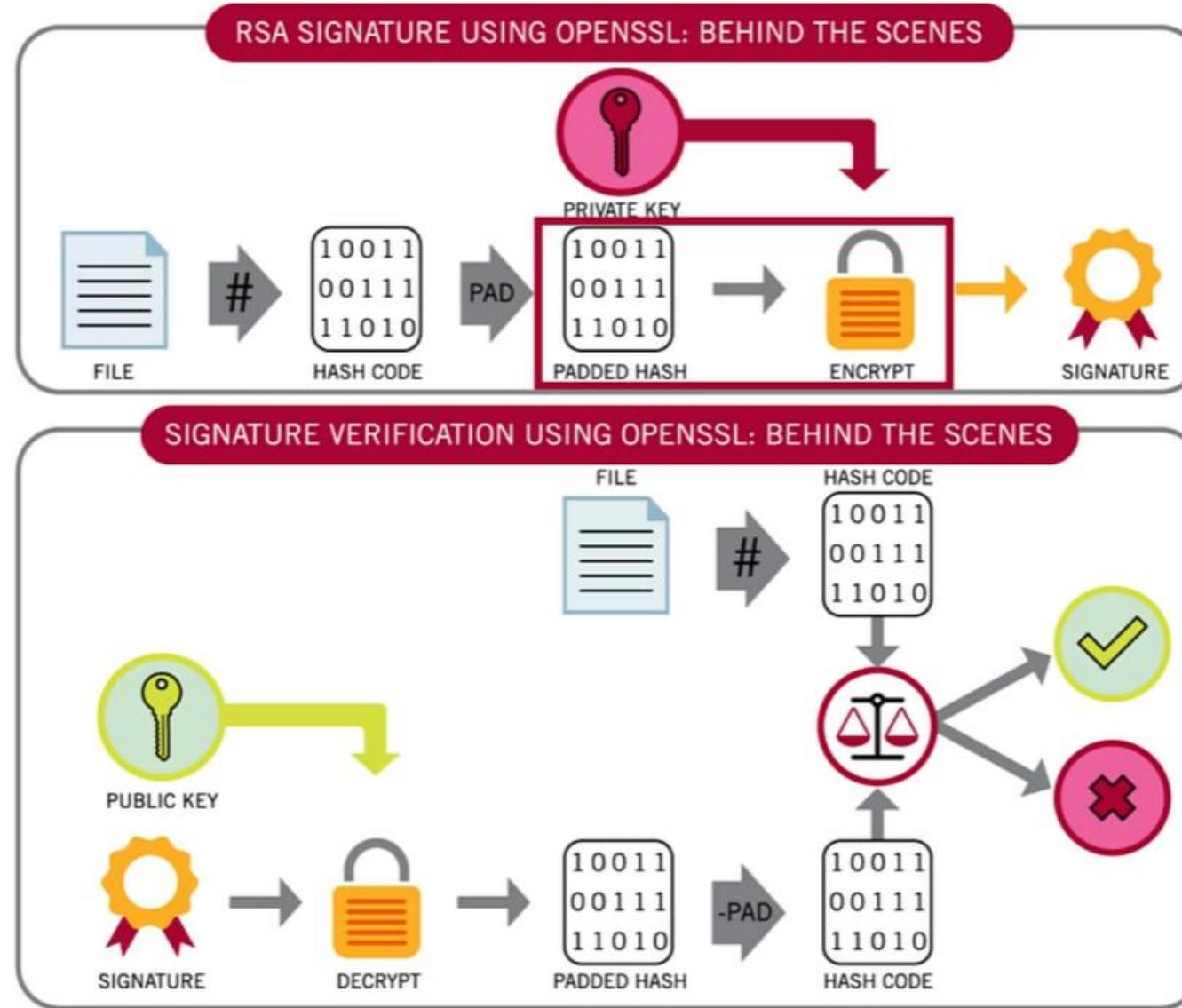
## RSA Signature Using OPENSSL (2/5)



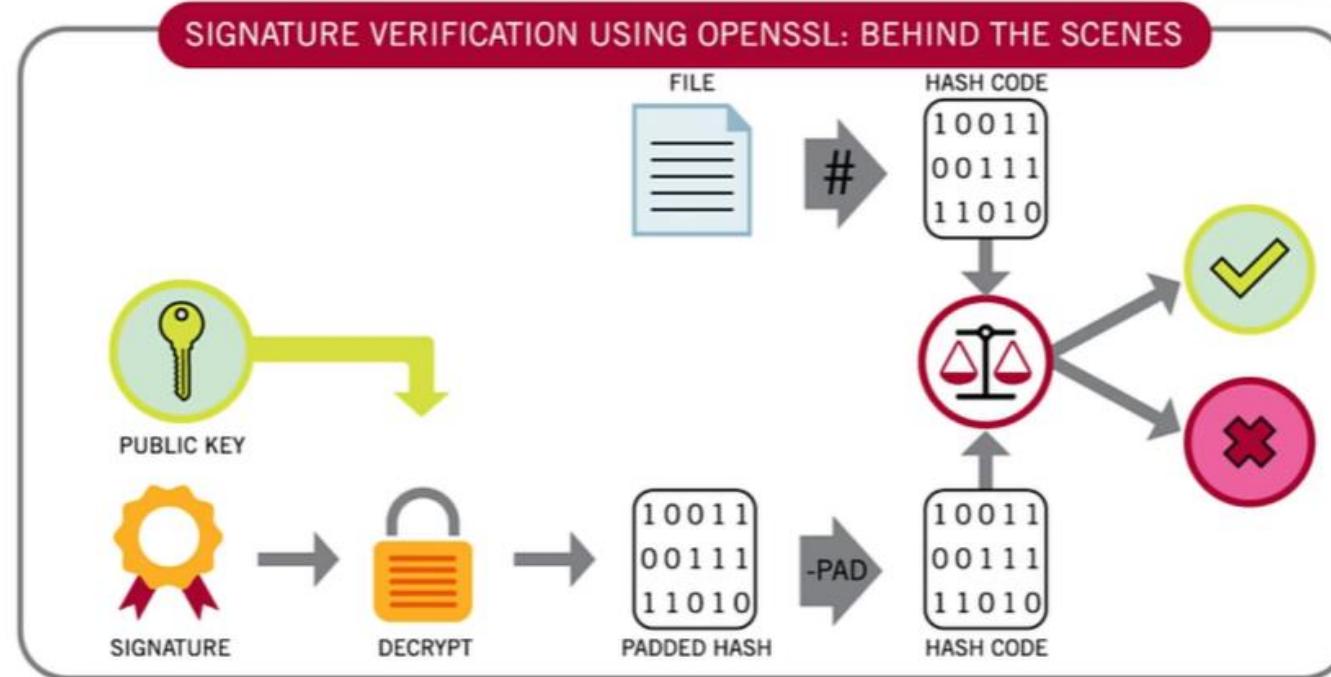
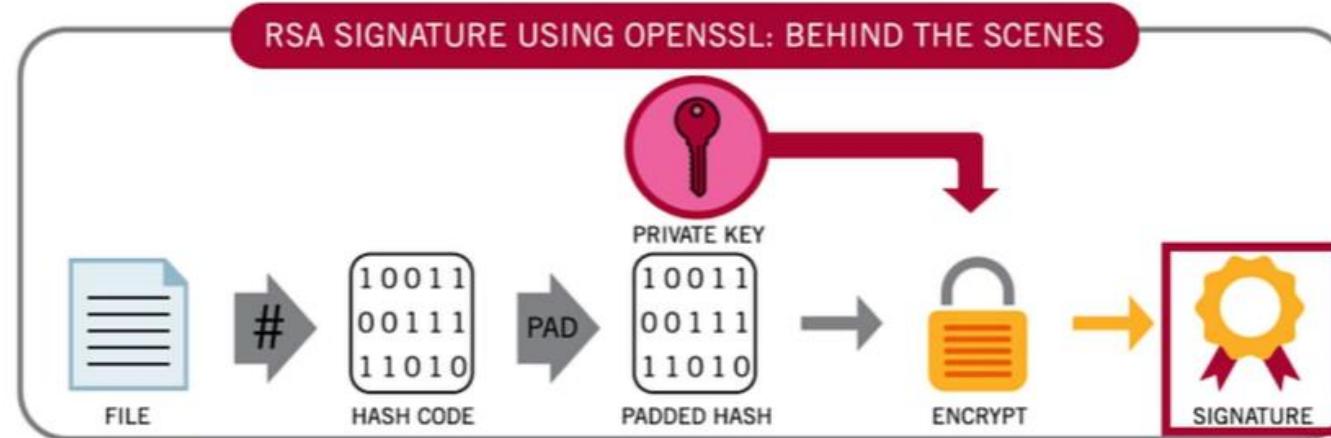
## RSA Signature Using OPENSSL (3/5)



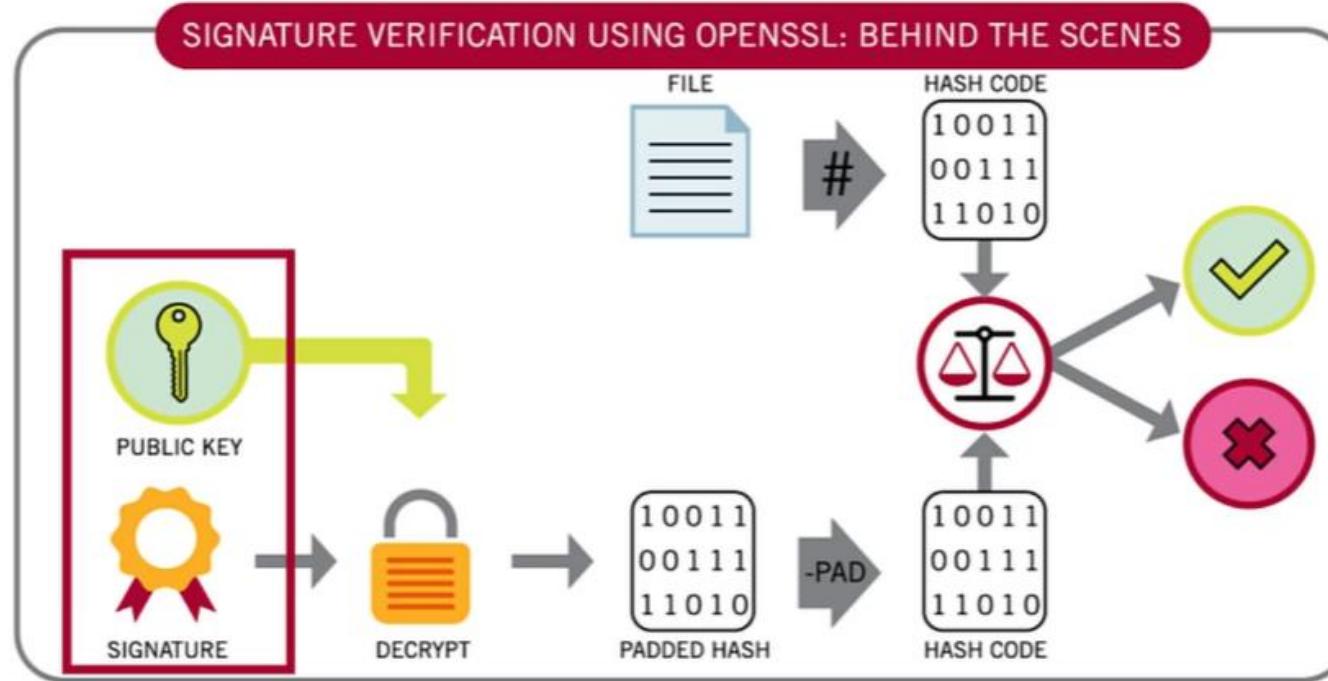
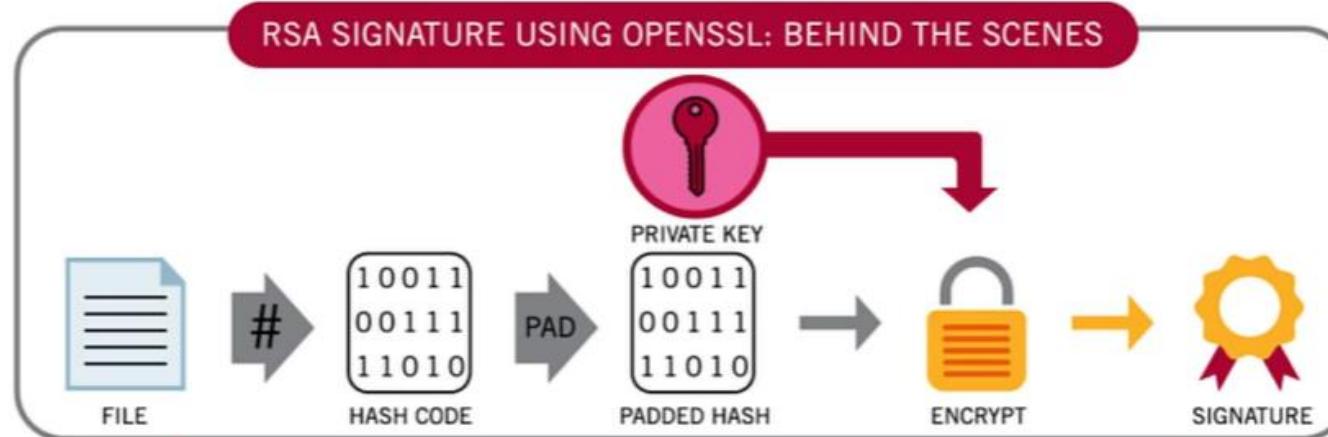
# RSA Signature Using OPENSSL (4/5)



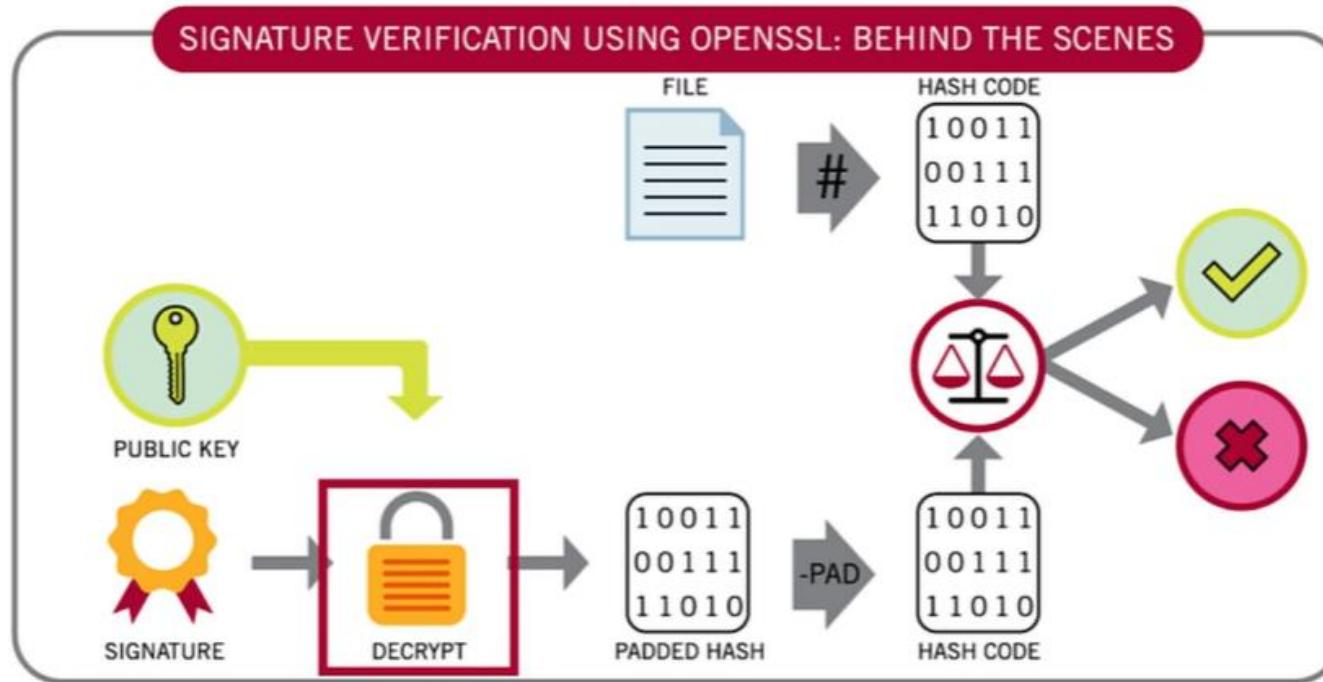
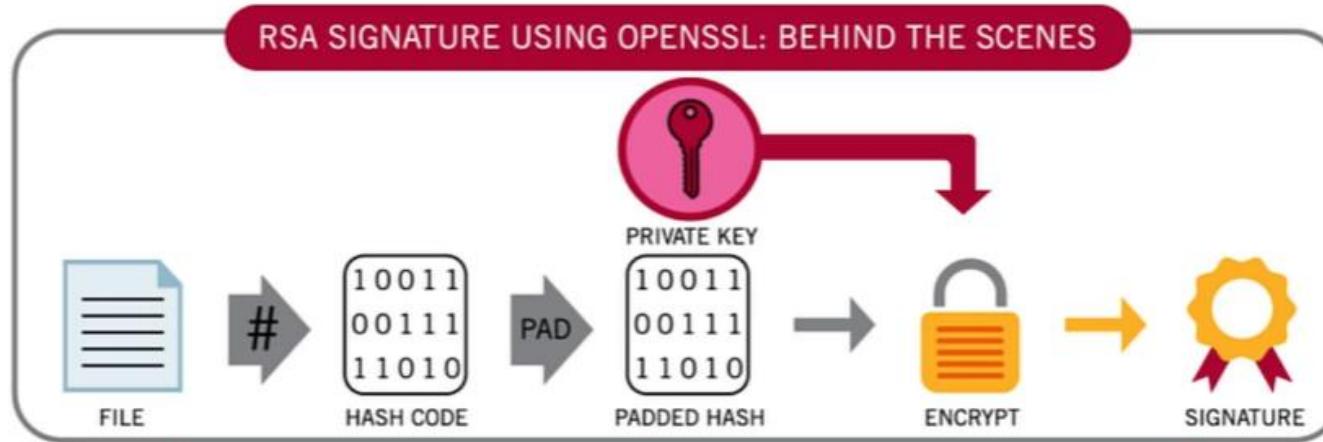
# RSA Signature Using OPENSSL (5/5)



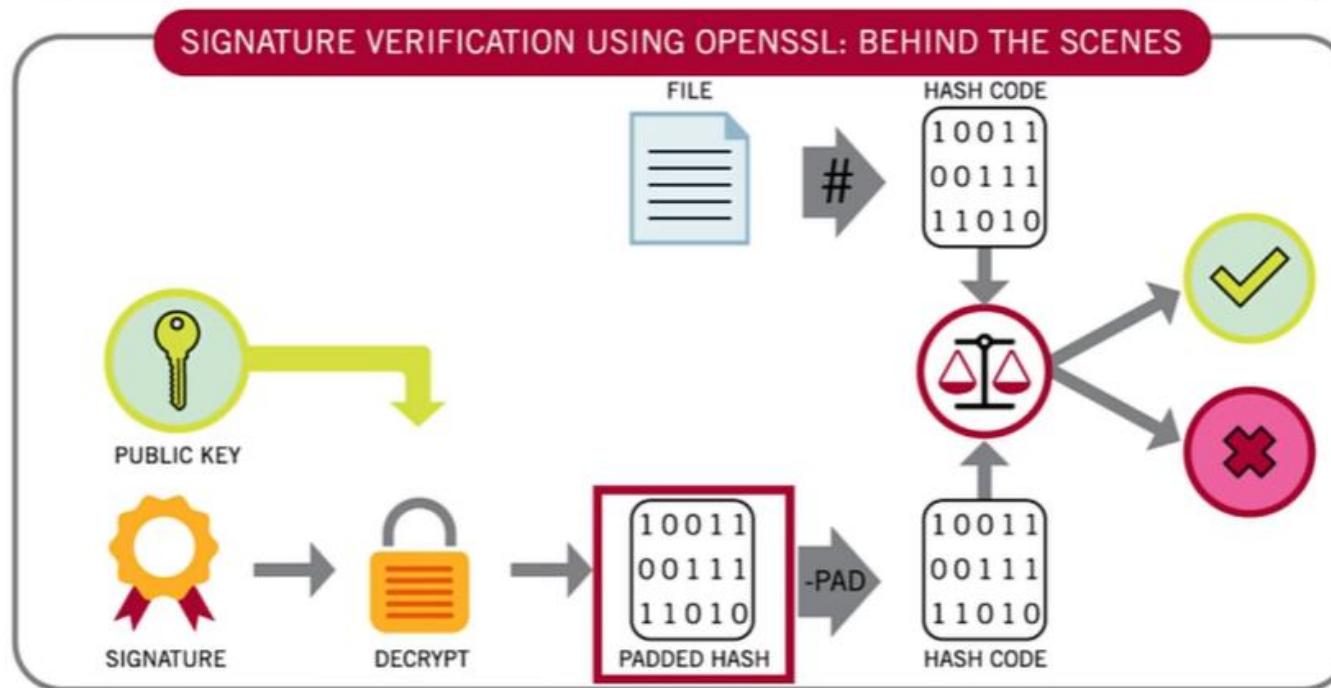
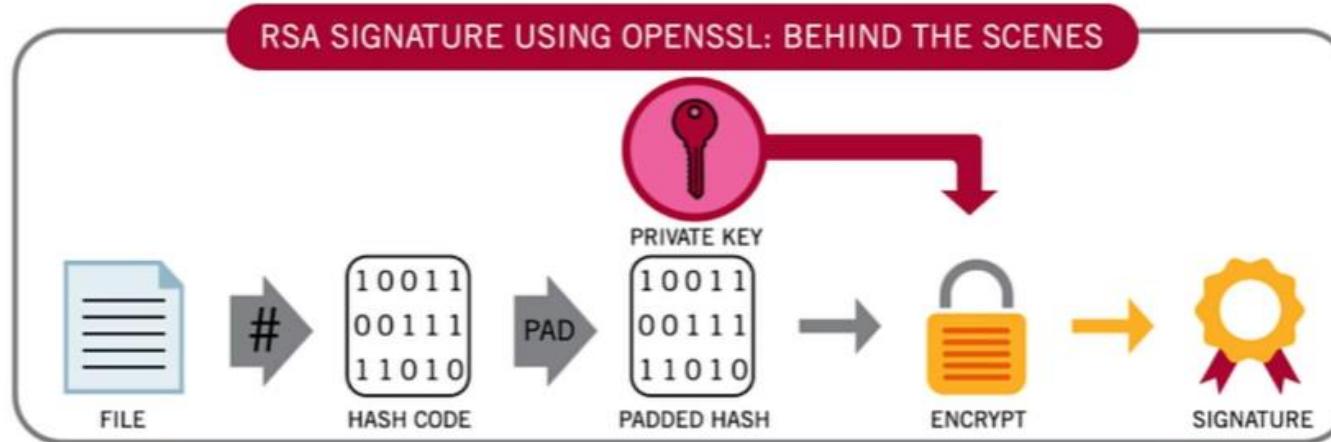
# Decrypting The File (1/13)



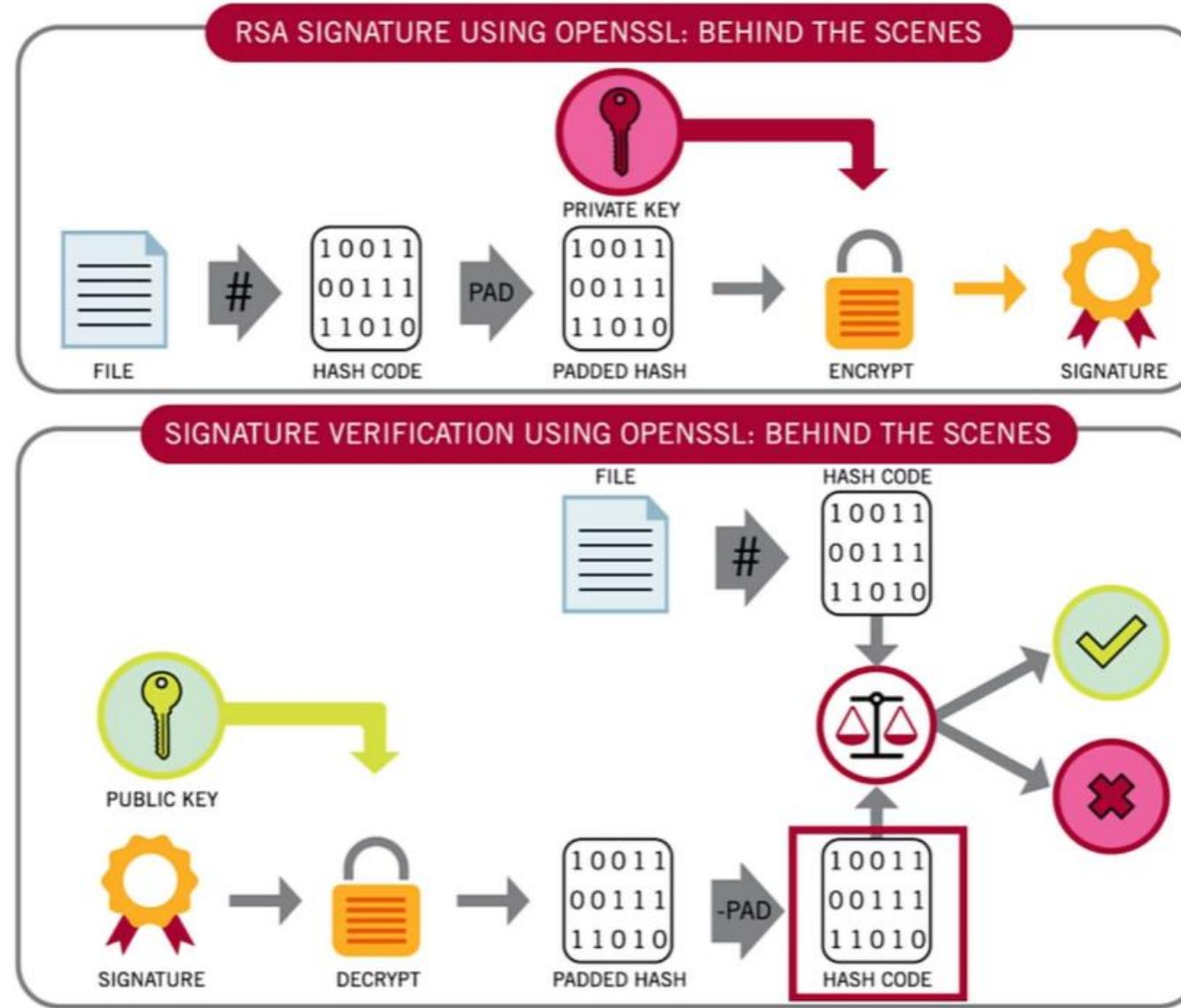
## Decrypting The File (2/13)



