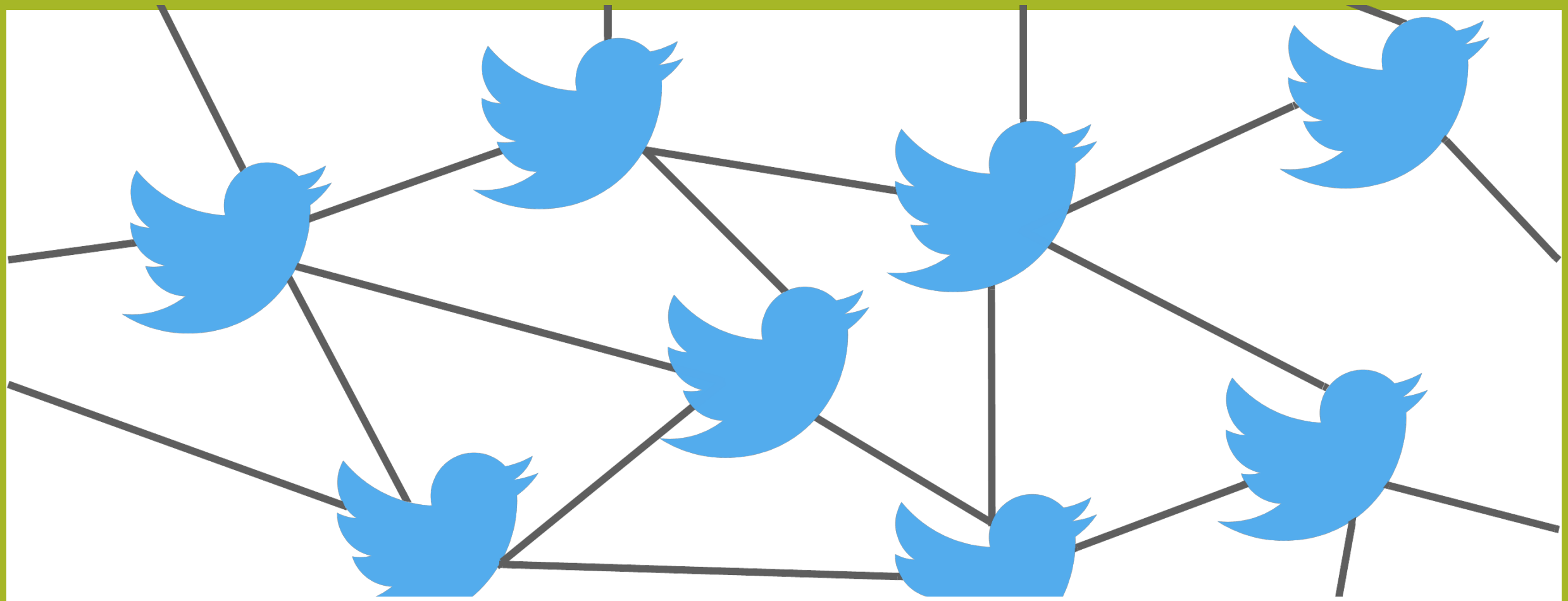


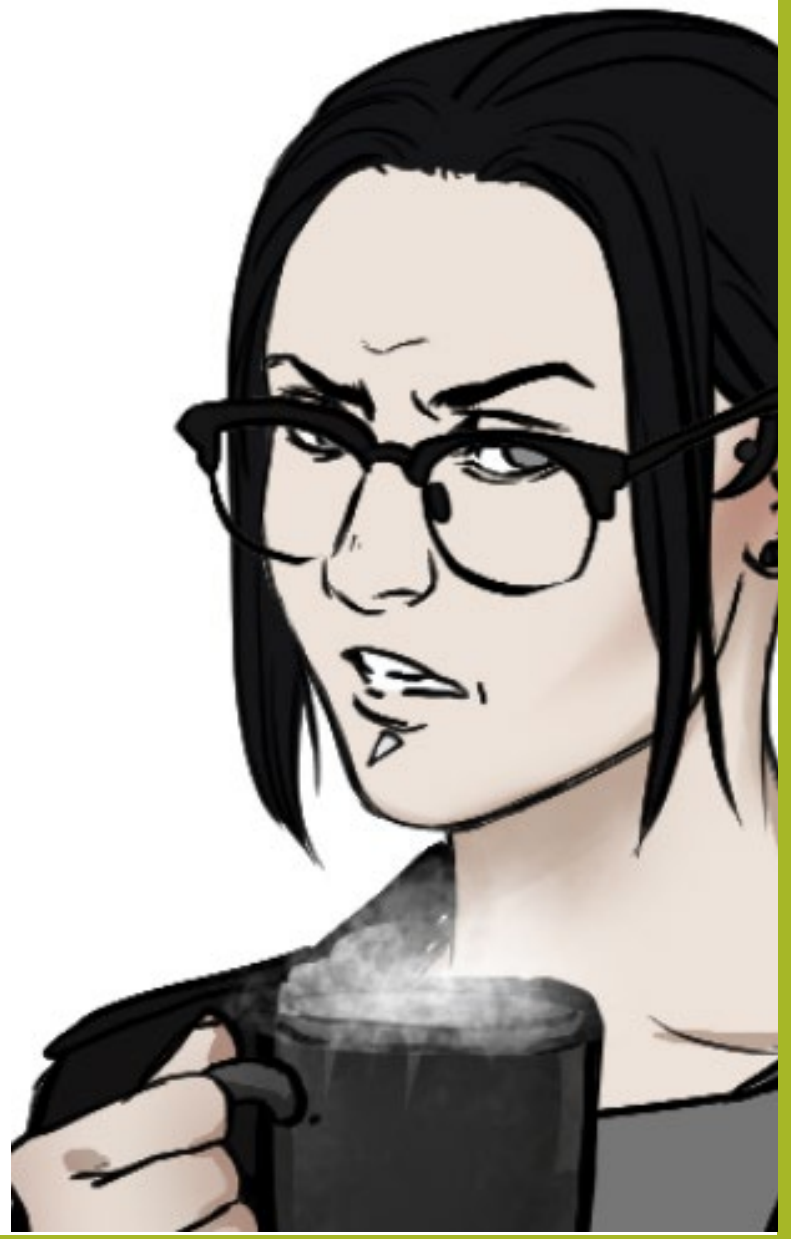
TWEETERS ANONYMOUS

Applying Mix Networks & Onion Encryption for Anonymity

Cassandra Young (@muteki_rtw)

@muteki_rtw

- Azure/O365 SysAdmin (with a focus on Security)
- Computer Science Grad Student @ Penn Engineering
  (aka glutton for punishment)
- Organizer, Blue Team Village @ Def Con