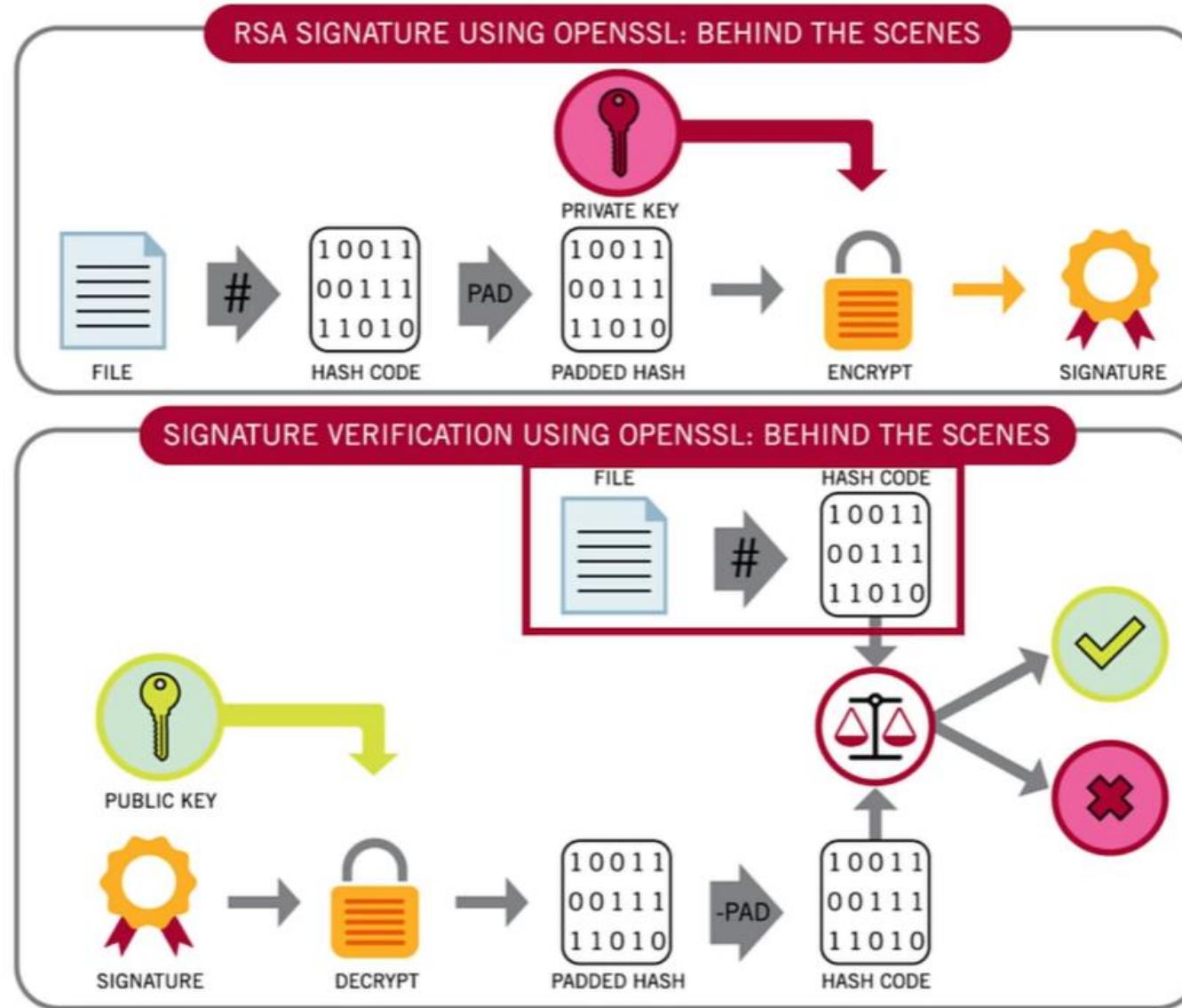
## Decrypting The File (3/13)



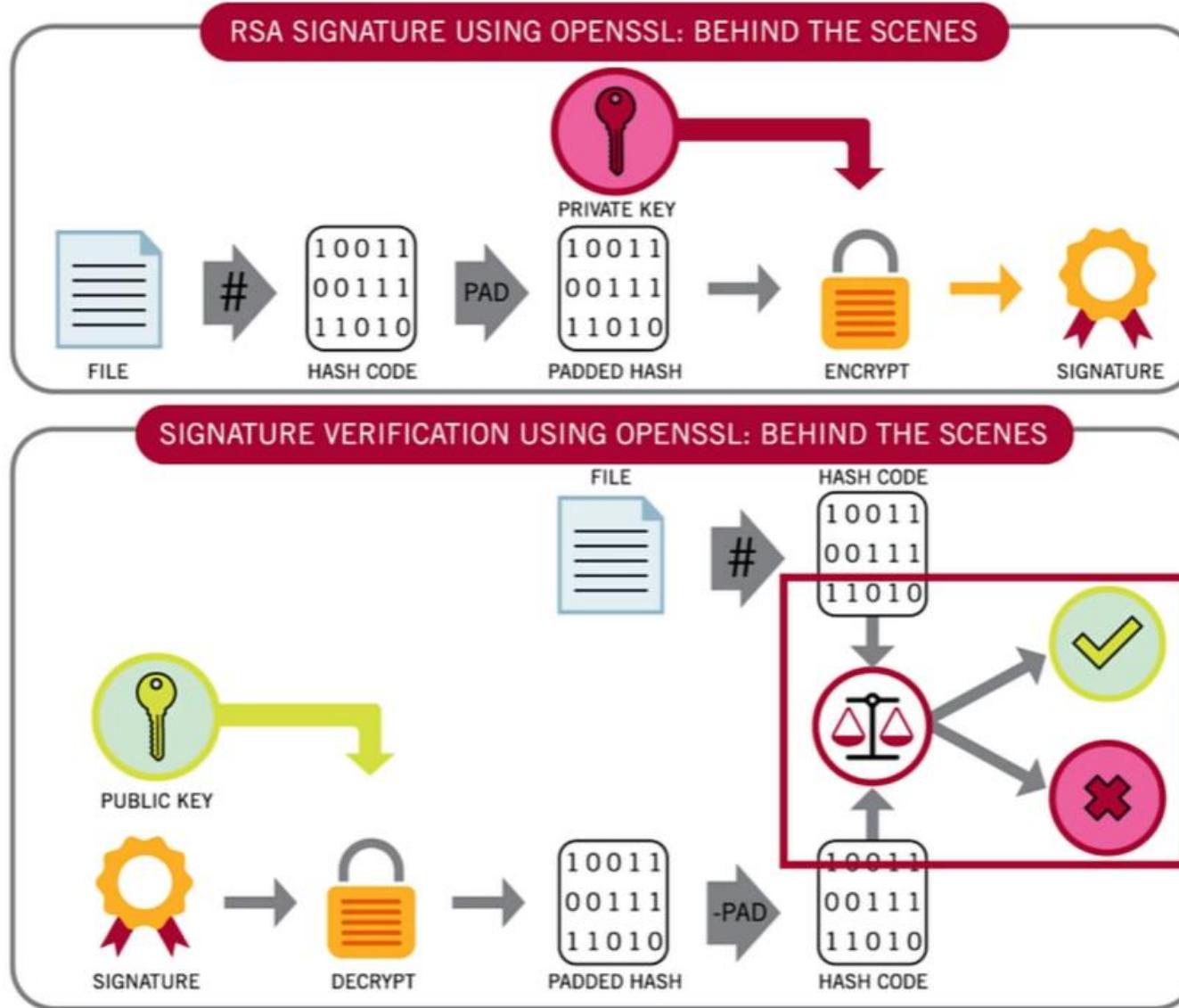
## Decrypting The File (4/13)



## Decrypting The File (5/13)



## Decrypting The File (6/13)



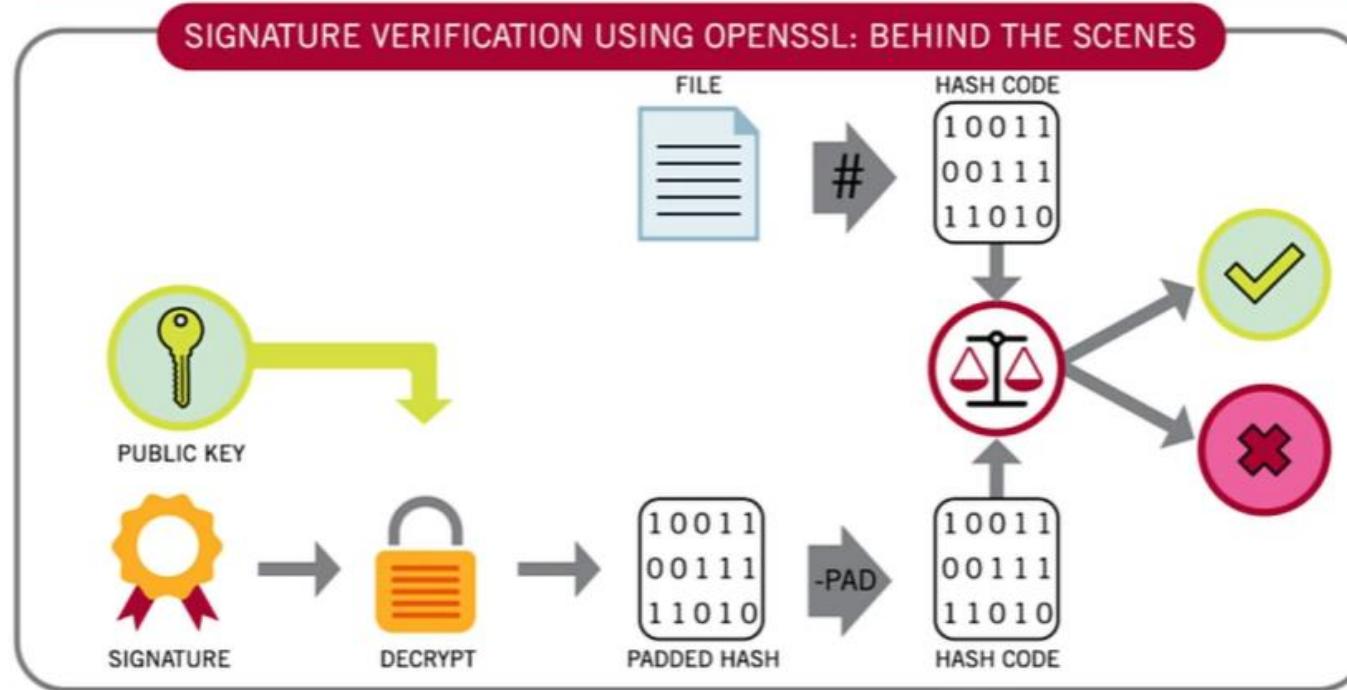
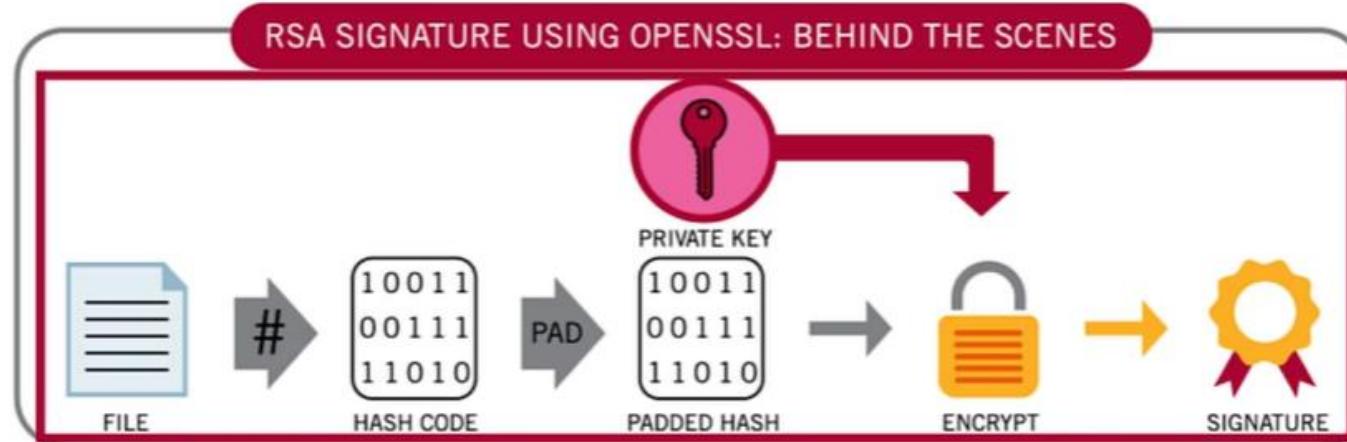
## Decrypting The File (7/13)

```
~ MIT/Courses/OAuth2/Keys — johnwilliams@Johns-MBP — ..s/OAuth2/Keys — -zsh — 88x25
(base) Keys > openssl dgst -sha1 -sign myprivate.pem -out sha1.sign hello.txt
(base) Keys > cat sha1.sign
ov?U?:??Bc?????Gü'2??????x??T8+mTc?q??????r?_??]?X}??wA|??%
(base) Keys > █
```

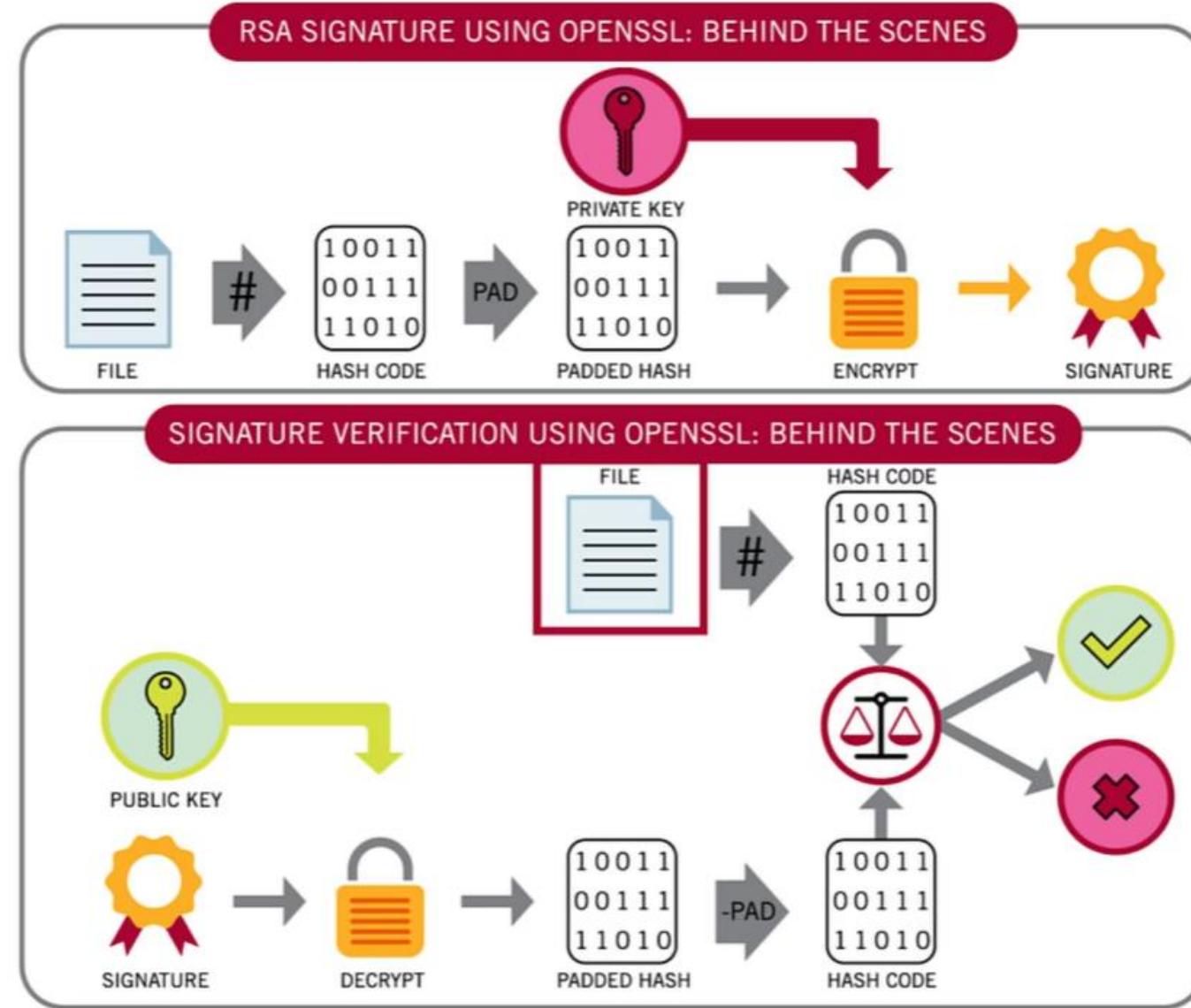
## Decrypting The File (8/13)

```
~/MIT/Courses/OAuth2/Keys — johnwilliams@Johns-MBP — ..s/OAuth2/Keys — -zsh — 88x25
(base) Keys > openssl dgst -sha1 -verify mypublic.pem -signature sha1.sign hello.txt
Verified OK
(base) Keys > █
```

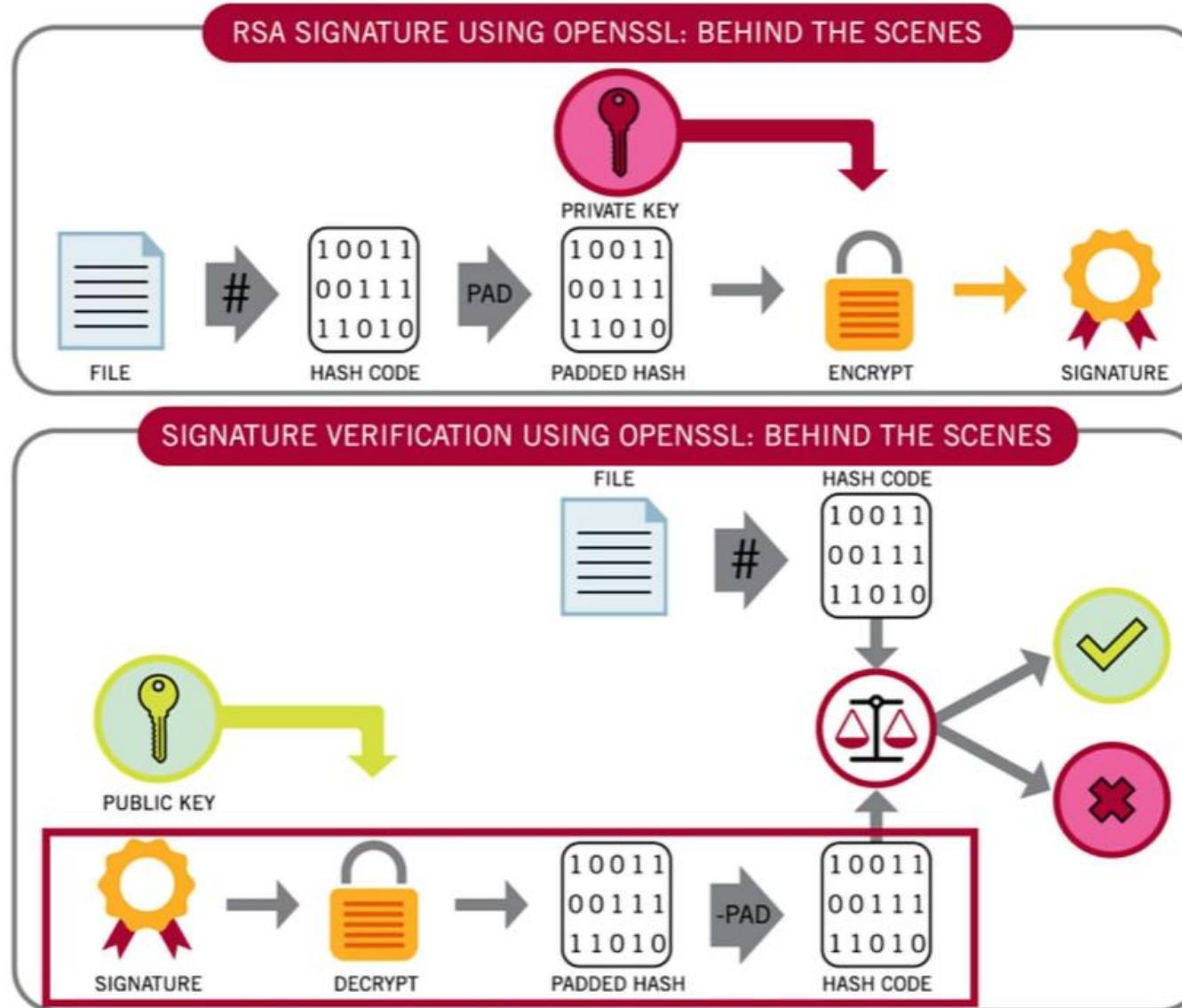
## Decrypting The File (9/13)



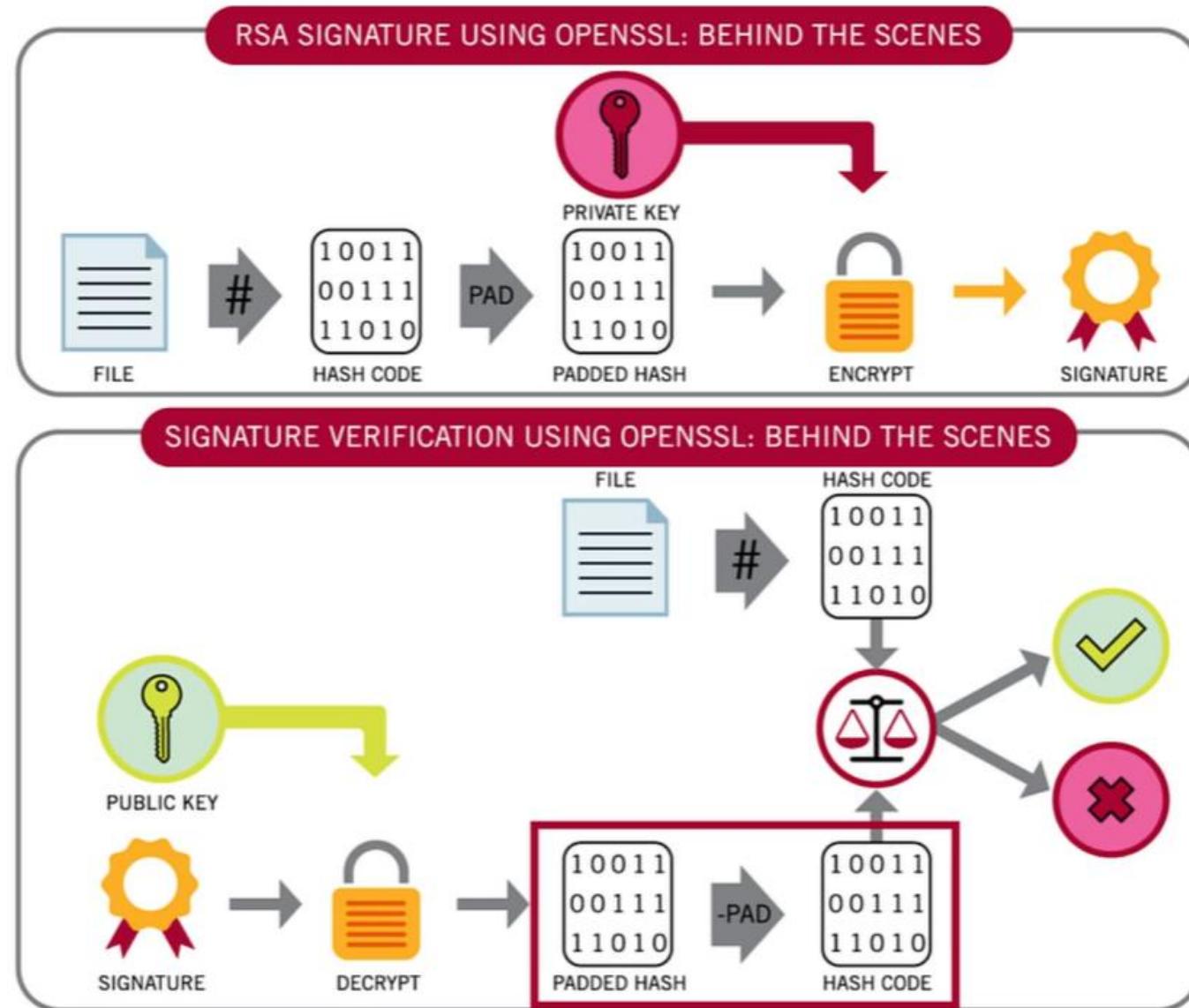
# Decrypting The File (10/13)



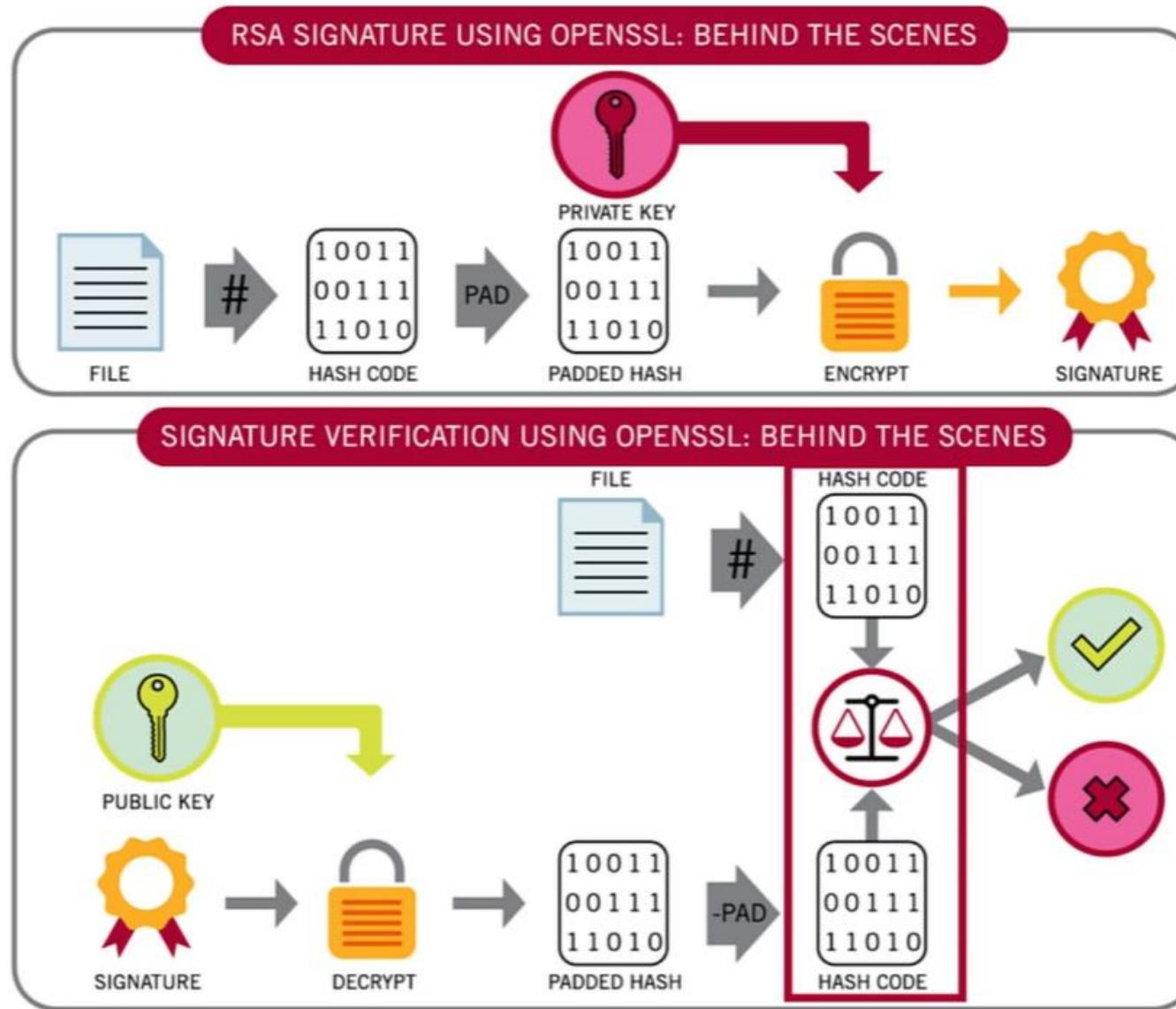
# Decrypting The File (11/13)



# Decrypting The File (12/13)



# Decrypting The File (13/13)



# Encrypting The Message (1/2)

```
~/MIT/Courses/OAuth2/Keys — johnwilliams@Johns-MBP — ..s/OAuth2/Keys — -zsh — 88x25
(base) Keys > ls
friend.pem      hello.txt      mypublic.pem
friendpublic.pem myprivate.pem  sha1.sign
(base) Keys > openssl rsautl -encrypt -pubin -inkey friendpublic.pem -ssl -in hello.txt
-out encrypthello.txt
(base) Keys > cat encrypthello.txt
0
S???1?:]D?/?C?2?[?<2??4??6،\H?X?&?s??????%
(base) Keys >
```

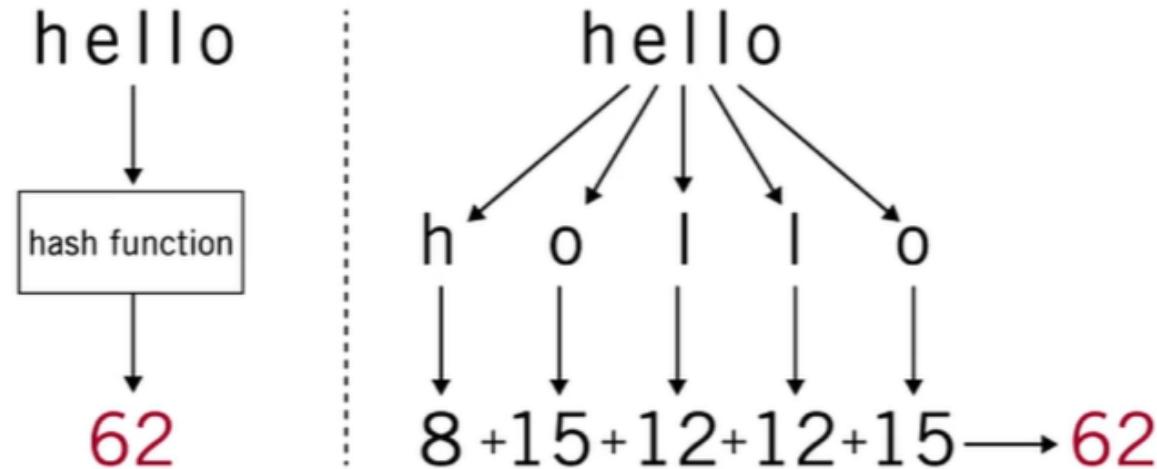
## Decrypting The Message (2/2)

```
~ /MIT/Courses/OAuth2/Keys — johnwilliams@Johns-MBP — ..s/OAuth2/Keys — -zsh — 88x25
(base) Keys > openssl rsautl -decrypt -inkey friend.pem -in encryptHello.txt -out myDecryptedMessage.txt
(base) Keys > cat myDecryptedMessage.txt
Hello World
(base) Keys > █
```

# Signing The Encrypted Message Instead Of Plain Text

```
~/MIT/Courses/OAuth2/Keys — johnwilliams@Johns-MBP — ..s/OAuth2/Keys — -zsh — 88x25
(base) Keys > openssl rsautl -decrypt -inkey friend.pem -in encrypthello.txt -out myDecryptedMessage.txt
(base) Keys > cat myDecryptedMessage.txt
Hello World
(base) Keys > cat encrypthello.txt
0
S???1?:]D?/?C?2?[?<2??4??6، \H?X?&?s??????%
(base) Keys > █
```

# What Is A Hash?



- Hash is like a "finger print" of a document
- If anything in the document changes, the hash changes
- Can't reconstruct the document from the hash
- One way process: Document → Hash

## Hash Function Examples

Let  $h(k) = k \% 15$ . Then,

if $k =$	25	129	35	2501	47	36
$h(k) =$	10	9	5	11	2	6

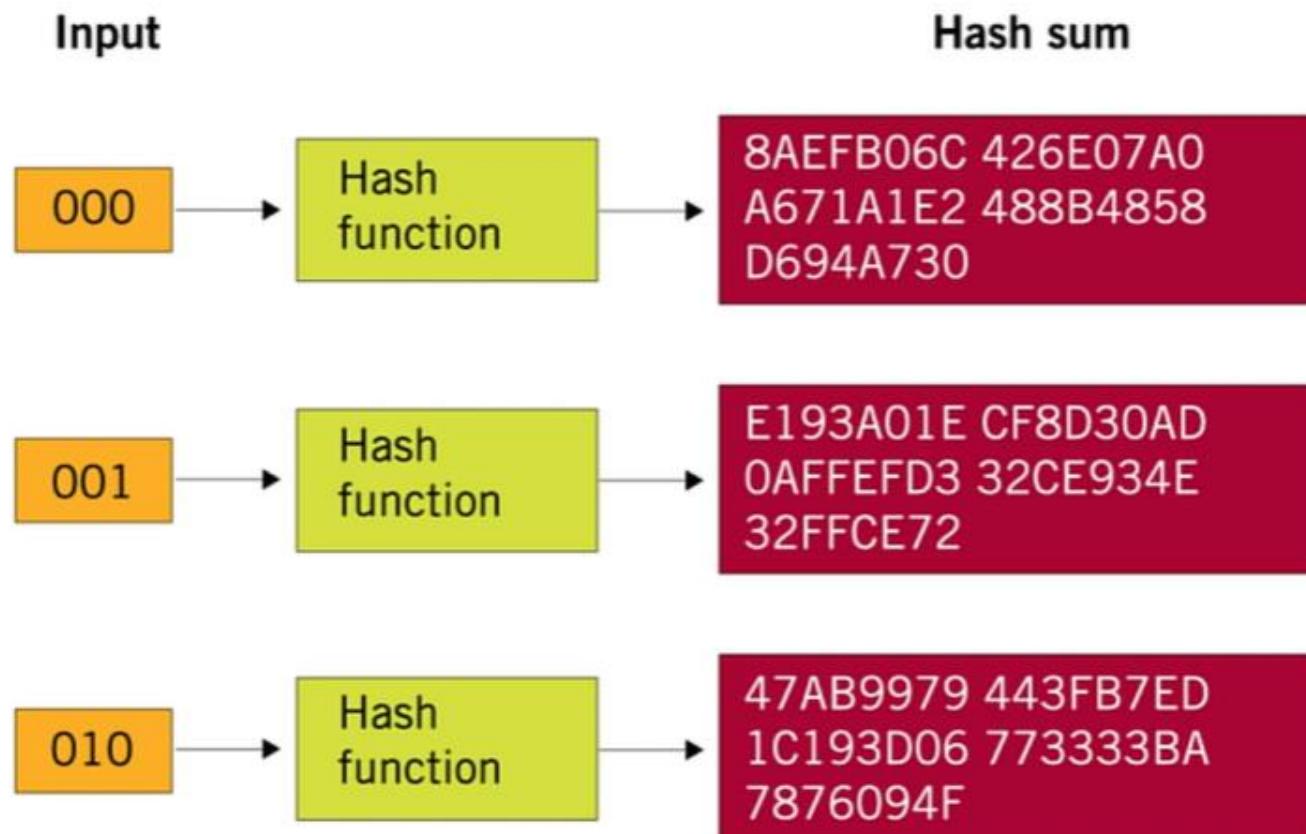
Storing the keys in the array is straightforward:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
-	-	47	-	-	35	36	-	-	129	25	2501	-	-	-

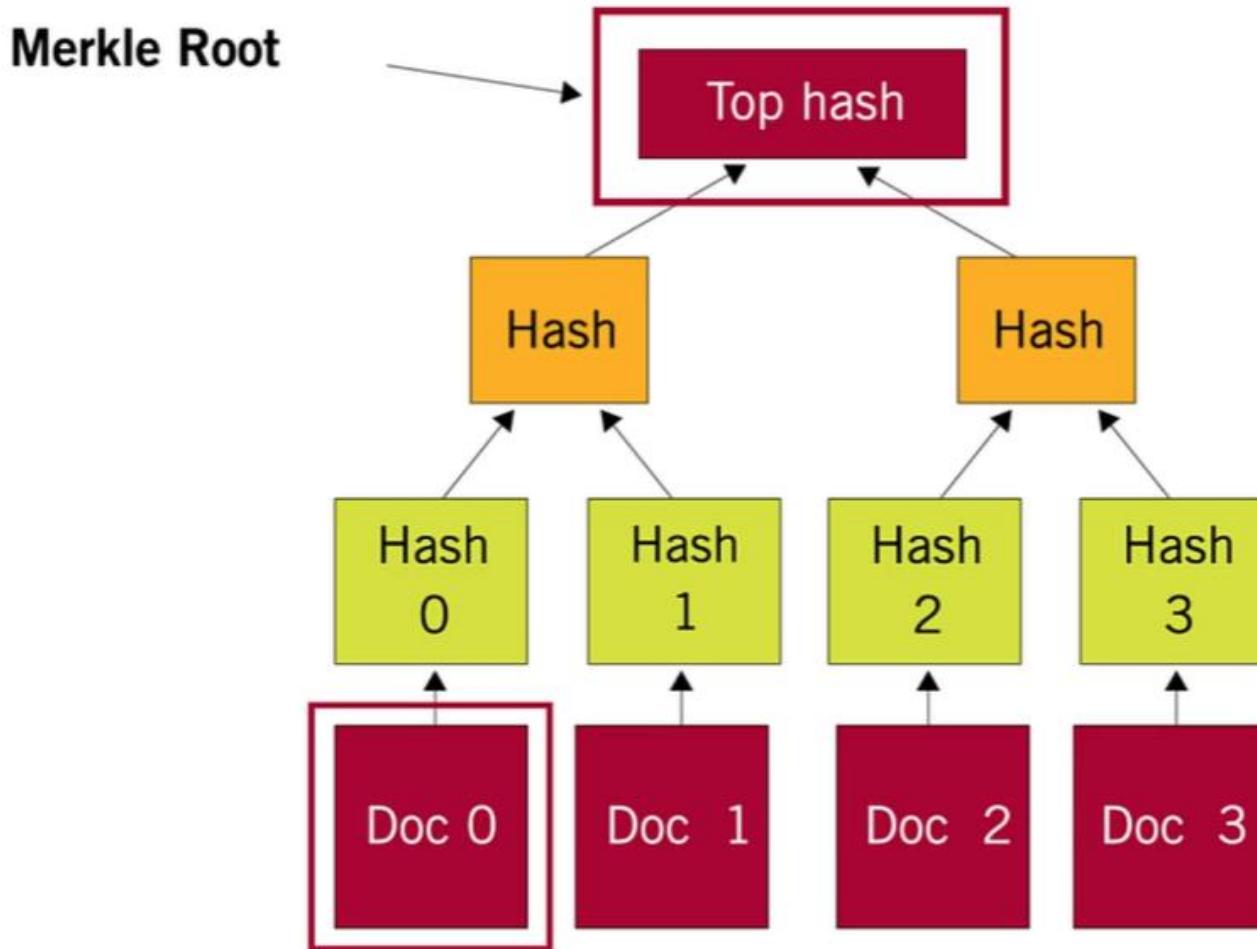
Thus, *delete* and *find* can be done in  $O(1)$ , and  
also *insert*

# Avalanche Property

Avalanche property - Small one bit change in input leads to radically different hash sum



# Merkle Tree



# Hash Function Demo

The screenshot shows a web browser window with the URL `file:///C:/chain-master/HashDemo/hashDemo.html` in the address bar. The page content is titled "Hash Demo". It contains a form with the following fields:

- Input Text:
- Input Type:
- SHA Variant:
- Number of Rounds:
- Output Type:
- Output Hash:

Copyright © 2008-2017 [Brian Turek](#)

# Hash Function: Blockie (1/2)

Hash Demo

Input Text:

Input Type: TEXT

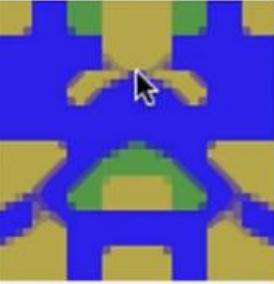
SHA Variant: SHA-256

Number of Rounds: 1

Output Type: HEX

Output Hash: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855

Copyright © 2008-2017 [Brian Turek](#)



# Hash Function: Blockie (2/2)

Hash Demo

Input Text:

Input Type:

SHA Variant:

Number of Rounds:

Output Type:

Output Hash:



Copyright © 2008-2017 Brian Turek

# Creating Blockie (1/2)

jsSHA - SHA Hashes in JavaScript

file:///XPRO/Course1/hashDemoBlockies/hashDemoBlockies.html

Error

Hash Demo

Input Text: John Williams

Input Type: TEXT

SHA Variant: SHA-256

Number of Rounds: 1

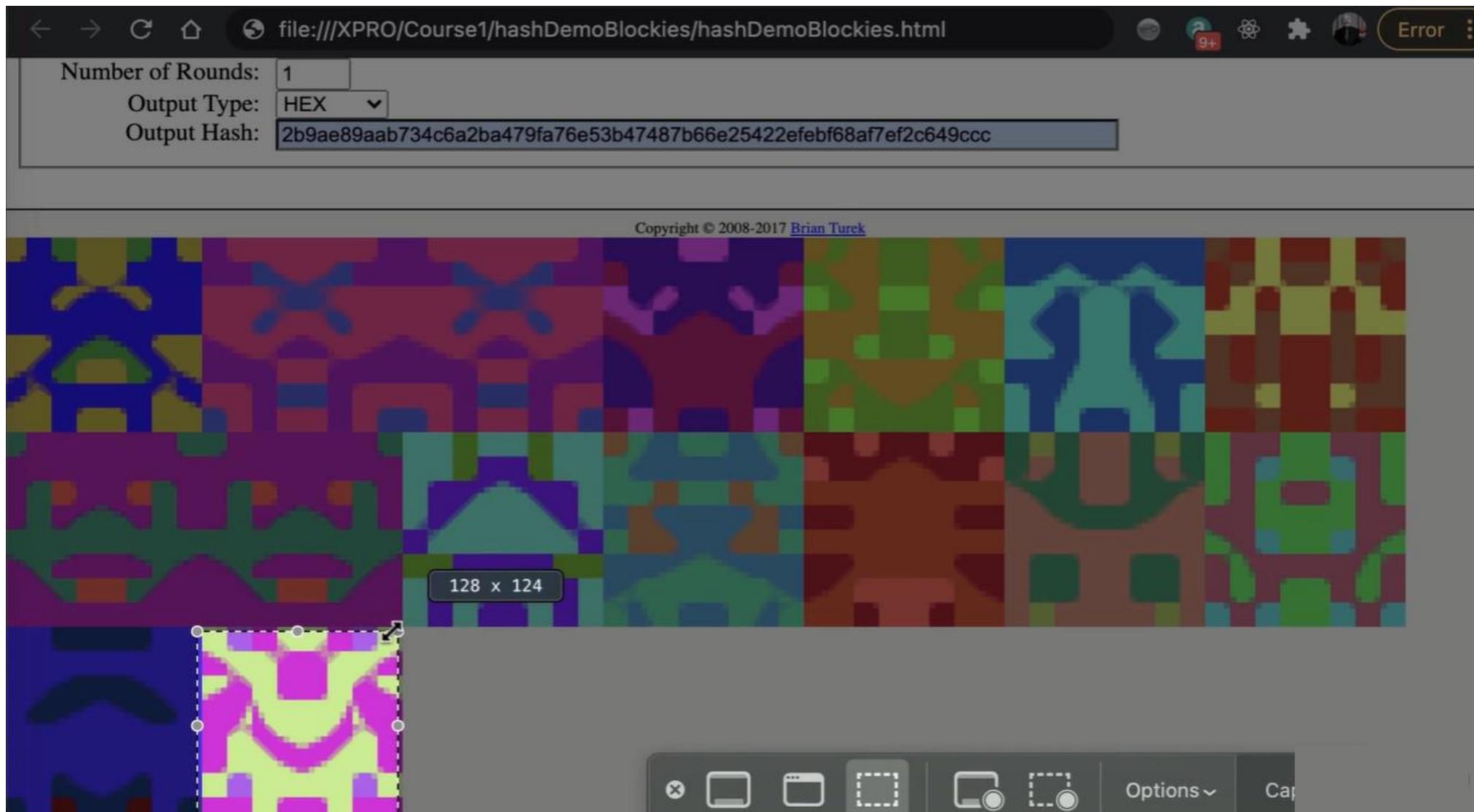
Output Type: HEX

Output Hash: 2b9ae89aab734c6a2ba479fa76e53b47487b66e25422efebf68af7ef2c649ccc

Copyright © 2008-2017 Brian Turek



## Creating Blockie (2/2)



# Permute Letters And Print Them All Out (1/12)

P ( 

a	b	c
---	---	---

 ) can be broken down to small tasks

Pick  
loop

a

P ( 

b	c
---	---

 )

b

P ( 

c
---

 )

```
Permute(string){  
for(i = 0; i < len;i++){  
pick = string[i];  
?????
```

b

P ( 

a	c
---	---

 )

c

P ( 

b
---

 )

c

P ( 

a	b
---	---

 )

## Permute Letters And Print Them All Out (2/12)

P ( a b c ) can be broken down to small tasks

Pick  
loop

a

b

c

P (

P

P (

b

c

perute(string){

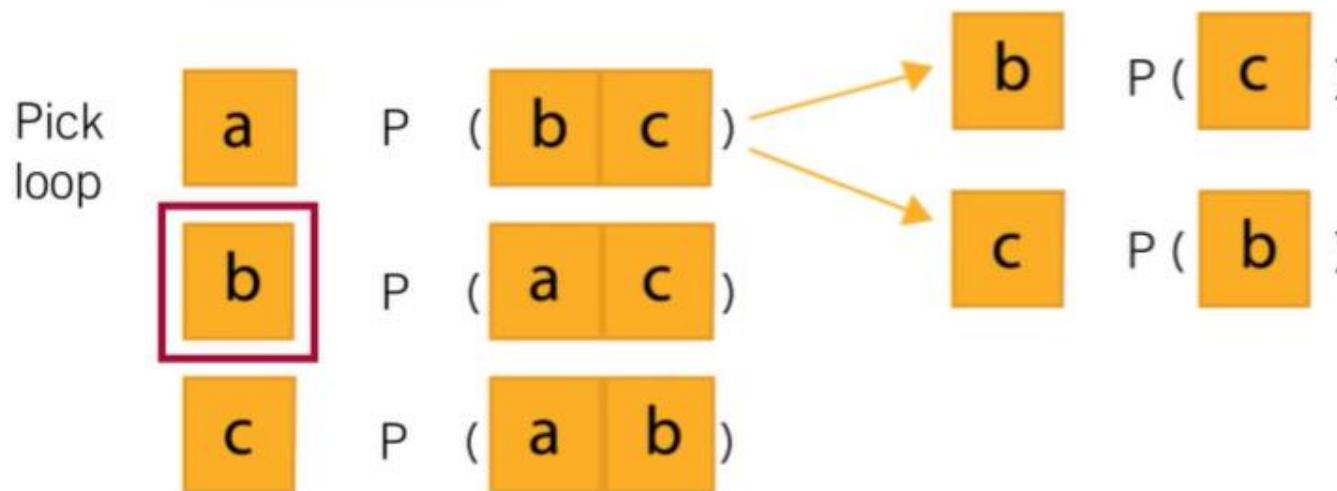
    for(i < len;i++){

        //

Now go and try this

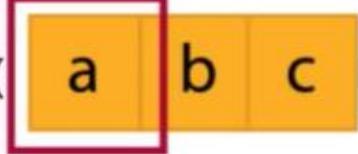
## Permute Letters And Print Them All Out (3/12)