- Jack of All Trades but nerdy for scripting, Python & cloud security / serverless microservices
- Lives for international travel, scuba diving, powerlifting, woodworking, jigsaw puzzles and baking



Cassandra Young (aka muteki)

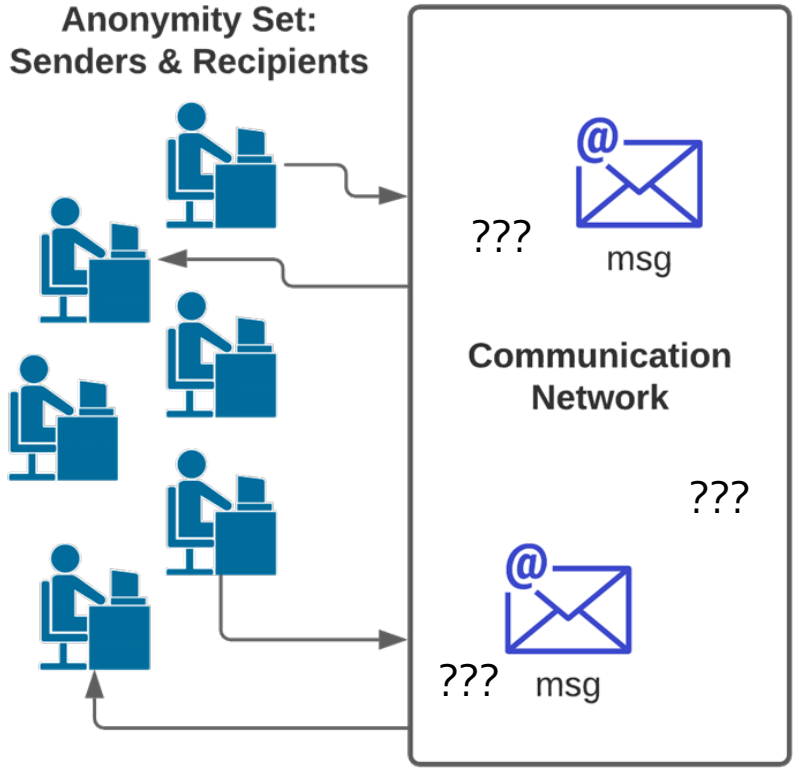
Pronouns: she/her
Twitter: @muteki_rtw
LinkedIn: linkedin.com/in/cassandray
GitHub: github.com/muteki-apps (go here for slides & talk links!)