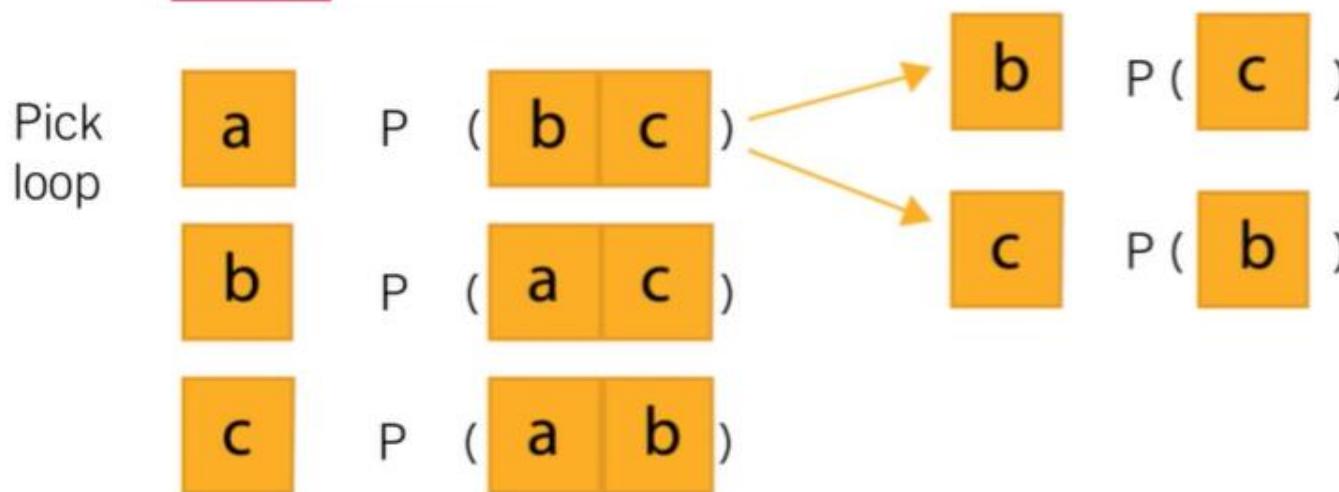
P ( a | b | c ) can be broken down to small tasks



```
Permute(string){  
    for(i = 0; i < len;i++){  
        pick = string[i];  
        remainder =  
            string.slice(0,i)+string(i+1,len)  
        For (var perm of permute(remainder)){  
            permutations. push(pick + perm);  
        }  
    }  
    return permutations;  
}
```

## Permute Letters And Print Them All Out (4/12)

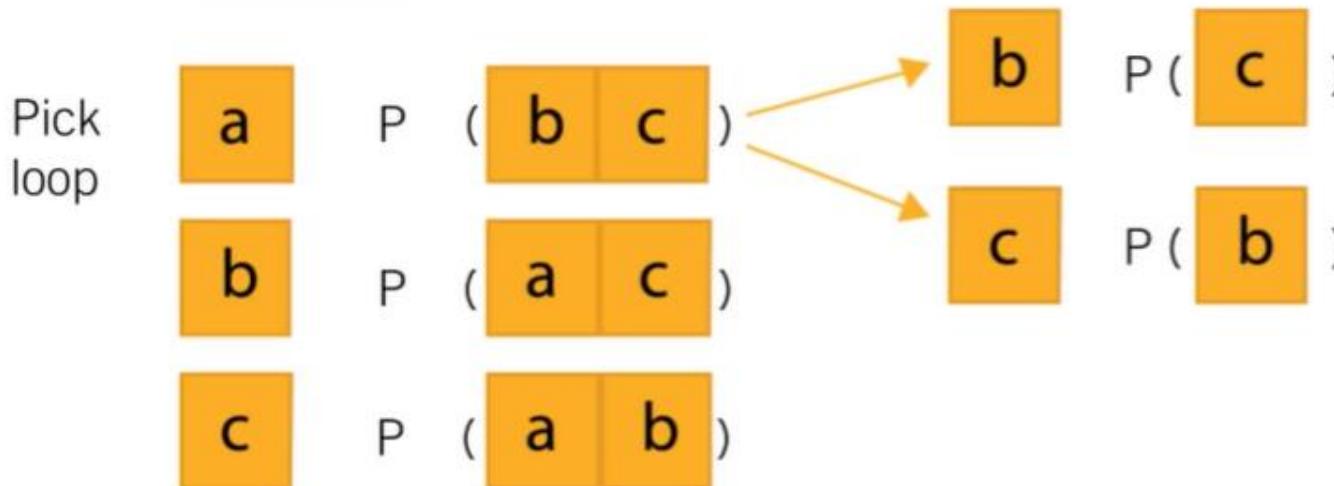
P () can be broken down to small tasks



```
Permute(string){  
    for(i = 0; i < len;i++){  
        pick = string[i];  
        remainder =  
            string.slice(0,i)+string(i+1,len)  
        For (var perm of permute(remainder)){  
            permutations. push(pick + perm);  
        }  
    }  
    return permutations;  
}
```

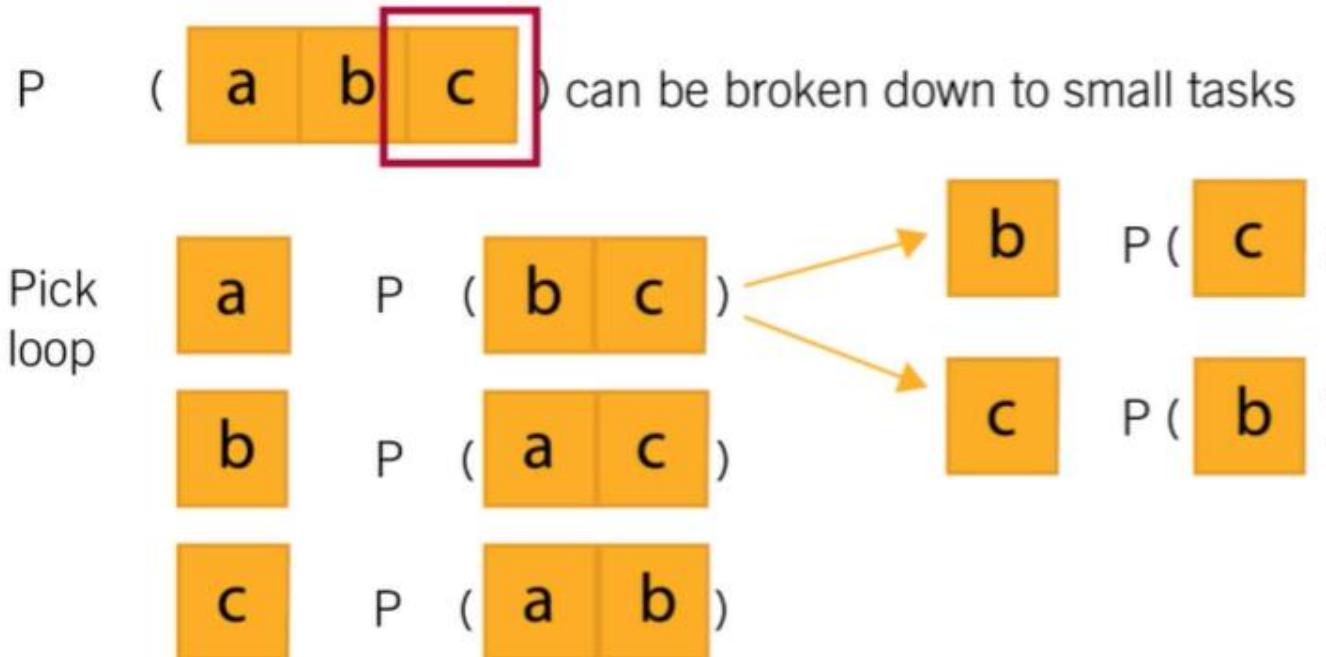
## Permute Letters And Print Them All Out (5/12)

P ( [ a | b | c ] ) can be broken down to small tasks



```
Permute(string){  
    for(i = 0; i < len;i++){  
        pick = string[i];  
        remainder =  
            string.slice(0,i)+string(i+1,len)  
        For (var perm of permute(remainder)){  
            permutations. push(pick + perm);  
        }  
    }  
    return permutations;  
}
```

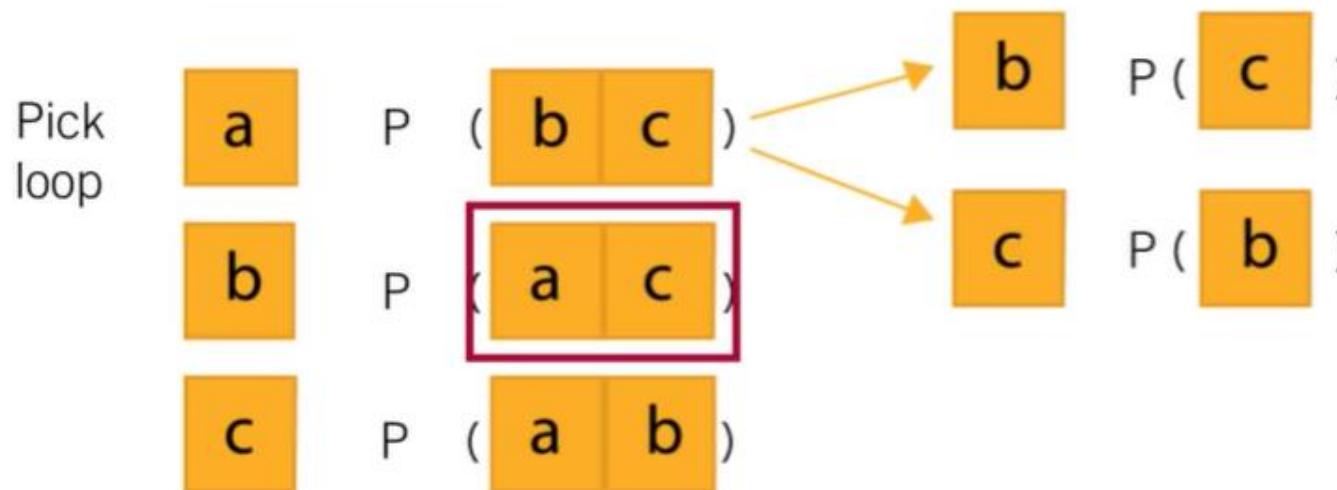
## Permute Letters And Print Them All Out (6/12)



```
Permute(string){  
    for(i = 0; i < len;i++){  
        pick = string[i];  
        remainder =  
            string.slice(0,i)+string(i+1,len)  
        For (var perm of permute(remainder)){  
            permutations. push(pick + perm);  
        }  
    }  
    return permutations;  
}
```

# Permute Letters And Print Them All Out (7/12)

P ( [ a | b | c ] ) can be broken down to small tasks



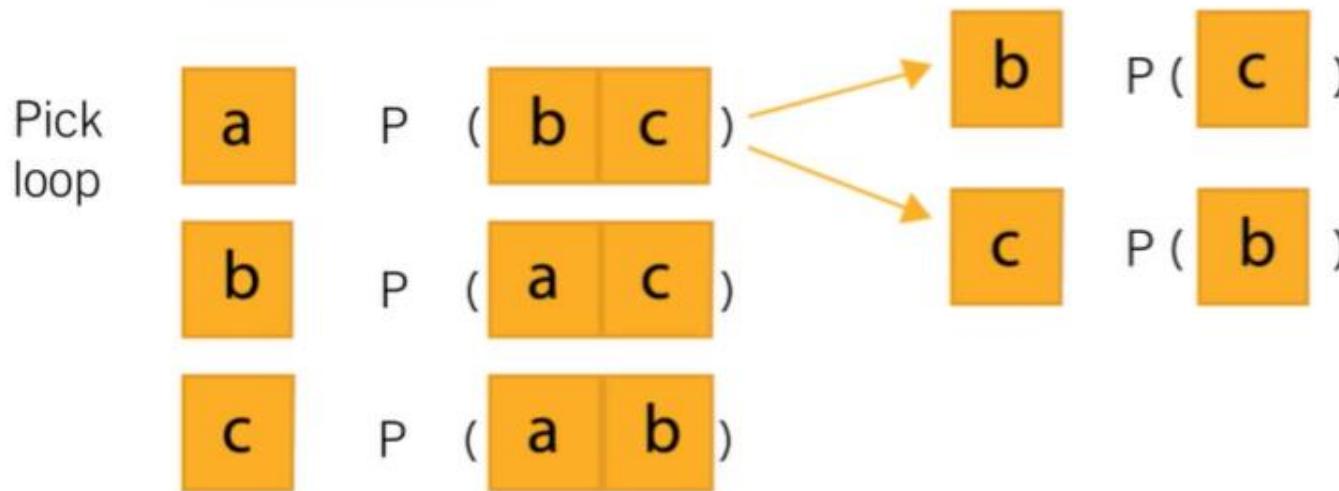
```
Permute(string){  
    for(i = 0; i < len;i++){  
        pick = string[i];  
        remainder =  
            string.slice(0,i)+string(i+1,len)  
        For (var perm of permute(remainder)){  
            permutations. push(pick + perm);  
        }  
    }  
    return permutations;  
}
```

# Permute Letters And Print Them All Out (8/12)

P ( 

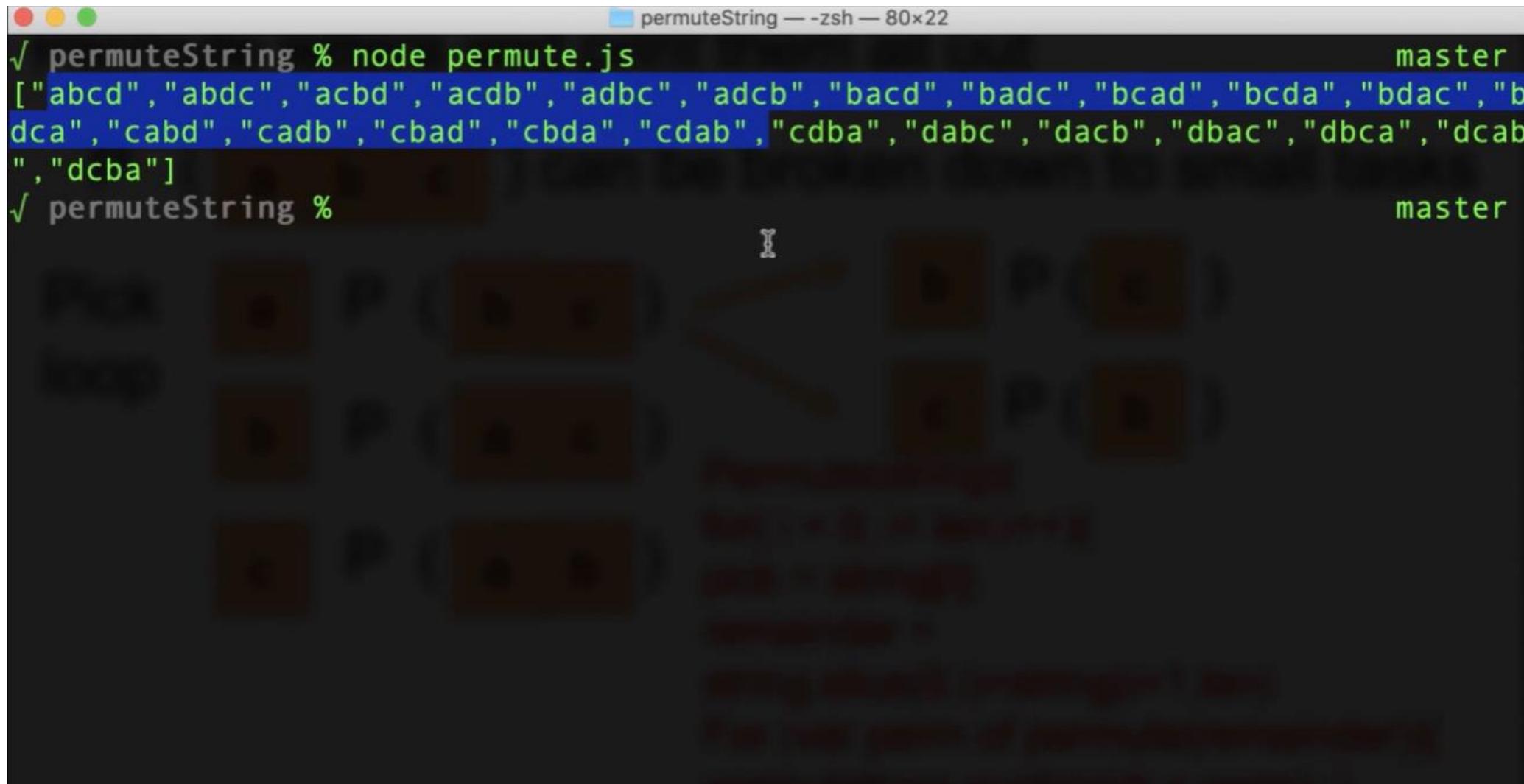
a	b	c
---	---	---

 ) can be broken down to small tasks



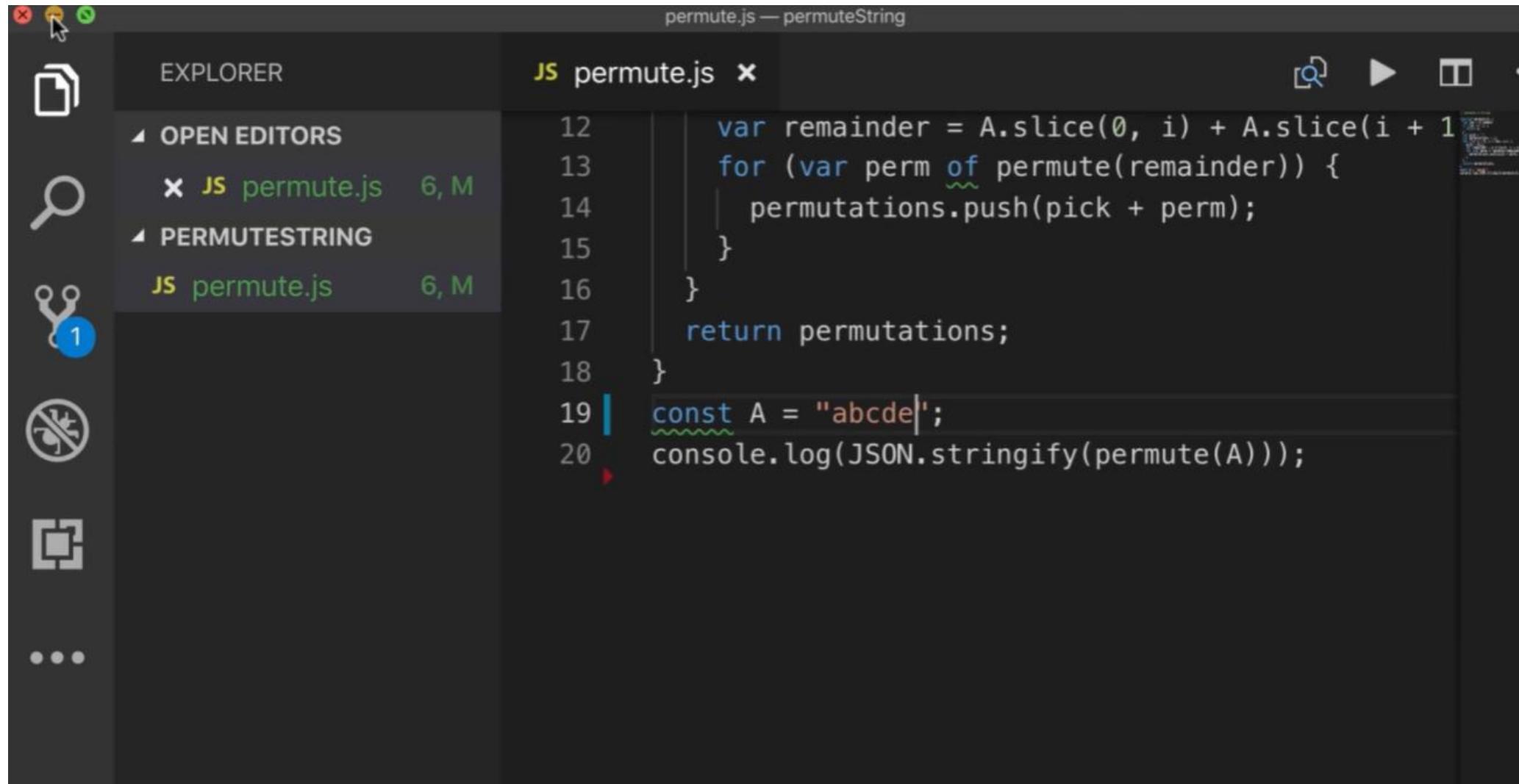
```
Permute(string){  
    for(i = 0; i < len;i++){  
        pick = string[i];  
        remainder =  
            string.slice(0,i)+string(i+1,len)  
        For (var perm of permute(remainder)){  
            permutations. push(pick + perm);  
        }  
    }  
    return permutations;  
}
```

## Permute Letters And Print Them All Out (9/12)



```
permuteString % node permute.js
["abcd", "abdc", "acbd", "acdb", "adbc", "adcb", "bacd", "badc", "bcad", "bcda", "bdac", "b
dca", "cabd", "cadb", "cbad", "cbda", "cdab", "cdba", "dabc", "dacb", "dbac", "dbca", "dcab
", "dcba"]
permuteString %
```

# Permute Letters And Print Them All Out (10/12)



A screenshot of the Visual Studio Code (VS Code) interface. The title bar says "permute.js — permuteString". The left sidebar shows the "EXPLORER" view with two files: "JS permute.js" and "JS permuteString". The main editor area contains the following JavaScript code:

```
12 var remainder = A.slice(0, i) + A.slice(i + 1);
13 for (var perm of permute(remainder)) {
14     permutations.push(pick + perm);
15 }
16 }
17 return permutations;
18 }
19 const A = "abcde";
20 console.log(JSON.stringify(permute(A)));
```

# Permute Letters And Print Them All Out (11/12)

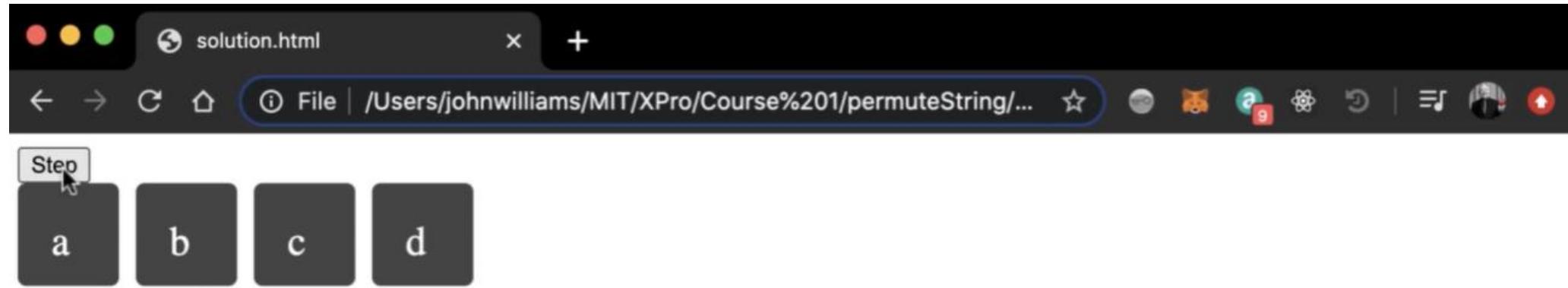
```
permuteString % node permute.js                                master
[ "abcd", "abdc", "acbd", "acdb", "adbc", "adcb", "bacd", "badc", "bcad", "bcda", "bdac", "b
dca", "cabd", "cadb", "cbad", "cbda", "cdab", "cdba", "dabc", "dacb", "dbac", "dbca", "dcab
", "dcba"]
permuteString % node permute.js                                master
[ "abcde", "abced", "abdce", "abdec", "abecd", "abedc", "acbde", "acbed", "acdbe", "acdeb"
, "acebd", "acedb", "adbce", "adbec", "adcbe", "adceb", "adebc", "adecb", "aebcd", "aebdc"
, "aecbd", "aecdb", "aedbc", "aedcb", "bacde", "baced", "badce", "badec", "baecd", "baedc"
, "bcade", "bcaed", "bcdae", "bcdea", "bcead", "bceda", "bdace", "bdaec", "bdcae", "bdcea"
, "bdeac", "bdeca", "beacd", "beadc", "becad", "becda", "bedac", "bedca", "cabde", "cabed"
, "cadbe", "cadeb", "caebd", "caedb", "cbade", "cbaed", "cbdae", "cbdea", "cbead", "cbeda"
, "cdabe", "cdaeb", "cdbae", "cdbea", "cdeab", "cdeba", "ceabd", "ceadb", "cebad", "cebda"
, "cedab", "cedba", "dabce", "dabec", "dacbe", "daceb", "daebc", "daecb", "dbace", "dbaec"
, "dbcae", "dbcea", "dbeac", "dbeca", "dcabe", "dcaeb", "dcbae", "dcbea", "dceab", "dceba"
, "deabc", "deacb", "debac", "debca", "decab", "decba", "eabcd", "eabdc", "eacbd", "eacdb"
, "eadbc", "eadcb", "ebacd", "ebadc", "ebcad", "ebcda", "ebdac", "ebdca", "ecabd", "ecadb"
, "ecbad", "ecbda", "ecdab", "ecdba", "edabc", "edacb", "edbac", "edbca", "edcab", "edcba"
]
permuteString %
```