Part One: Privacy and Anonymity

- What is Privacy?
  - Dictionary: "the state or condition of **being free from being observed** or disturbed by other people"
  - In tech: the **protection & obfuscation of personal data** relating to the individual's identity, past & present
- What is Anonymity?
  - A person/entity is anonymous when they **cannot be identified out of a set** of other users/entities

Part One: Privacy and Anonymity

- **Key Terms**
  - **Anonymity Set:** Any group of users/entities using a service
  - **Sender Anonymity:** The anonymity of one sender within the set of all senders
  - **Recipient Anonymity:** The anonymity of one recipient within the set of all recipients
  - **Unlinkability:** A state in which an attacker cannot reasonably determine (ie with statistical likelihood) if two items of interest are related.
  - **Global vs. Individual Anonymity:**
    - Global Anonymity: anonymity provided by the system to all its users together – stronger when all users in the system are statistically equally likely to relate to an item of interest.



Why Anonymity?

- Online status indicators can be used to analyze patterns of behavior
  - InfoSec use case: patterns of behavior can be combined with social engineering for an attack
- Your data is being sold whether you see ads or not
  - 32% of paid apps shared the exact same sensitive data with 3rd parties as free versions
- Law enforcement investigation may involve comparison of unencrypted data sets that expose data of untargeted users
- IoT devices such as smart speakers may accidentally capture private conversations when misactivated
- With GPS location data, one study found a home-finding algorithm correctly identified plausible home locations of 85% of the drivers tracked
- & many, many more unnerving statistics…

A Privacy-Focused Systematic Analysis of Online Status Indicators: https://camillec.com/PETS_OSIs.pdf
The Price is (Not) Right: https://www.petsymposium.org/2020/files/papers/issue3/popets-2020-0050.pdf
Open, Privacy-Preserving Protocols for Lawful Surveillance: https://arxiv.org/pdf/1607.03659.pdf
When Speakers are All Ears: https://www.petsymposium.org/2020/files/papers/issue4/popets-2020-0070.pdf
A Survey of Computational Location Privacy: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/12/computational-location-privacy-preprint.pdf


THIS IS FINE.

How does this relate to InfoSec?

- TL;DR: Anonymity is not just for criminals and hackers!
- The use of privacy-preserving technology should not be the sole basis for suspicion
  - Use of a VPN can render location-based alerts useless
  - Many VPNs and other tools piggyback off Tor

Privacy, Anonymity.. & Security?

- Privacy in the context of Security:
  - Protecting identity; A user's right to control access to their own personal data and how it is used, within contractual agreements
  - Regulatory acts such as HIPAA provide **standards for the lawful use and disclosure of personal data** (PII, PHI), and outline frameworks for securing it.
- Security:
  - Protecting data
  - Securing users' private data against **unintended disclosure** such as a data breach
- Privacy + Security = better protection of users, and a smaller blast radius

Anonymity is for everyone!

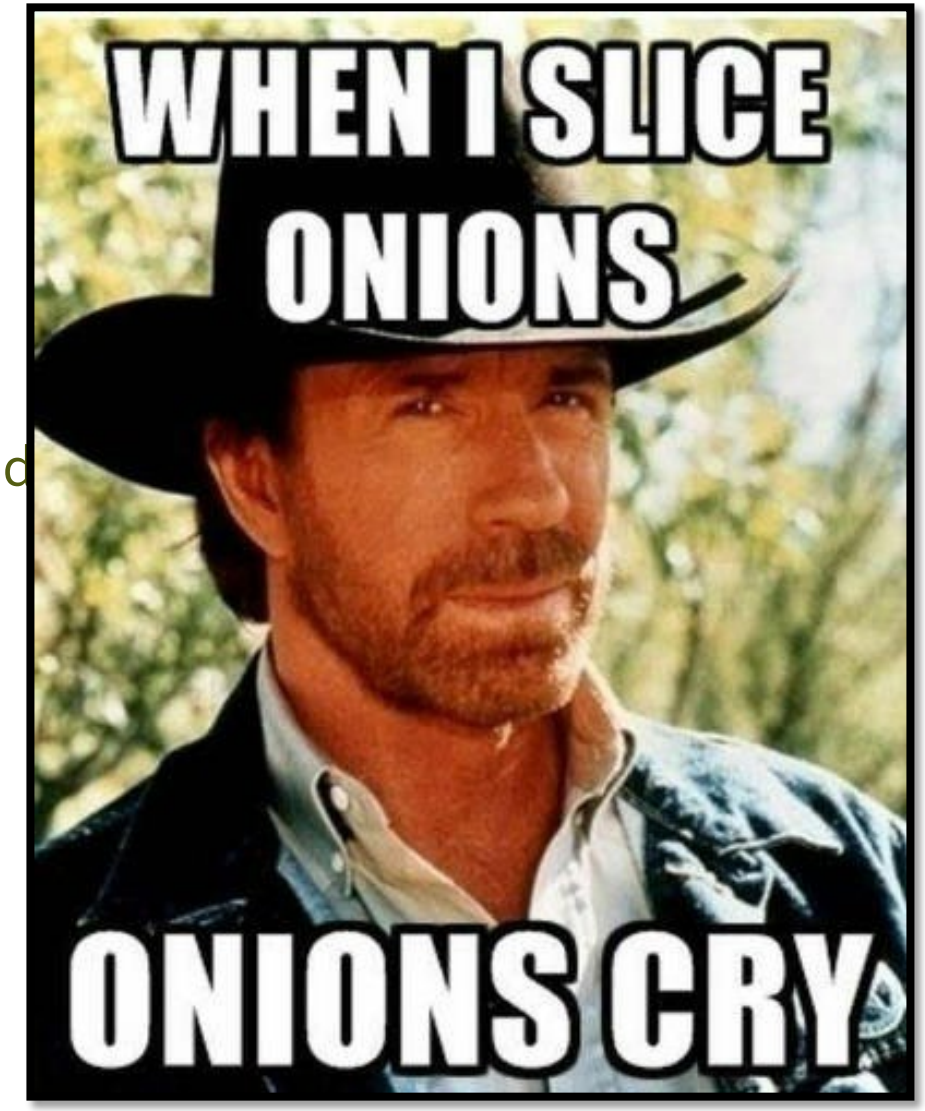
A Brief Survey of Anonymity Tech

- Commonly used tools:
  - Incognito mode, browser plugins, etc.
  - Secure messaging apps (ie Signal)
  - Pseudonyms, temporary email addresses, etc.
- The more involved side:
  - VPNs, proxy servers
- Down the rabbit hole:
  - Many tools are based on mix networks/onion encryption
  - Tor & Tor-based tools (ie Tails OS)

MIXES & ONION ENCRYPTION

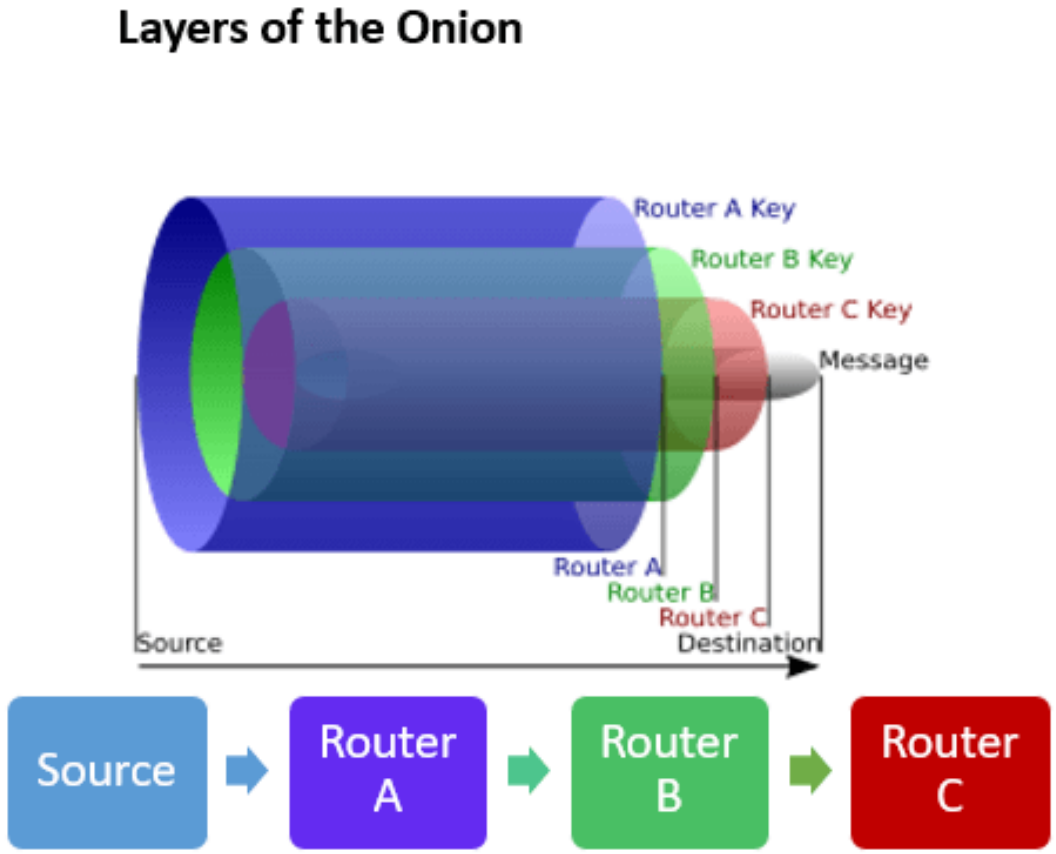
Mix Networks: The Chaum Paper

- 1981 Paper: "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms" by David Chaum
- Proposed mix network with "onion encryption" method to encrypt messages in layers
- Tor is an implementation based on this idea