# Permute Letters And Print Them All Out (12/12)

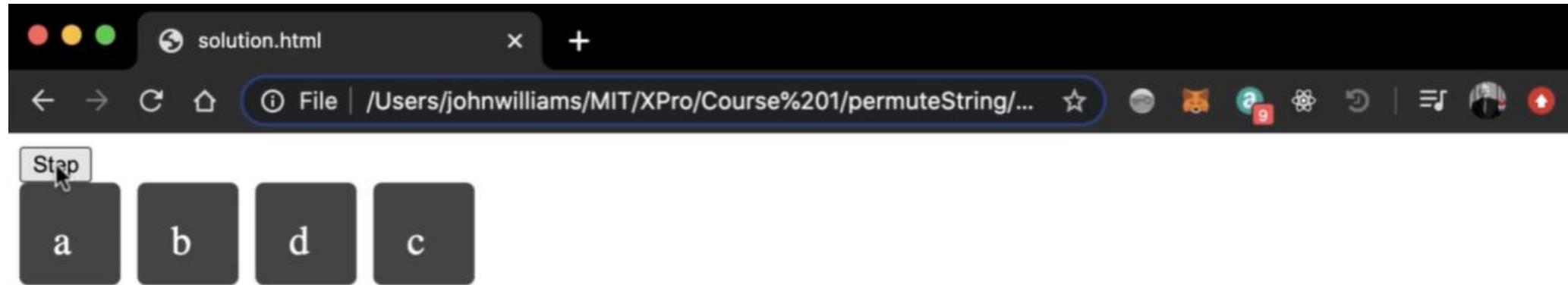
```
✓ pe 1 // permute a string
["ab" 2 function permute(A) {
dca" 3     let len = A.length;
", "d 4     if (len === 1) return A;
✓ pe 5     let pick = "";
, "ac 6     let permutations = [];
, "ae 7     for (let i = 0; i < len; i++) {
, "bc 8         pick = A[i];
, "bd 9         var remainder = A.slice(0, i) + A.slice(i + 1, len);
, "cd10        for (var perm of permute(remainder)) {
, "ce11            permutations.push(pick + perm);
, "db11        }
, "de12    }
, "ea13    }
, "ec14    return permutations;
] 14
✓ pe15  }
L6  | const A = "abcde";
L7  | console.log(JSON.stringify(permute(A)));
```

master  
,"bdac","b  
bca","dcab  
master  
e","acdeb"  
d","aebdc"  
d","baedc"  
e","bdcea"  
e","cabed"  
d","cbeda"  
d","cebda"  
e","dbaec"  
b","dceba"  
d","eacdb"  
d","ecadb"  
b","edcba"  
master

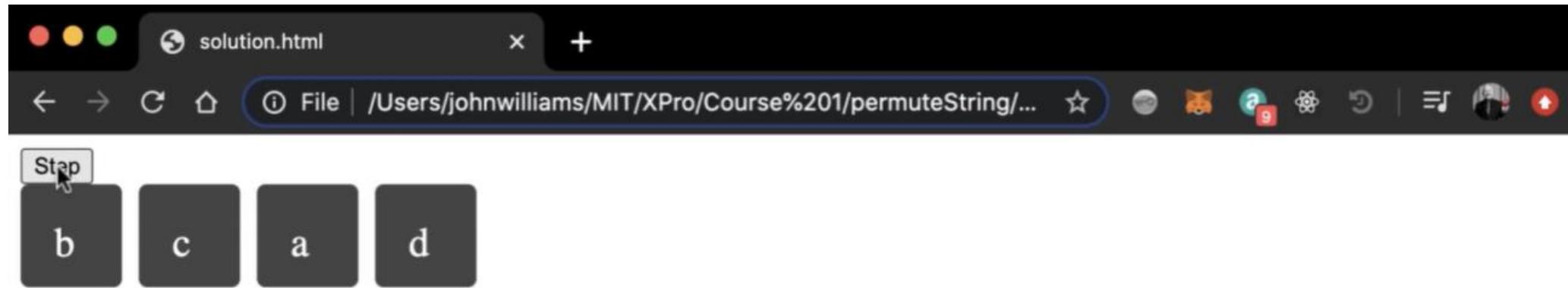
# Permute Exercise (1/8)



## Permute Exercise (2/8)



## Permute Exercise (3/8)



# Permute Exercise (4/8)

The screenshot shows a Microsoft Visual Studio Code interface. The Explorer sidebar on the left lists files: 'permute.js' (5), 'solution.html' (M), and 'starter.html' (highlighted). The 'PERMUTESTRING' folder contains 'permute.js' (5), 'solution.html' (M), and 'starter.html' (highlighted). The Editor pane on the right displays 'starter.html — permuteString'. The code includes an HTML structure with a script tag pointing to 'permute.js', a CSS style block defining a grid layout with 4 columns and 1 row, and a button with an 'onclick' event. The CSS for the grid includes 'display: grid', 'grid-template-columns: repeat(4, 60px)', 'grid-template-rows: repeat(1, 60px)', and 'grid-gap: 10px'. The CSS for the box includes 'background-color: #444', 'color: #fff', 'border-radius: 5px', 'padding: 20px', and 'font-size: 150%'. The Editor pane also shows line numbers from 1 to 22.

```
<html>
<script src=".//permute.js"></script>
<style>
.grid {
    display: grid;
    grid-template-columns: repeat(4, 60px);
    grid-template-rows: repeat(1, 60px);
    grid-gap: 10px;
}
.box {
    background-color: #444;
    color: #fff;
    border-radius: 5px;
    padding: 20px;
    font-size: 150%;
}
</style>
<button onclick="step()">Step</button>
<div id="grid" class="grid">
    <div class="box 1">a</div>
    <div class="box 2">b</div>
    <div class="box 3">c</div>
```

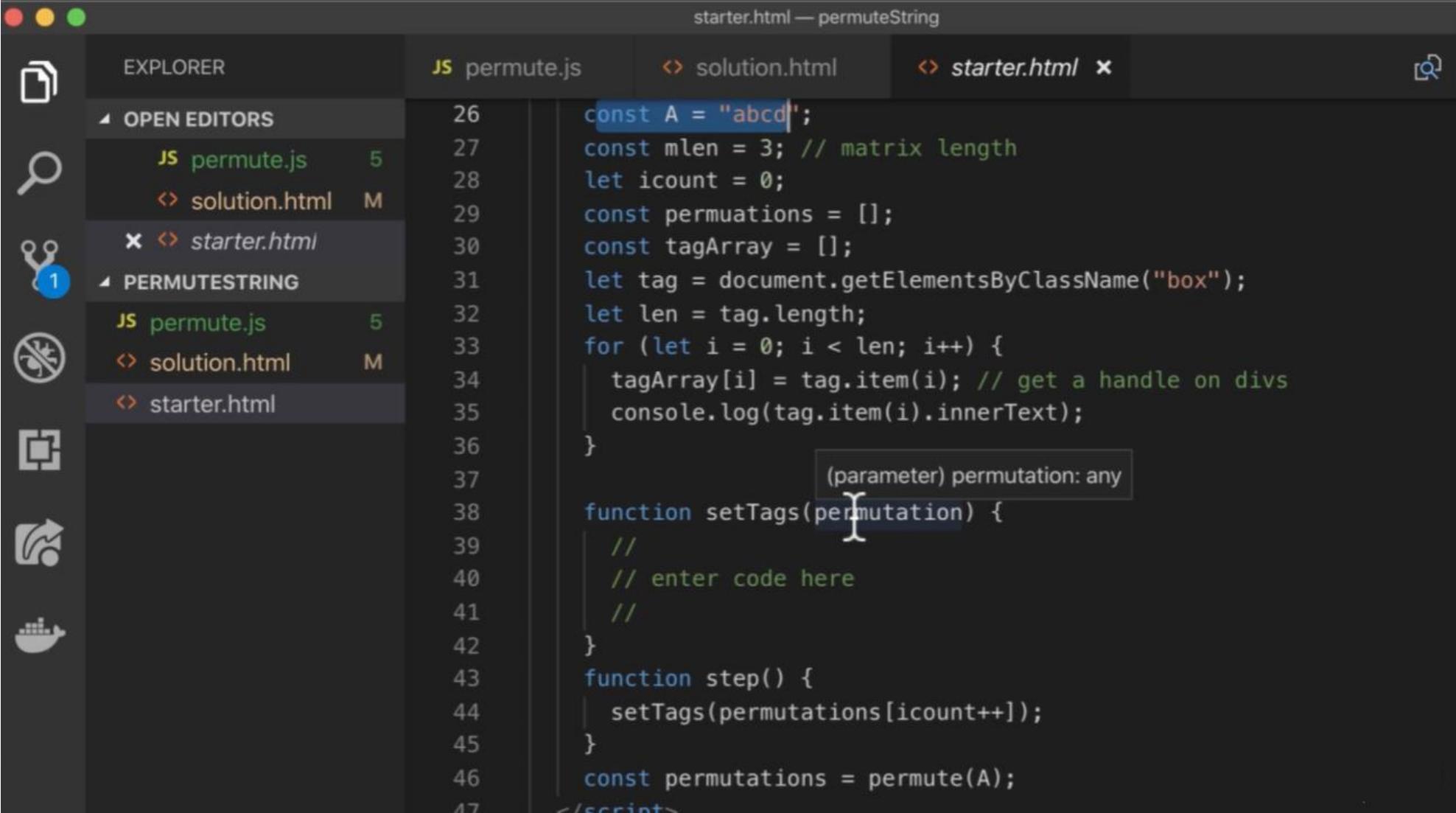
# Permute Exercise (5/8)

The screenshot shows a code editor interface with the following details:

- Title Bar:** starter.html — permuteString
- Explorer:** Shows three open editors:
  - permute.js (5 changes)
  - solution.html (M)
  - starter.html (highlighted)A folder named "PERMUTESTRING" also contains:
  - permute.js (5 changes)
  - solution.html (M)
  - starter.html (highlighted)
- Code Editor:** The "starter.html" file contains the following code:

```
20 <div class="box 1">a</div>
21 <div class="box 2">b</div>
22 <div class="box 3">c</div>
23 <div class="box 4">d</div>
24 </div>
25 <script>
26 const A = "abcd";
27 const mlen = 3; // matrix length
28 let icount = 0;
29 const permutations = [];
30 const tagArray = [];
31 let tag = document.getElementsByTagName("div");
32 let len = tag.length;
33 for (let i = 0; i < len; i++) {
34   tagArray[i] = tag.item(i); // get a handle on divs
35   console.log(tag.item(i).innerText);
36 }
37
38 function setTags(permuation) {
39   //
40   // enter code here
41   //
```

# Permute Exercise (6/8)



The screenshot shows a code editor interface with the following details:

- Explorer Sidebar:** On the left, there's a sidebar titled "OPEN EDITORS" containing three entries under a group named "PERMUTESTRING": "permute.js" (5), "solution.html" (M), and "starter.html".
- Code Editor:** The main area displays a script named "starter.html" with the following content:

```
const A = "abcd";
const mlen = 3; // matrix length
let icount = 0;
const permuations = [];
const tagArray = [];
let tag = document.getElementsByClassName("box");
let len = tag.length;
for (let i = 0; i < len; i++) {
    tagArray[i] = tag.item(i); // get a handle on divs
    console.log(tag.item(i).innerText);
}
function setTags(permuation) {
    //
    // enter code here
    //
}
function step() {
    setTags(permutations[icount++]);
}
const permutations = permute(A);
</script>
```
- Toolbars and Status:** At the top, there are standard OS-style window controls (red, yellow, green) and a status bar indicating "starter.html — permuteString".

# Permute Exercise (7/8)

The screenshot shows a code editor interface with the following details:

- Explorer Sidebar:** Shows the file structure under "OPEN EDITORS".
  - "PERMUTESTRING" folder:
    - permute.js
    - solution.html
    - starter.html (selected)
- Editor Area:** The "starter.html" file is open, showing the following code:

```
const permutations = [];
const tagArray = [];
let tag = document.getElementsByClassName("box");
let len = tag.length;
for (let i = 0; i < len; i++) {
    tagArray[i] = tag.item(i); // get a handle on divs
    console.log(tag.item(i).innerText);
}

function setTags(permuation) {
    //
    // enter code here
    //
}
function step() {
    setTags(permutations[icount++]);
}
const permutations = permute(A);
</script>
</html>
```

# Permute Exercise (8/8)

The screenshot shows a code editor interface with the following details:

- Title Bar:** starter.html — permuteString
- Explorer:** Shows three open editors:
  - permute.js (5 changes)
  - solution.html (M)
  - starter.html (highlighted)A blue circle with the number 1 is positioned next to the PERMUTESTRING folder icon.
- Code Editor:** Displays the following JavaScript code:

```
23 | <div class="box 4">d</div>
24 |
25 |
26 | <script>
27 | const A = "abcd";
28 | const mlen = 3; // matrix length
29 | let icount = 0;
30 | const permutations = [];
31 | const tagArray = [];
32 | let tag = document.getElementsByClassName("box");
33 | let len = tag.length;
34 | for (let i = 0; i < len; i++) {
35 |   tagArray[i] = tag.item(i); // get a handle on divs
36 |   console.log(tag.item(i).innerText);
37 |
38 |   function setTags(permuation) {
39 |     //
40 |     // enter code here
41 |     //
42 |   }
43 |   function step() {
44 |     setTags(permutations[icount++]);
45 |   }
46 | }
```

## Rotate An Image 90 Degrees Clockwise (1/2)



## Rotate An Image 90 Degrees Clockwise (2/2)

Matrix A

1	2	3
4	5	6
7	8	9



A Rotated

1	4	7
2	5	8
3	6	9

# Transpose A Matrix – Make The Rows Into Columns (1/8)

Rotate Matrix 90 Degrees Clockwise

Matrix A

1	2	3
4	5	6
7	8	9



Transpose of A

1	4	7
2	5	8
3	6	9

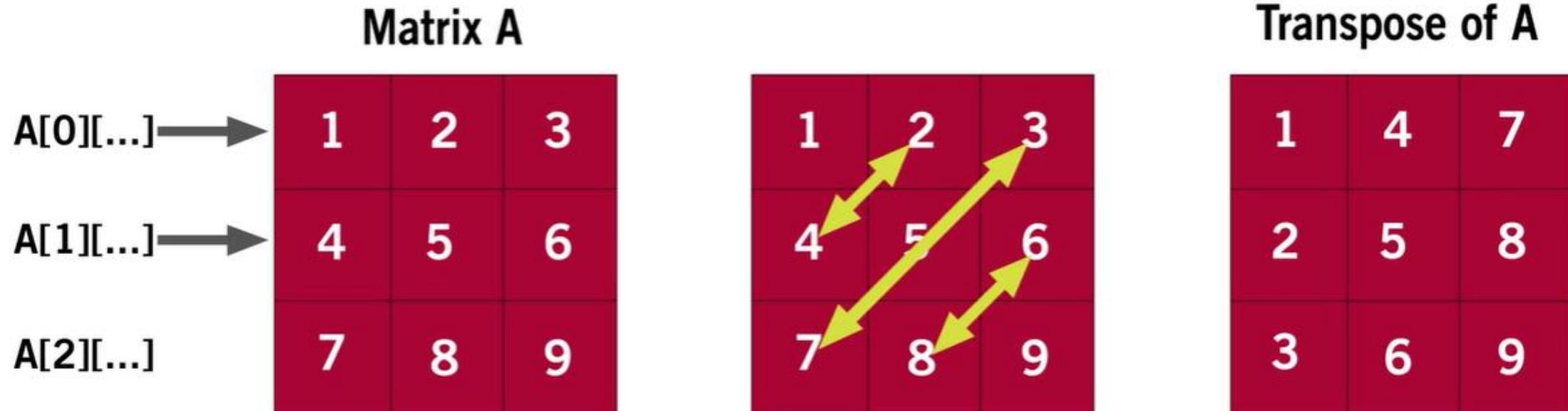
A Rotated

1	2	3
4	5	6
7	8	9



7	4	1
8	5	2
9	6	3

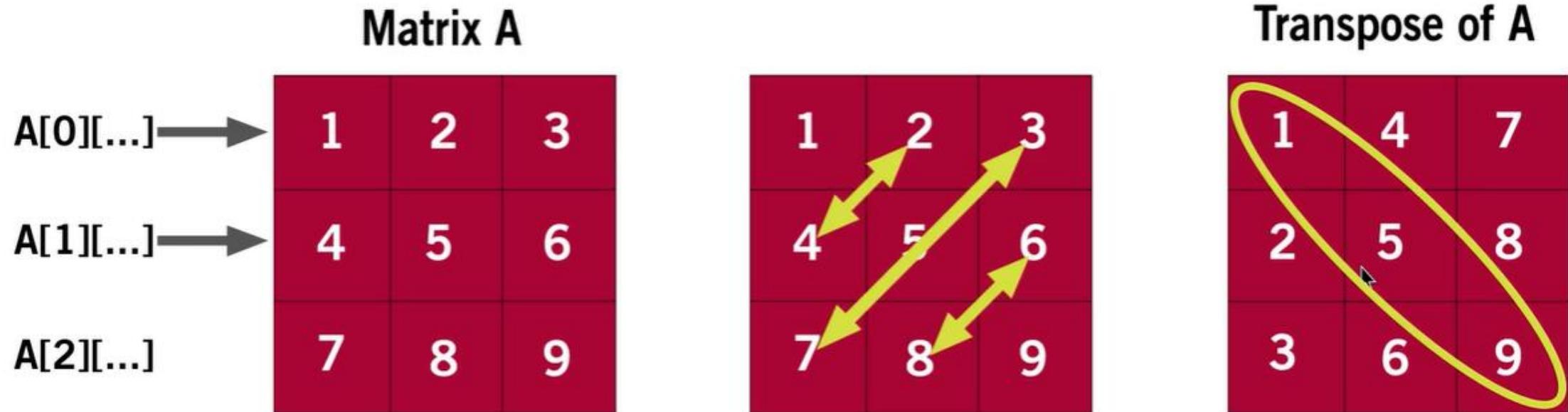
## Transpose A Matrix – Make The Rows Into Columns (2/8)



$A = [[1, 2, 3],$   
 $[4, 5, 6],$   
 $[7, 8, 9]]$

SWAP  $A[i][j]$  with  $A[j][i]$   
SWAP  $A[0][1]$  – with  $A[1][0]$   
SWAP  $A[0][2]$  – with  $A[2][0]$   
SWAP  $A[1][2]$  – with  $A[2][1]$

## Transpose A Matrix – Make The Rows Into Columns (3/8)



$A = [ [1, 2, 3],$   
   $[4, 5, 6],$   
   $[7, 8, 9] ]$

**SWAP  $A[i][j]$  with  $A[j][i]$**   
**SWAP  $A[0][1]$  – with  $A[1][0]$**   
**SWAP  $A[0][2]$  – with  $A[2][0]$**   
**SWAP  $A[1][2]$  – with  $A[2][1]$**

QUESTION

## Transpose A Matrix – Make The Rows Into Columns (4/8)

```
17  
18  
19  
20  
21 |  
22  
23  
24  
25  
26 function swap(A, i, j, k, l) {  
27 | let temp = A[j][i];  
28 | A[j][i] = A[l][k];  
29 | A[l][k] = temp;  
30 }  
31  
32  
33  
34  
35  
36
```

## Transpose A Matrix – Make The Rows Into Columns (5/8)

```
6
7  function transpose(A) {
8    // we need only swap upper triangle with lower
9    let N = A.length;
10   for (let i = 0; i < N - 1; i++) {
11     for (let j = i + 1; j < N; j++) {
12       swap(A, i, j, j, i);
13     }
14   }
15 }
16
17
18 function swap(A, i, j, k, l) {
19   let temp = A[j][i];
20   A[j][i] = A[l][k];
21   A[l][k] = temp;
22 }
```

## Transpose A Matrix – Make The Rows Into Columns (6/8)

```
25
26
27
28
29
30 function exchangeCols(A) {
31     // work in from outside
32     let N = A.length; // we need only swap N/2 (rounded down)
33     let n = Math.floor(N / 2); // round down to integer
34     for (let col = 0; col < n; col++) {
35         for (let row = 0; row < N; row++) {
36             swap(A, col, row, N - 1 - col, row);
37         }
38     }
39 }
40
41 function rotateMatrix(A) {
42     transpose(A);
43     exchangeCols(A);
44 }
45 console.log('Original A = ' + JSON.stringify(A));
```

## Transpose A Matrix – Make The Rows Into Columns (7/8)

```
33     let n = Math.floor(N / 2); // round down to integer
34     for (let col = 0; col < n; col++) {
35         for (let row = 0; row < N; row++) {
36             swap(A, col, row, N - 1 - col, row);
37         }
38     }
39 }
40
41 function rotateMatrix(A) {
42     transpose(A);
43     exchangeCols(A);
44 }
45 console.log('Original A = ' + JSON.stringify(A));
46 transpose(A);
47 console.log('Transpose A = ' + JSON.stringify(A));
48 exchangeCols(A);
49 console.log('A Rotated = ' + JSON.stringify(A));
```

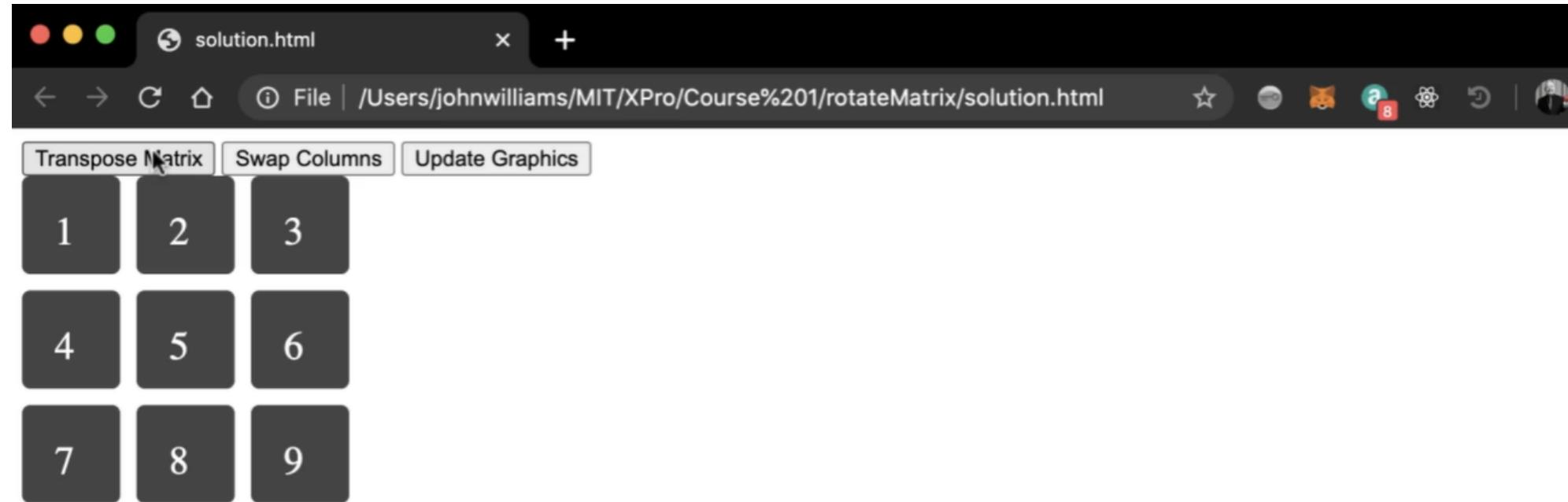
# Transpose A Matrix – Make The Rows Into Columns (8/8)

```
1 | const A = [[1, 2, 3],[4, 5, 6],[7, 8, 9]];
2 | function transpose(A) {
3 |   // we need only swap upper triangle with lower
4 |   let N = A.length;
5 |   for (let i = 0; i < N - 1; i++) {
6 |     for (let j = i + 1; j < N; j++) {
7 |       swap(A, i, j, j, i);
8 |     }
9 |   }
10 | }
11 | function swap(A, i, j, k, l) {
12 |   let temp = A[j][i];
13 |   A[j][i] = A[l][k];
14 |   A[l][k] = temp;
15 | }
16 |+ function exchangeCols(A) { ...
25 | }
26 | function rotateMatrix(A) {
27 |   transpose(A);
28 |   exchangeCols(A);
29 | }
30 | console.log('Original A = ' + JSON.stringify(A));
31 | transpose(A);
32 | console.log('Transpose A = ' + JSON.stringify(A));
33 | exchangeCols(A);
34 | console.log('A Rotated = ' + JSON.stringify(A));
```



Try now to write  
exchangeCols

# Transpose Matrix Exercise (1/4)



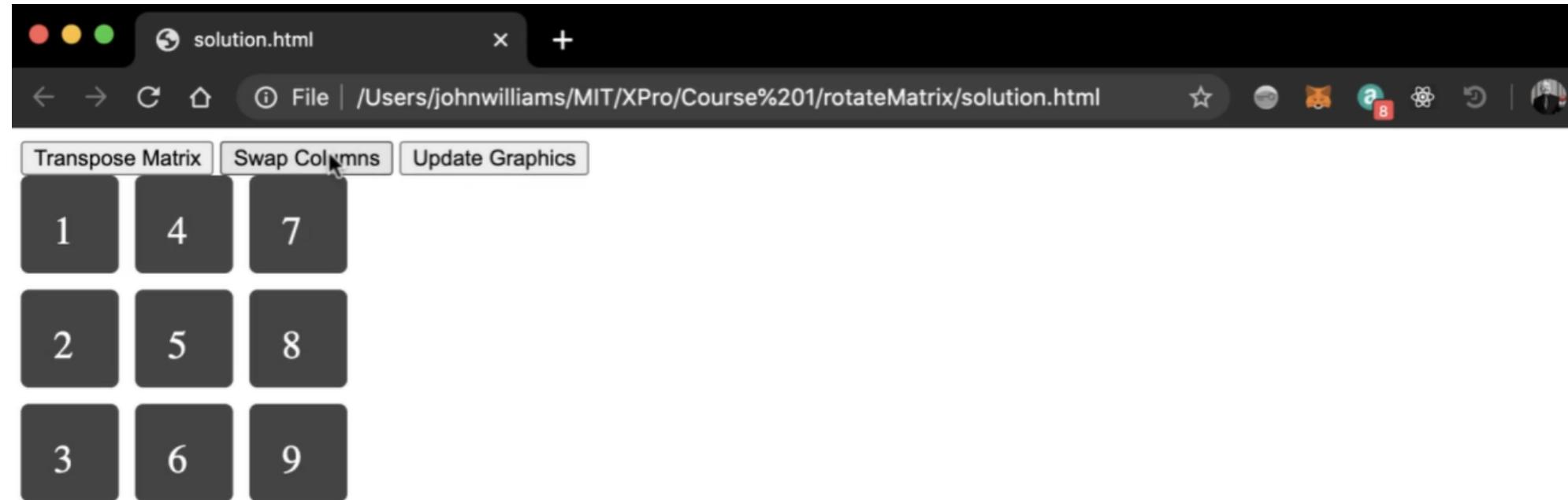
# Transpose Matrix Exercise (2/4)

A screenshot of a web browser window titled "solution.html". The address bar shows the URL: "/Users/johnwilliams/MIT/XPro/Course%201/rotateMatrix/solution.html". The browser interface includes standard controls like back, forward, and search.