Packaged Messages: Onion Encryption

- Core concept of Mix Networks, Onion Routing… and Tor
- Encryption in layers, like an onion ;)
- Different keys for each hop
- Messages sent along a route (‘circuit’ in Tor)



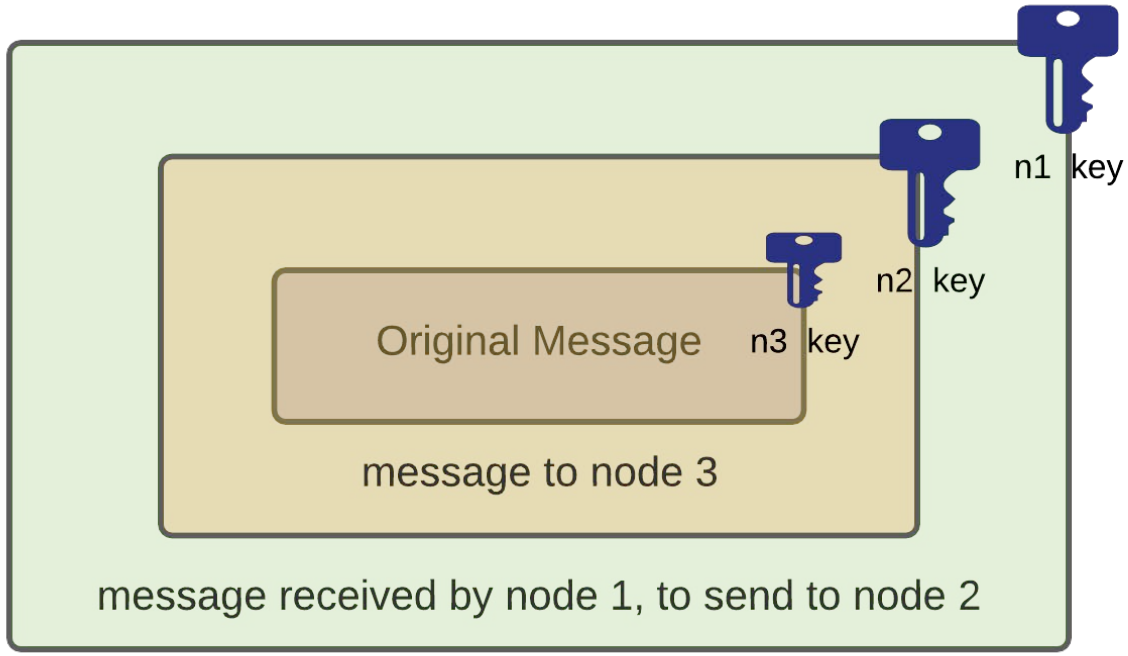
1) receives all public keys from central source
2) encrypts in layers using public keys
3) sends fully layered message to Node 1

1) receives encrypted message from client
2) decrypts w/ public key
3) decrypted message contains an encrypted message & instructions to forward to Node 2

1) receives encrypted message from client
2) decrypts w/ public key
3) decrypted message contains an encrypted message & instructions to forward to Node 3

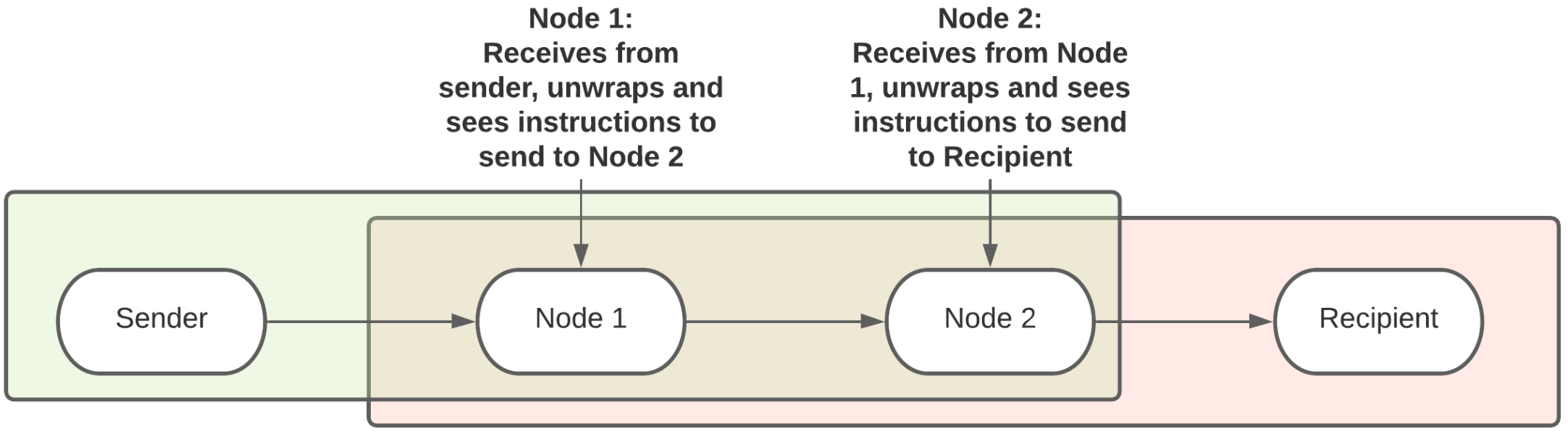
1) receives encrypted message from client
2) decrypts w/ public key
3) reads decrypted message, which is now cleartext since Node 3 is the final node

User Client → **Node 1** → **Node 2** → **Node 1**



Onion Encryption in Action

- Each node can only see:
  - The node/client it received the message from
  - The next node or final destination of the message
  - …and not both!




TWEETERS
ANONYMOUS
Implementing Mixes & Onion Encryption
with Python & AWS

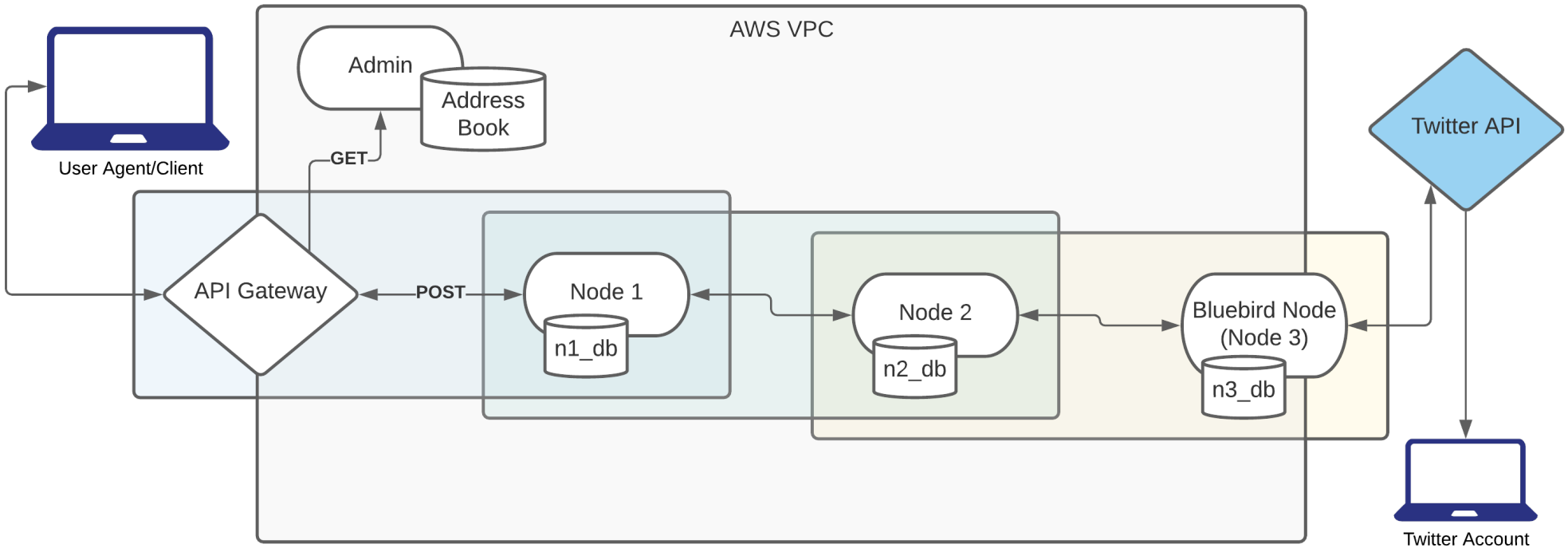
What does it do?