The main content area displays a 3x3 grid of dark gray boxes containing numbers 1 through 9. Above the grid are three buttons: "Transpose Matrix", "Swap Columns", and "Update Graphics". The "Update Graphics" button is currently active, indicated by a mouse cursor hovering over it.

1	4	7
2	5	8
3	6	9

# Transpose Matrix Exercise (3/4)



# Transpose Matrix Exercise (4/4)

A screenshot of a web browser window titled "solution.html". The address bar shows the file path: "/Users/johnwilliams/MIT/XPro/Course%201/rotateMatrix/solution.html". The browser interface includes standard controls like back, forward, and search.

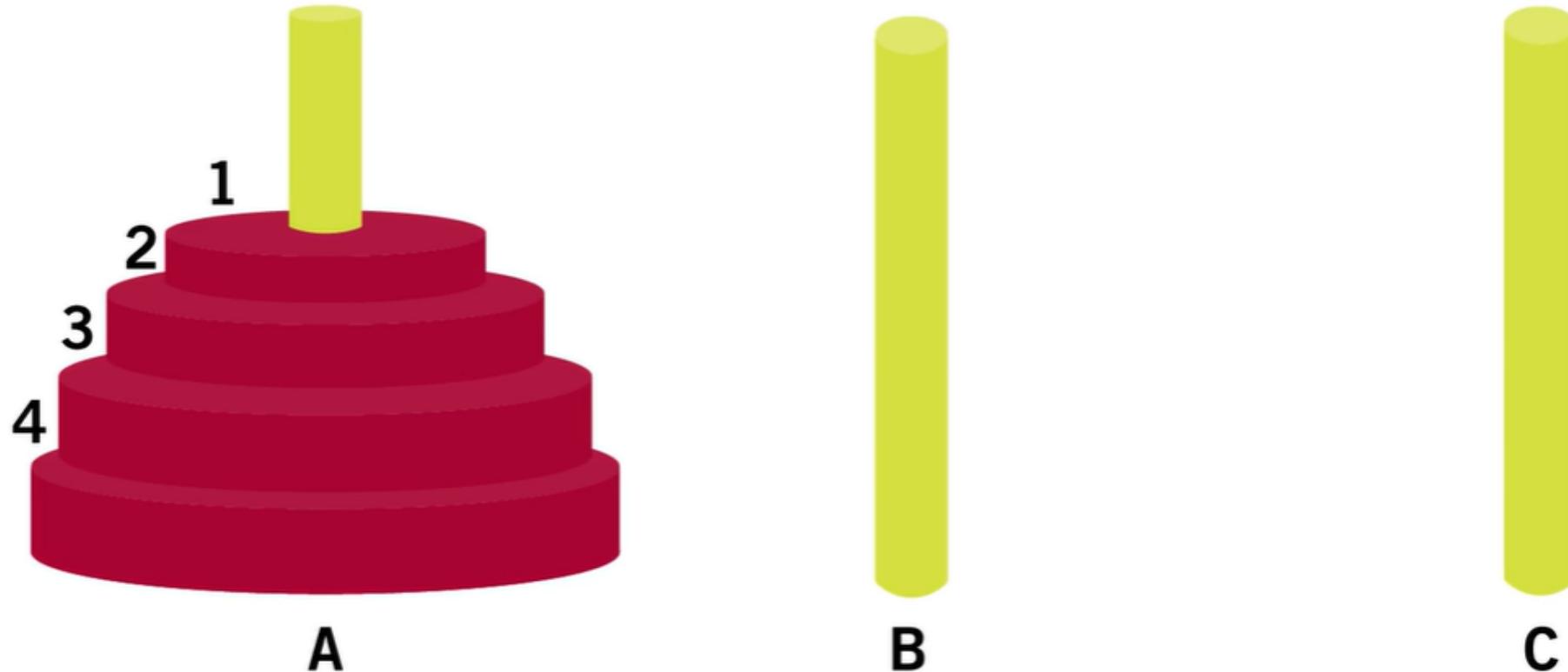
The main content area displays a 3x3 grid of dark gray boxes containing numbers. The first row contains 7, 4, and 1. The second row contains 8, 5, and 2. The third row contains 9, 6, and 3. The number 6 is highlighted with a white background and black border, indicating it is currently selected or being edited. A cursor arrow is positioned over the number 6.

Below the grid, there are three buttons: "Transpose Matrix", "Swap Columns", and "Update Graphics".

7	4	1
8	5	2
9	6	3

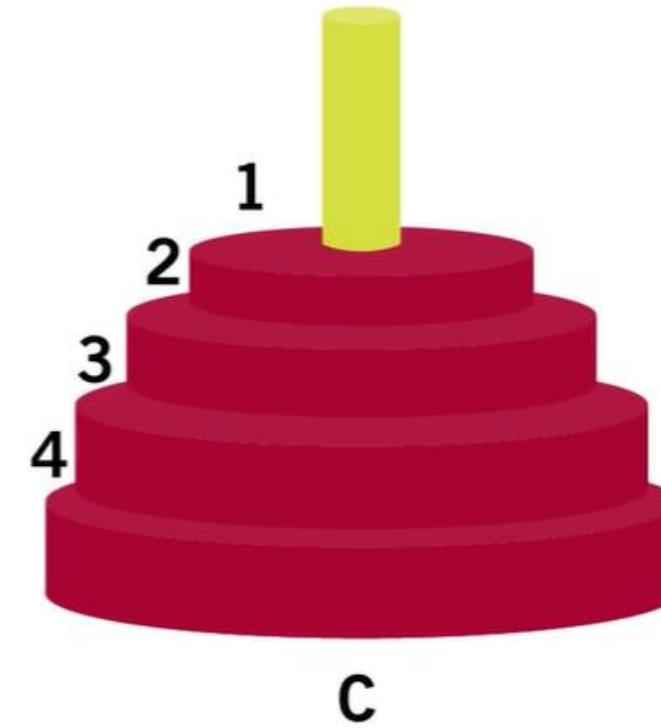
# Tower Of Hanoi (1/19)

- Goal Move all disks from A to C
- Only top disc of any stack can be moved
- It must be moved to sit on a larger disk



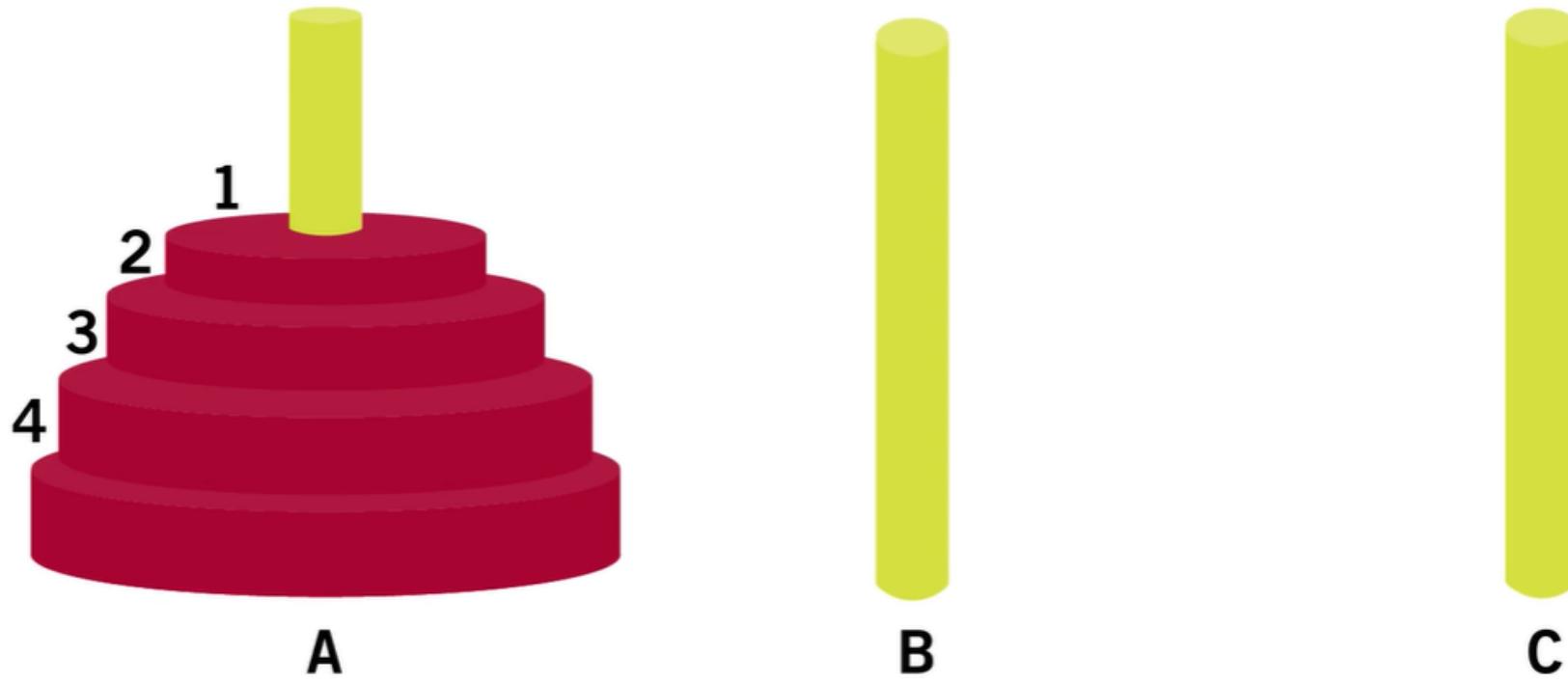
# Tower Of Hanoi (2/19)

Move Disks From A To C



# Tower Of Hanoi (3/19)

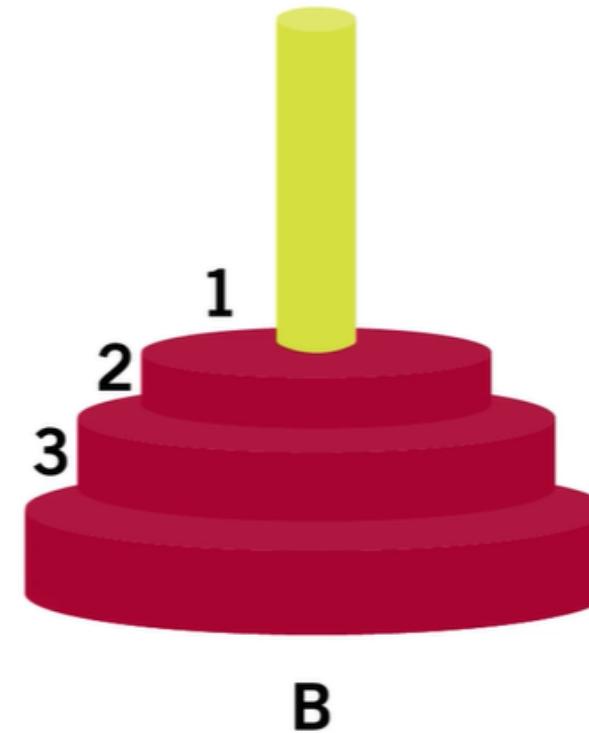
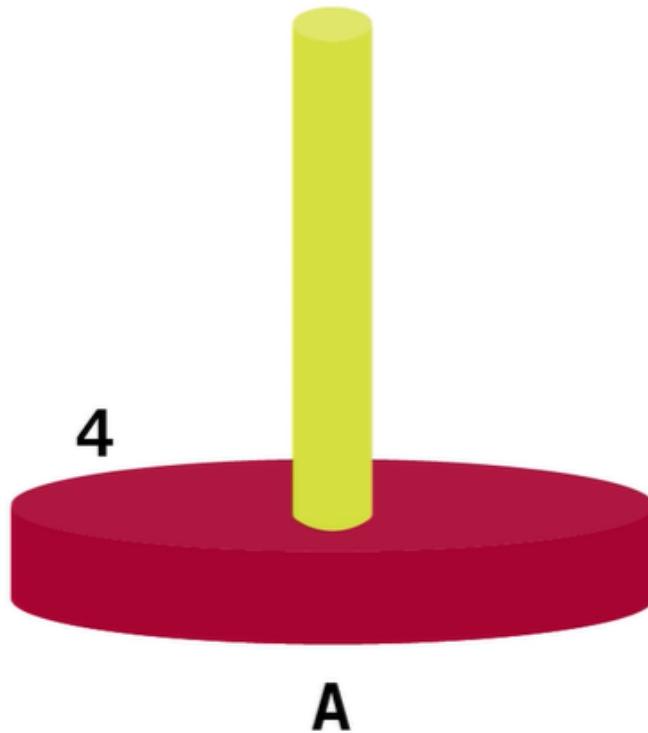
Move n disks from A to C, using B  
(n, A, C, B)



Move n-1 (n-1, A, B, C)

# Tower Of Hanoi (4/19)

Move (n-1, A, B, C)

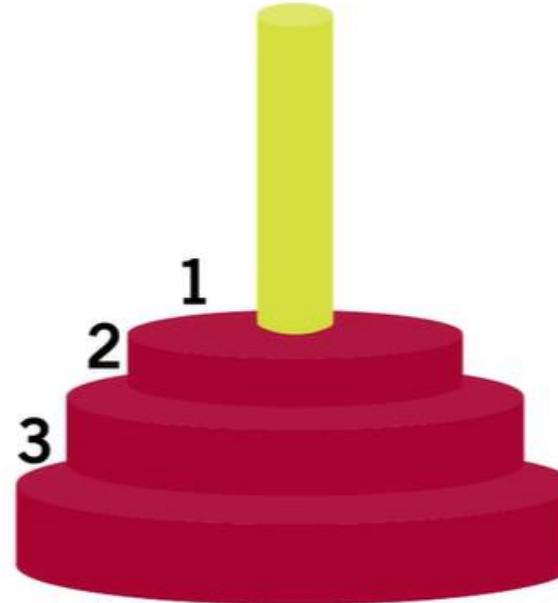


# Tower Of Hanoi (5/19)

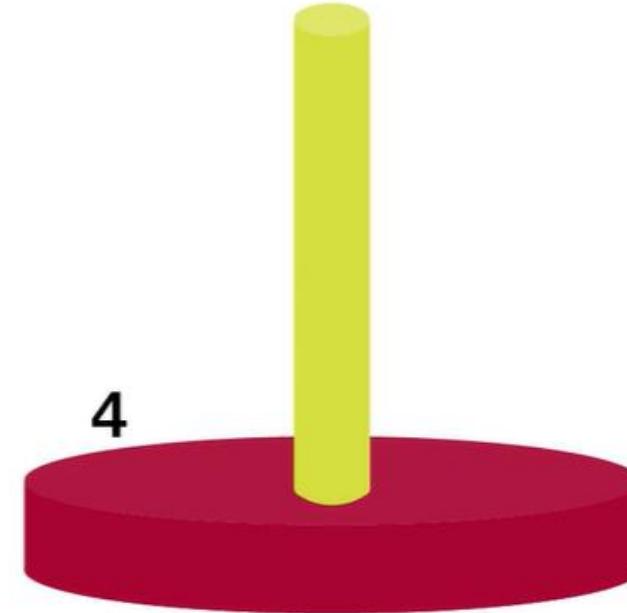
Move (n-1, A, B, C)  
Move disk n from A to C



A



B



C

# Tower Of Hanoi (6/19)

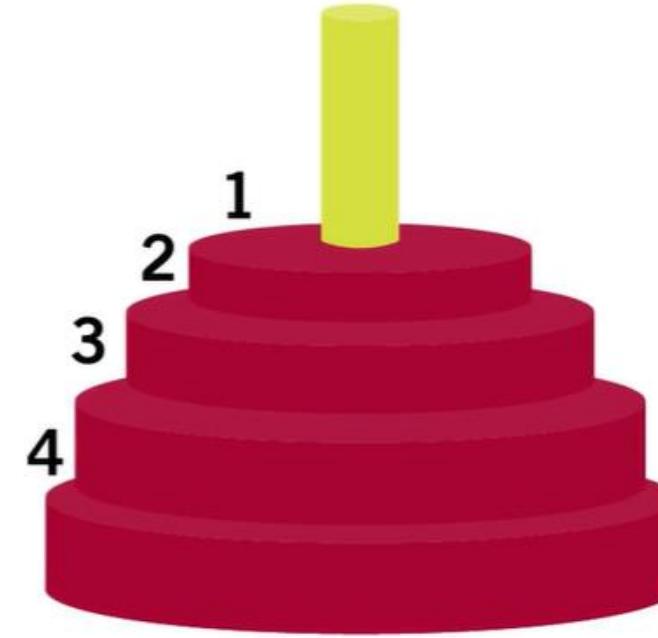
Move ( $n - 1$ , A, B, C)  
Move disk  $n$  from A to C  
Move ( $n - 1$ , B, C, A)



A



B



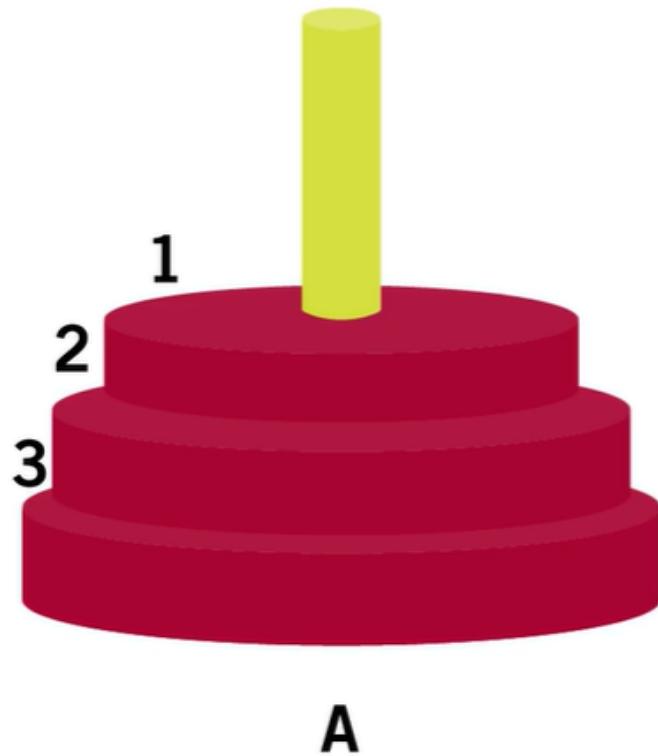
C

# Tower Of Hanoi (7/19)

$\text{Move } (3, A, C, B) = \text{Move } (2, A, B, C)$

Move disk 3 A to C

Move (2, B, C, A)

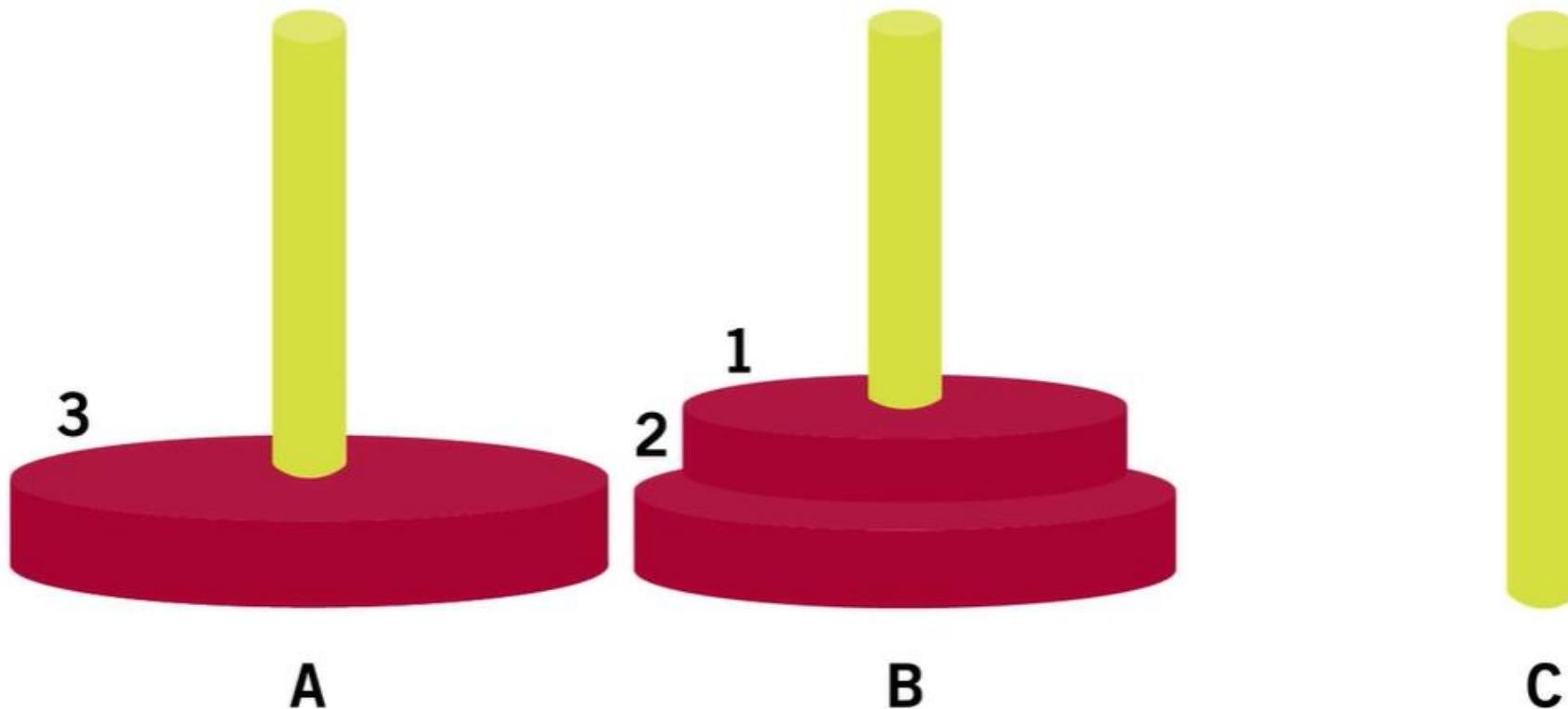


## Tower Of Hanoi (8/19)

Move (3, A, C, B) = Move (2, A, B, C)

Move disk 3 A to C

Move (2, B, C, A)



# Tower Of Hanoi (9/19)

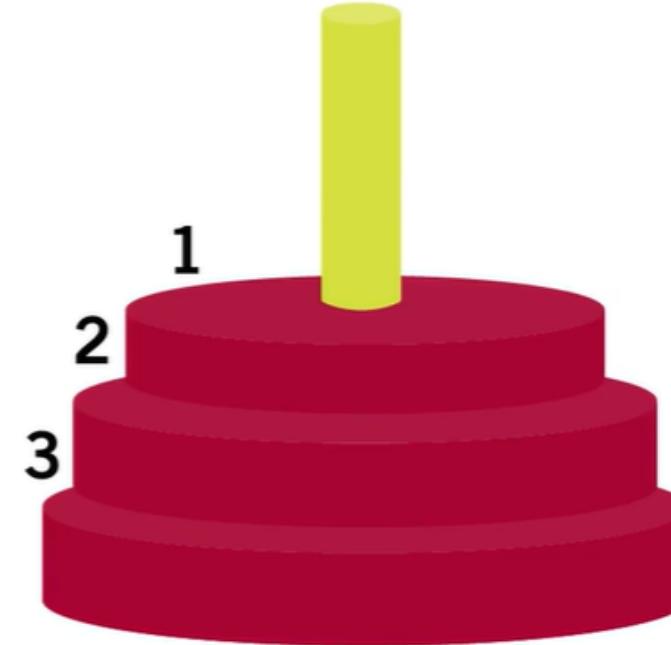
Move (3, A, C, B) = Move (2, A, B, C)  
Move disk 3 A to C  
Move (2, B, C, A)