1. User submits a Tweet via a Python script
2. Script pulls necessary public keys from API and **encrypts the message in layers**
3. Message is submitted to API Gateway, and is **passed through Lambda nodes**
4. Each **node decrypts & reads** message, which includes 'action': forward or publish
5. **Final node publishes** Tweet to the "Tweeters Anonymous" account via Twitter API
6. (TBD) Final node sends URL via anonymous 'return address'

Tech Stack

- Python (Onion Encryption, UI & “server” side code)
- AWS:
  - API Gateway – interface for UI
  - Lambda - nodes and other functions
  - DyanamoDB – config & batch storage for nodes, address book
  - EC2 ? - Might add for initialization/management
- Twitter API
- Terraform – for infrastructure creation and code deployment

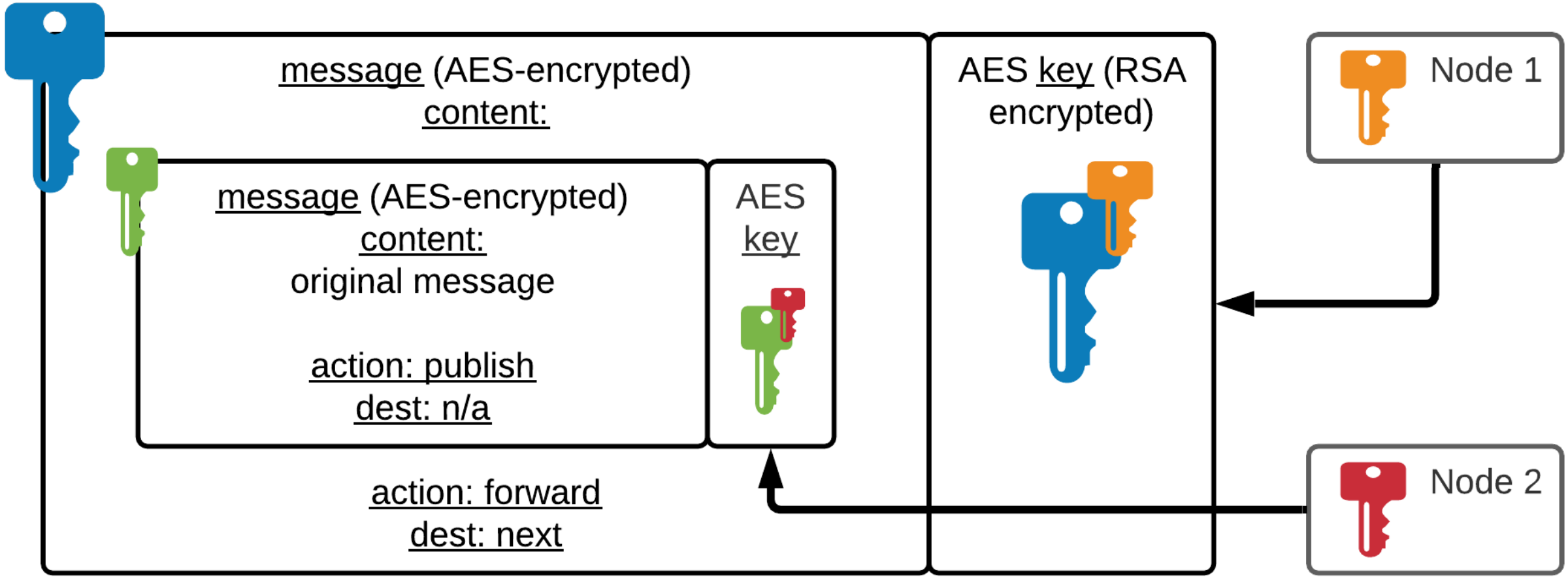
Tweeters Anonymous: Architecture



Onion Encryption in Python

- Challenge: Chaum's paper specifies public key encryption, but there's a size issue…
  - RSA key sizes: 1,024 or 2,048 or 4,096 bits
  - Tweet length: 280 char * 8 bits = 2,240 bits
  - And each layer (of 3) requires a larger key!
  - … clearly, that math doesn't add up!
- Solution: Hybrid encryption!
  - Encrypt message with one-time-use AES256 key
  - Encrypt AES key with RSA key
  - Wrap both in next layer of encryption w/ next AES256 key + RSA-encrypted AES256 key

Hybrid Encryption



1) package the message in JSON, 2) add padding, 3) encrypt with AES

```
package = json.dumps({
    "content" : msg,
    "action" : action,
    "dest" : dest
})

package = add_padding(package)

# get aes key and encryptor
aes_key, encryptor = generate_aes_key()

# encode in bytes and encrypt w/ AES key
enc_pkg = encryptor.update(package.encode()) + encryptor.finalize()
```