A

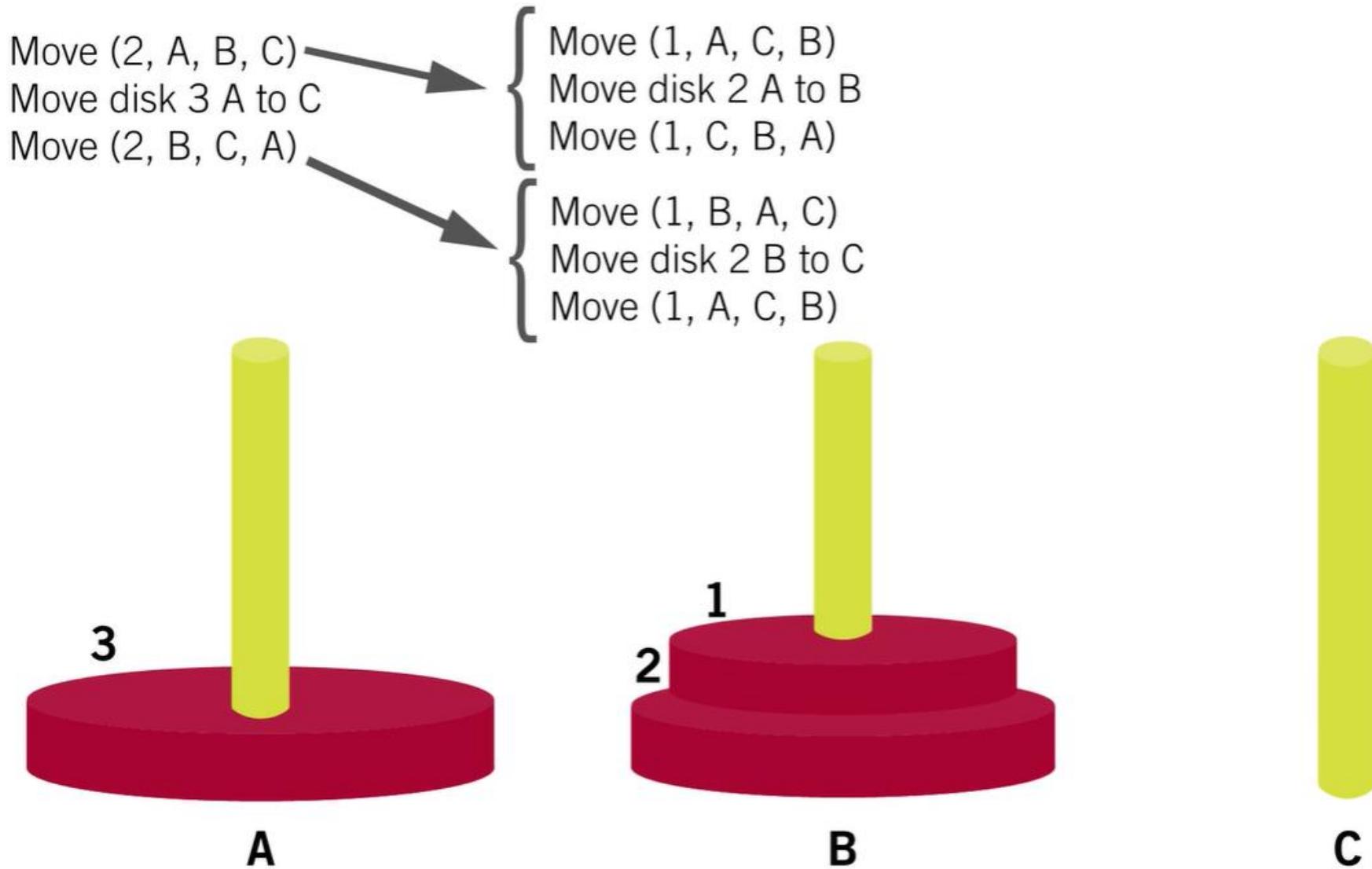


B

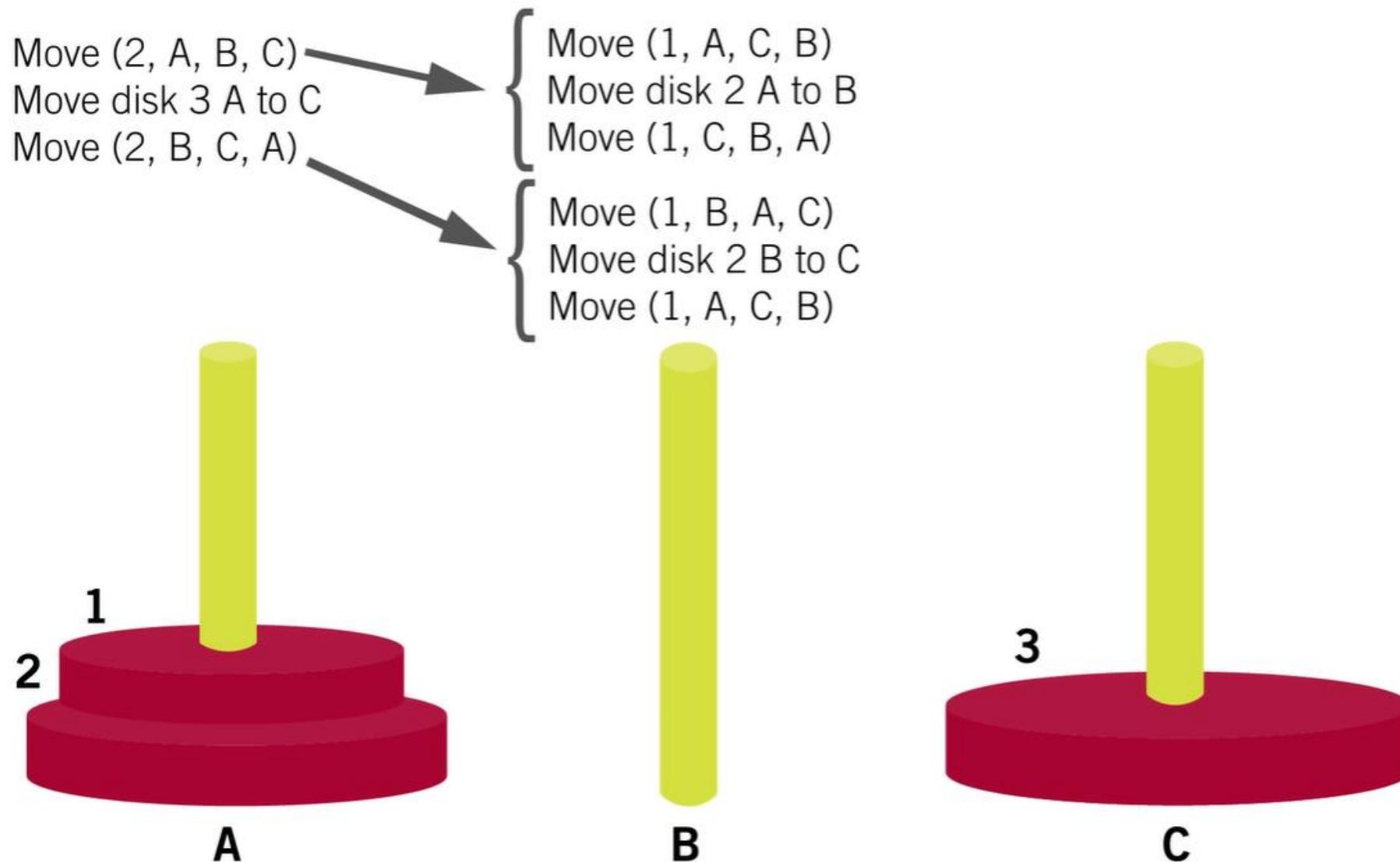


C

# Tower Of Hanoi (10/19)

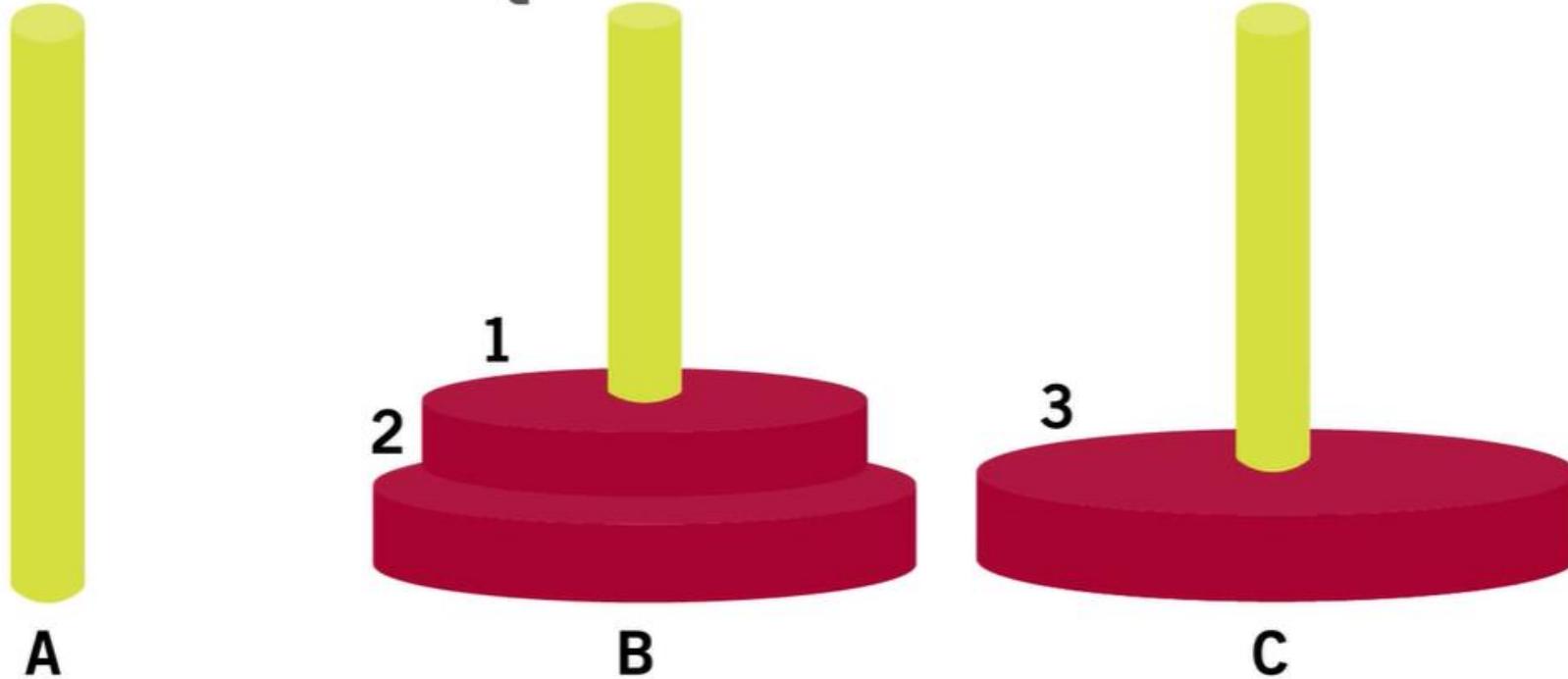


## Tower Of Hanoi (11/19)

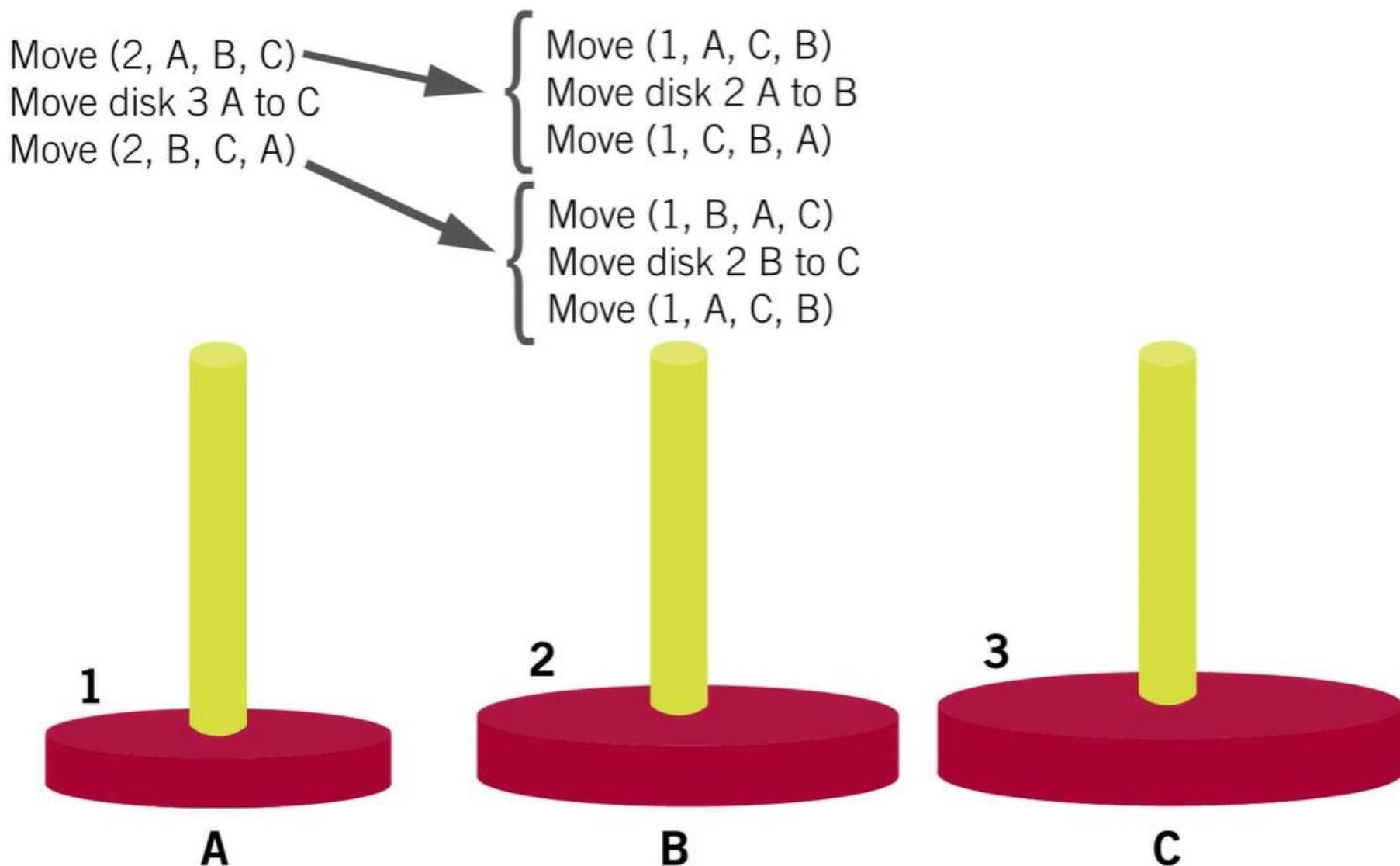


## Tower Of Hanoi (12/19)

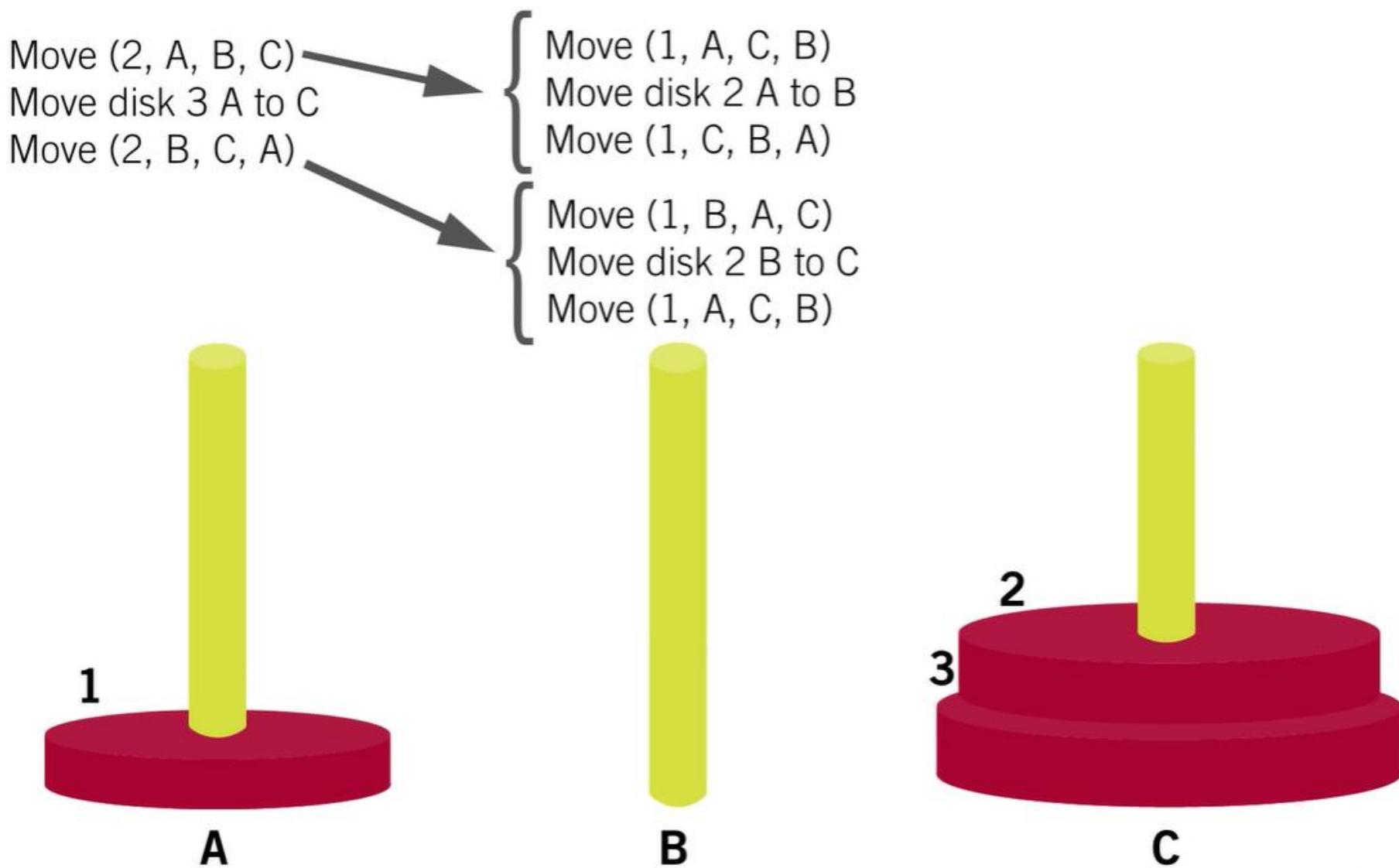
Move (2, A, B, C) → { Move (1, A, C, B)  
Move disk 3 A to C  
Move (2, B, C, A) → { Move (1, C, B, A)  
Move (1, B, A, C)  
Move disk 2 B to C  
Move (1, A, C, B)



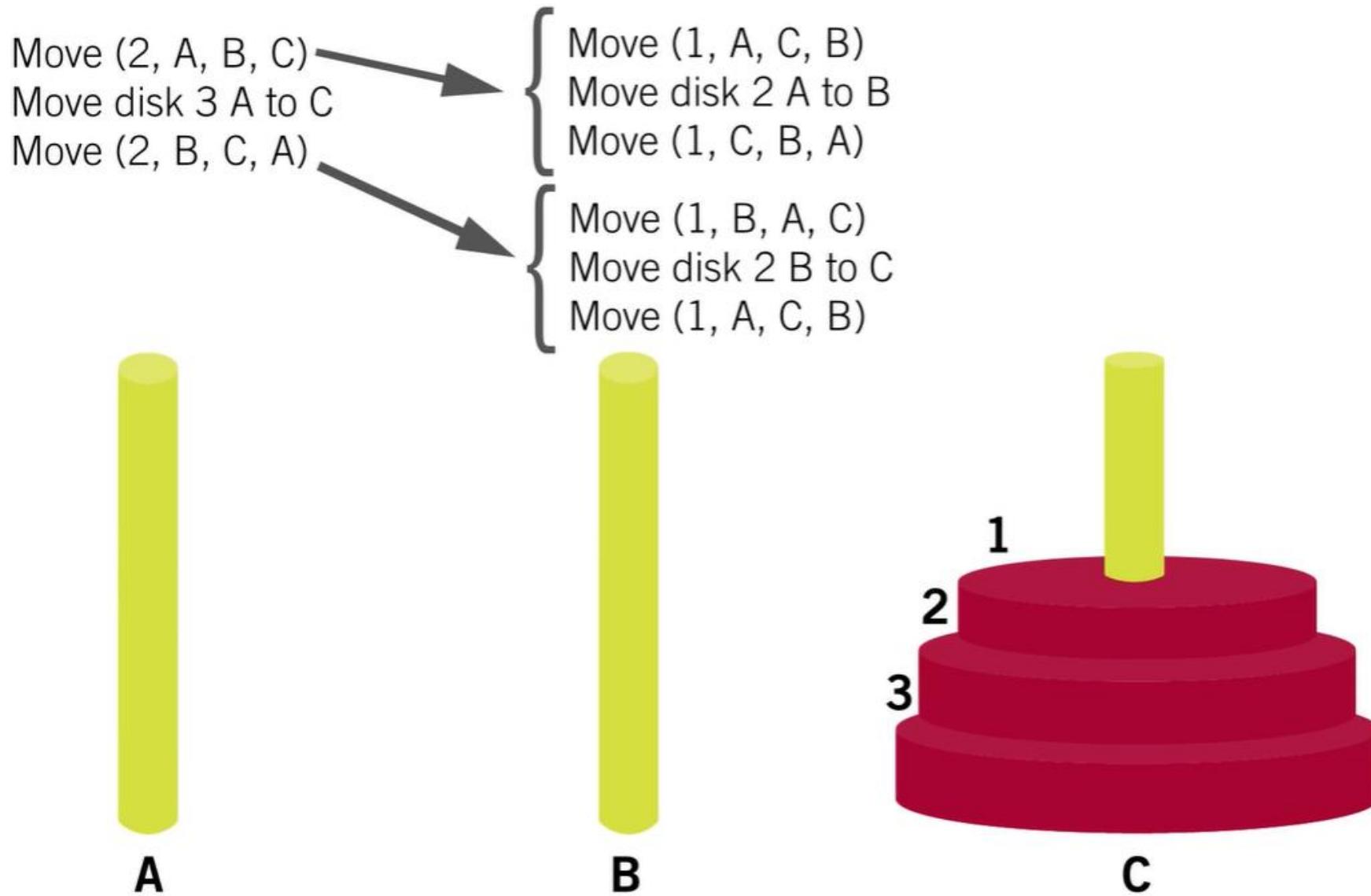
## Tower Of Hanoi (13/19)



## Tower Of Hanoi (14/19)

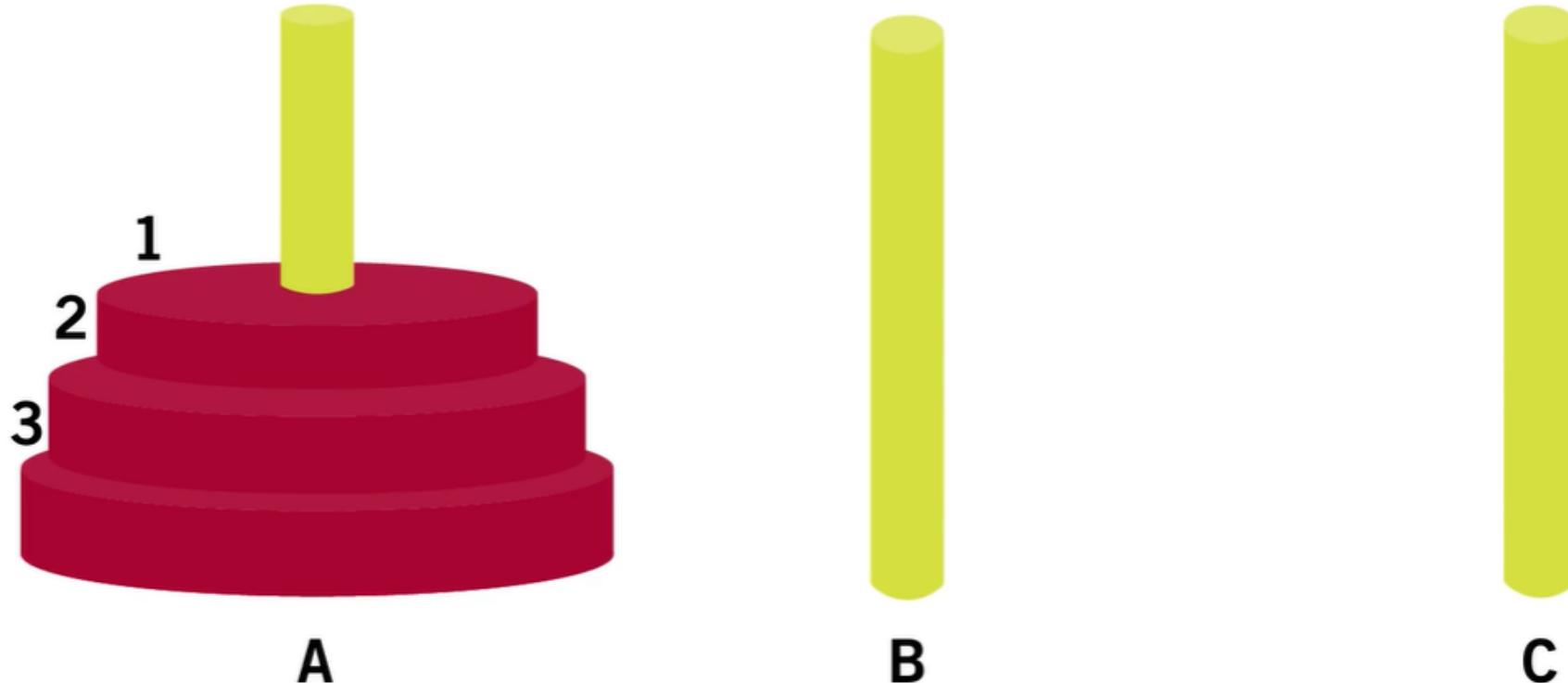


## Tower Of Hanoi (15/19)



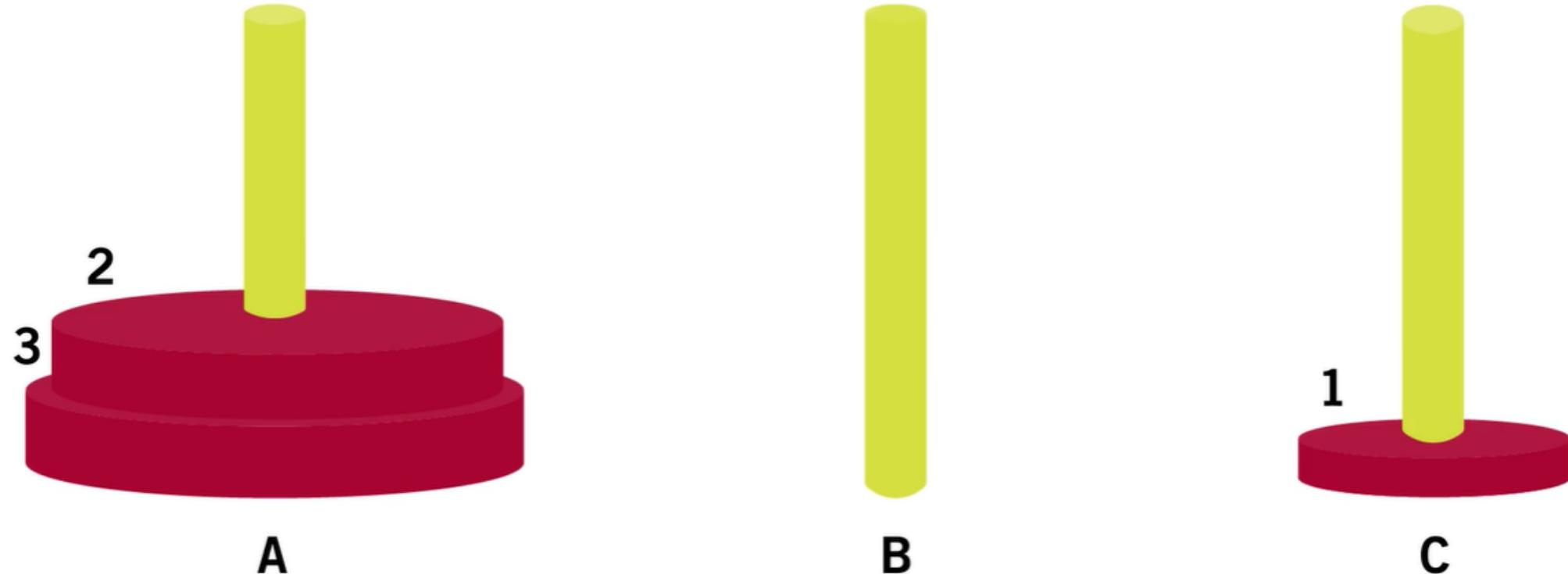
# Tower Of Hanoi (16/19)

Move (2, A, B, C) = Move (1, A, C, B){if (n = 1): Move disk 1 A to C}  
Move disk 2 A to B  
Move (1, C, B, A){if (n = 1): Move disk C to B}



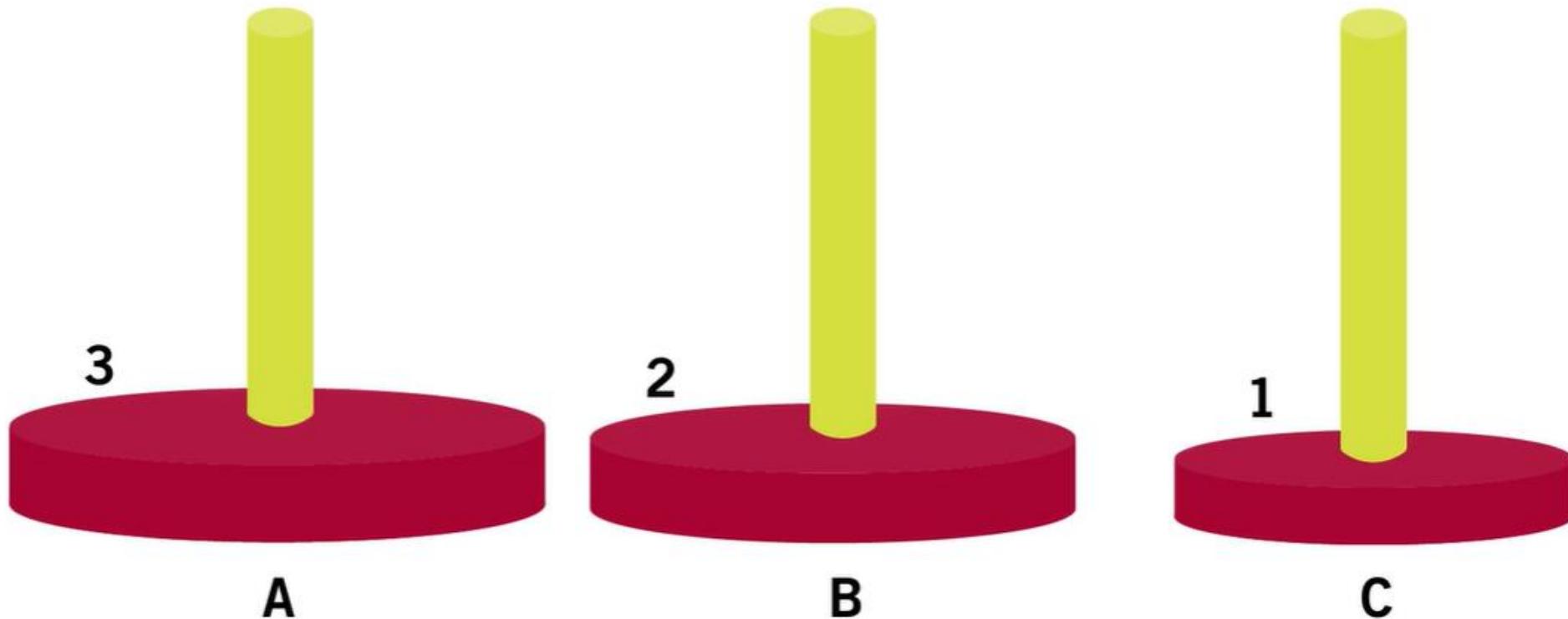
# Tower Of Hanoi (17/19)

Move (2, A, B, C) = Move (1, A, C, B){if (n = 1): Move disk 1 A to C}  
Move disk 2 A to B  
Move (1, C, B, A){if (n = 1): Move disk C to B}



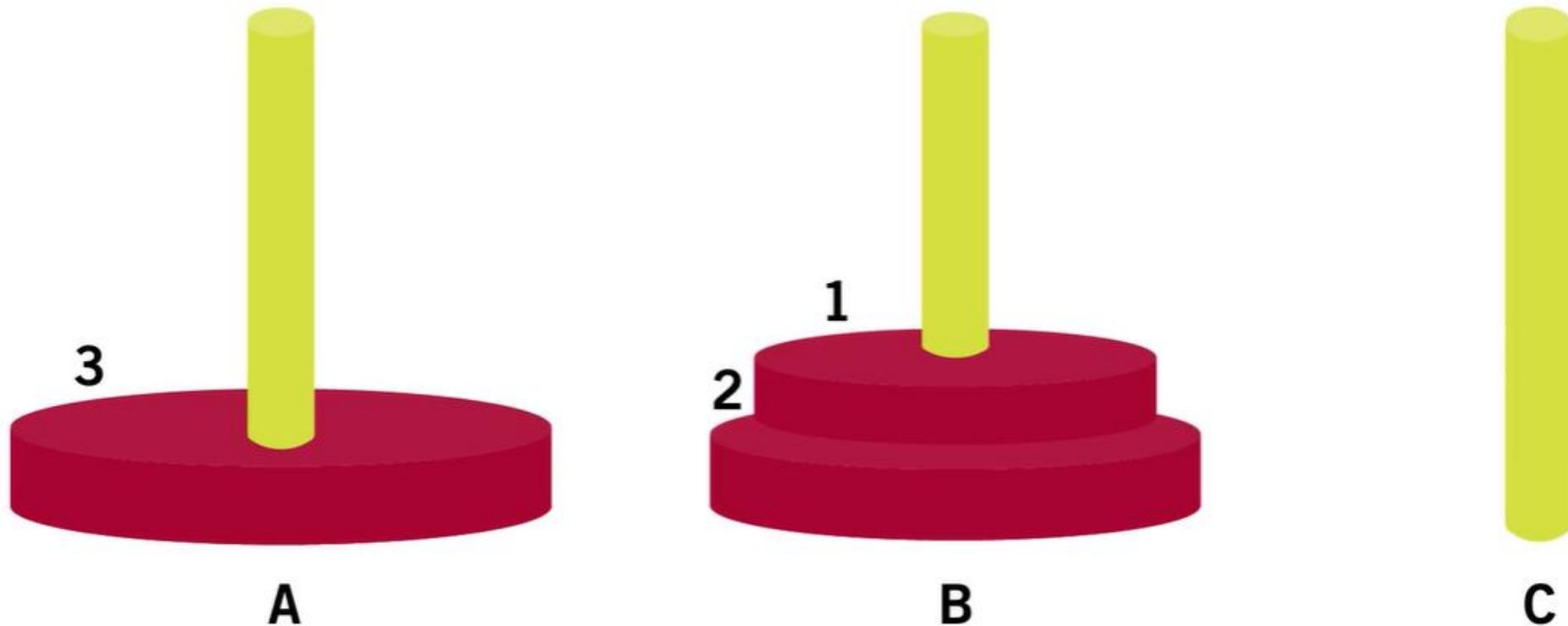
# Tower Of Hanoi (18/19)

Move (2, A, B, C) = Move (1, A, C, B){if (n = 1): Move disk 1 A to C}  
Move disk 2 A to B  
Move (1, C, B, A){if (n = 1): Move disk C to B}



# Tower Of Hanoi (19/19)

```
Move (2, A, B, C) = Move (1, A, C, B){if (n = 1): Move disk 1 A to C}  
                      Move disk 2 A to B  
                      Move (1, C, B, A){if (n = 1): Move disk C to B}
```



# Tower Of Hanoi: Code

```
const moveDisks = function (n, from, to, spare) {
    if (n == 1) {
        console.log("Move disk:" + n + " from:" + from + " to:" + to);
    } else {

        moveDisks(n - 1, from, spare, to);
        console.log("Move disk:" + n + " from:" + from + " to:" + to);
        moveDisks(n - 1, spare, to, from);
    }
};

const nDisks = 2;

moveDisks(nDisks, "A", "C", "B");
```

## Tower Of Hanoi: Output (1/4)

```
✓ hanoitower % node hanoi.js                                master
Move disk:1 from:A to:B
Move disk:2 from:A to:C
Move disk:1 from:B to:C
✓ hanoitower %                                              master
```

## Tower Of Hanoi: Output (2/4)

```
✓ hanoitower % node hanoi.js                                master
Move disk:1 from:A to:B
Move disk:2 from:A to:C
Move disk:1 from:B to:C
✓ hanoitower % node hanoi.js                                master
Move disk:1 from:A to:C
Move disk:2 from:A to:B
Move disk:1 from:C to:B
Move disk:3 from:A to:C
Move disk:1 from:B to:A
Move disk:2 from:B to:C
Move disk:1 from:A to:C
✓ hanoitower % █                                            master
```

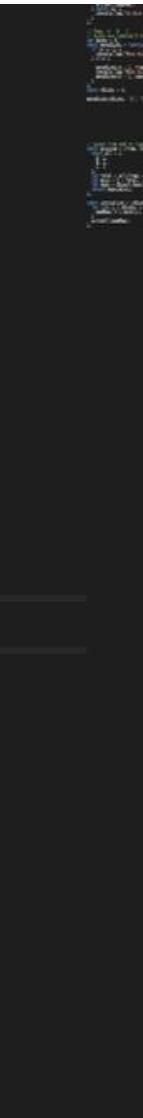
## Tower Of Hanoi: Output (3/4)

```
const moveDisks = function (n, from, to, spare) {
  if (n == 1) {
    console.log("Move disk:" + n + " from:" + from + " to:" + to);
  } else {

    moveDisks(n - 1, from, spare, to);
    console.log("Move disk:" + n + " from:" + from + " to:" + to);
    moveDisks(n - 1, spare, to, from);
  }
};

const nDisks = 4;

moveDisks(nDisks, "A", "C", "B");
```



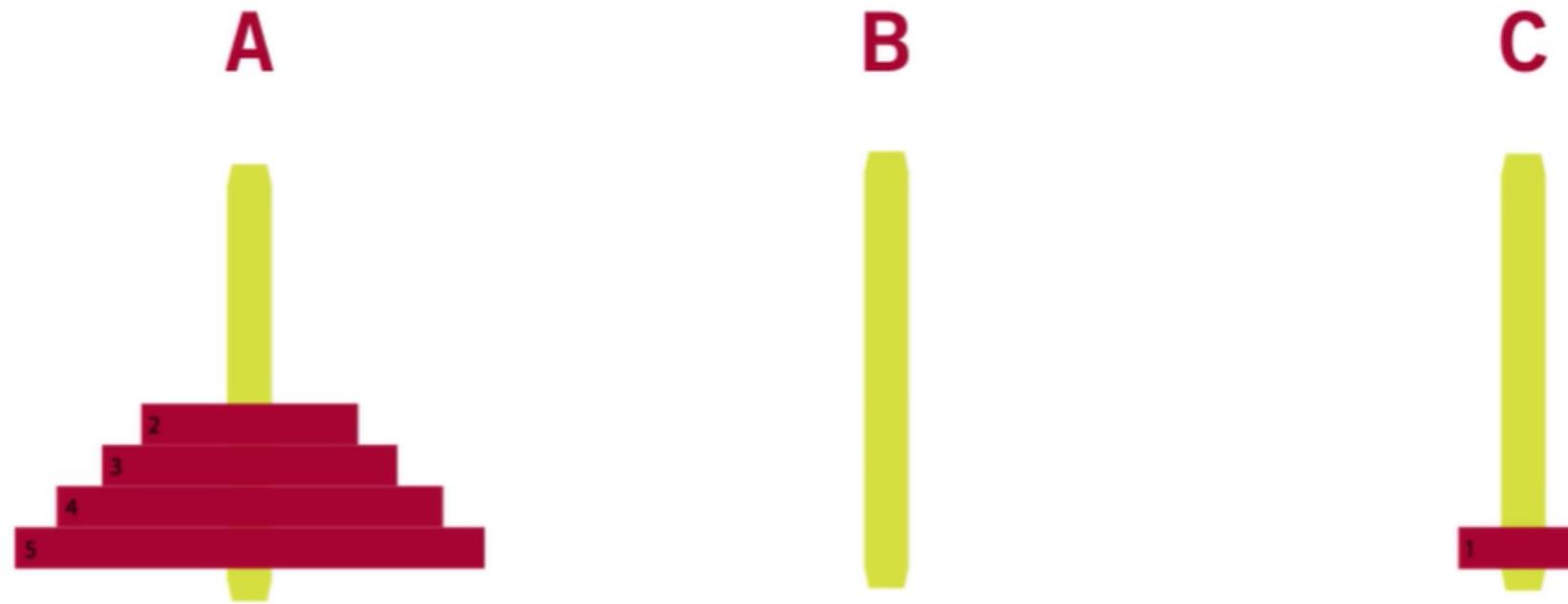
## Tower Of Hanoi: Output (4/4)

```
✓ hanoitower % node hanoi.js                                master
Move disk:1 from:A to:B
Move disk:2 from:A to:C
Move disk:1 from:B to:C
Move disk:3 from:A to:B
Move disk:1 from:C to:A
Move disk:2 from:C to:B
Move disk:1 from:A to:B
Move disk:4 from:A to:C
Move disk:1 from:B to:C
Move disk:2 from:B to:A
Move disk:1 from:C to:A
Move disk:3 from:B to:C
Move disk:1 from:A to:B
Move disk:2 from:A to:C
Move disk:1 from:B to:C
✓ hanoitower %
```

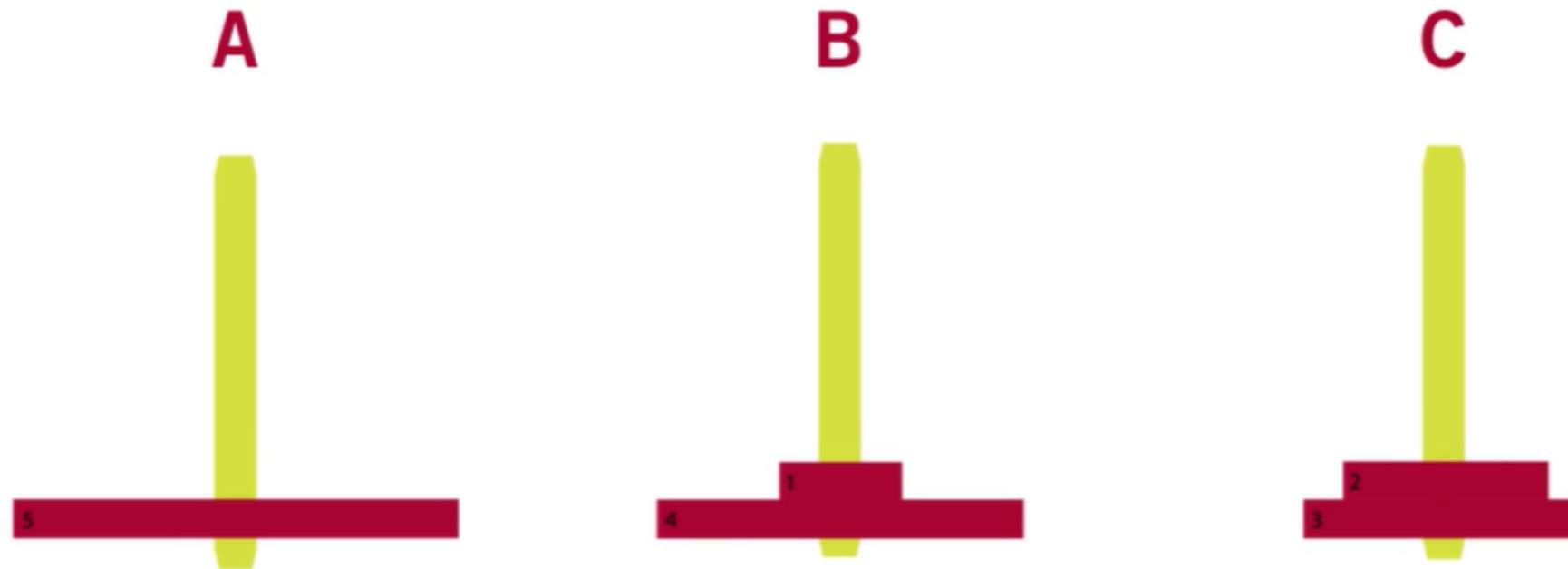
# Hanoi Tower Solution (1/7)



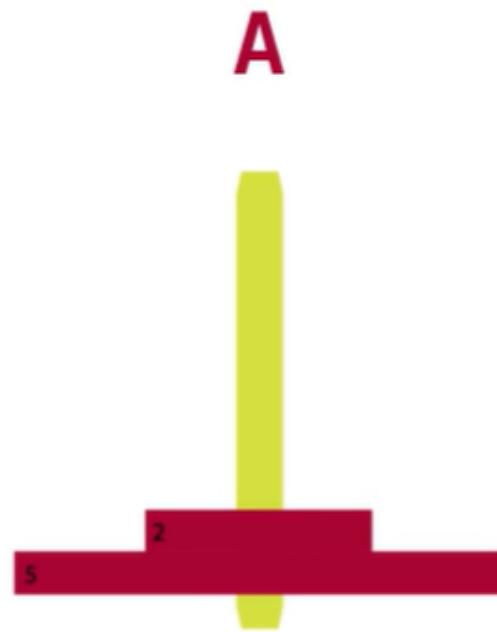
## Hanoi Tower Solution (2/7)



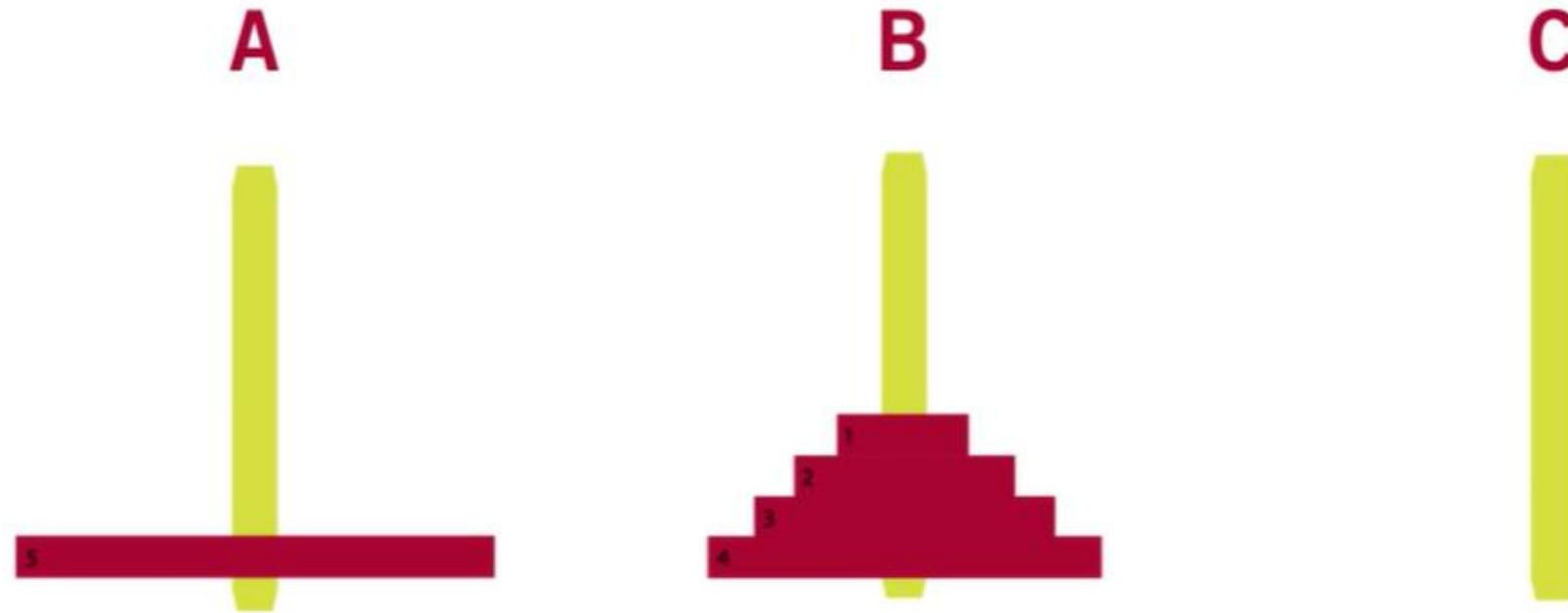
## Hanoi Tower Solution (3/7)



## Hanoi Tower Solution (4/7)



## Hanoi Tower Solution (5/7)

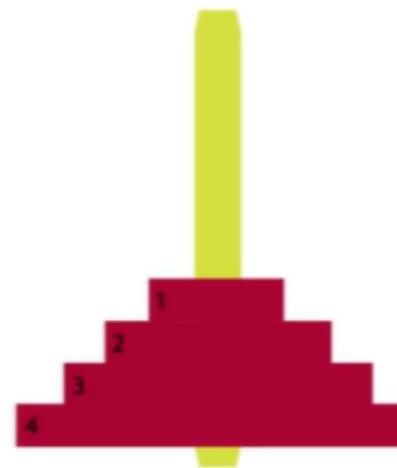


# Hanoi Tower Solution (6/7)

A



B



C



# Hanoi Tower Solution (7/7)

A



B



C





© MIT xPRO 2021. All rights reserved.