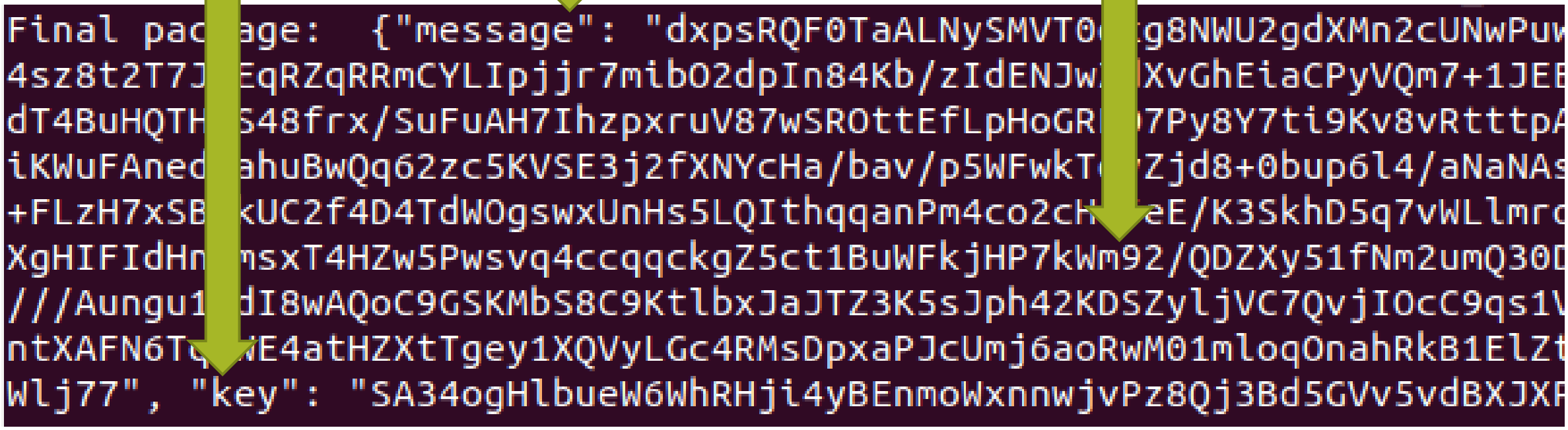
4a) wrap that AES-encrypted message + key in JSON, 4b) RSA encrypt the ARS key w/ padding, 4c) fix encoding shenanigans, 5) serialize JSON as string for next layer

```
# package 1) encrypted message as base64-encoded string, and 2) aes key, in json
# conversion from byte arr to base64 encoded str is necessary to make it JSON serializable
wrapped = {
    "message": base64.b64encode(enc_pkg).decode(),
    "key": base64.b64encode(
            rsa_key.encrypt(
            aes_key,
            padding.OAEP(
                padding.MGF1(algorithm=hashes.SHA256()), hashes.SHA256(), label=None
            ),
        )
    ).decode()
}
# return serialized json version of package (string object)
return json.dumps(wrapped)
```

Example of final package that will be sent through API Gateway to Node #1:

JSON-formatted string contains “message“
…and “key”

“message” is nested packages, all encrypted



Unwrapping: Extract & decrypt the AES key

```
Final package:  {"message":
4sz8t2T7JhEqRZqRRmCYLIpjjr7m
dT4BuHQTHcS48frx/SuFuAH7Ihzp
iKWuFAnedvahuBwQq62zc5KVSE3j
+FLzH7xSBxkUC2f4D4TdWOgswxUn
XgHIFIdHnomsxT4HZw5Pwsvq4ccq
///Aungu1rdI8wAQoC9GSKMbS8C9
ntXAFN6Tq+WE4atHZXtTgey1XQVy
Wlj77", "key": "SA34ogHlbueW
```

```python
unwrapped = json.loads(msg) # load to JSON from string

# extract the AES key, which is wrapped w/ RSA key
key_pkg = base64.b64decode(unwrapped["key"])
key_parts = rsa_prv.decrypt(
    key_pkg,
    padding.OAEP(padding.MGF1(algorithm=hashes.SHA256()), hashes.SHA256(), label=None),
)
aes_key, aes_iv = extract_aes_key(key_parts)
cipher = Cipher(algorithms.AES(aes_key), modes.CBC(aes_iv), backend=backend)
decryptor = cipher.decryptor()
```

Example of Decryption in Progress:

```
Decrypted message:  {'content': '{"message": "S0YGeo9bR/ci2t9Yom+0rOaa+78bDS69qdylVDv6Ll
jTE5ZCZ2os9GYn6/pqeoZHDGU8eNX1/K9ydjBPKJCYHcrITFF6m3Pftqc62I1uqLvYmq4szIoYCnBT9+yPBHoObY
8ivpncOKe7IamFAyVMHOkR9MvwCH9JB60a5O2yA3TO9nD5IuBAHQrD8bdtd4AEESP2i7KrUBs5pi8hXDVlFCrQyL
ACHNqv4GKZbgh/rk4oHc8hN9ChZXnxgo/F0fuu1b0+kRgiAt29rpl5x0YJG/7XmTEf2HmxpEwxySynO7IL9tzpeO
78kB6vP08Xf7XKRDpGIKg5oNfcQ4wOI6GP4RmABMxEOJUOUVbbUORFQFREUorMgIeQQHLGGLqoqrnS3xIfd9h/fQ
tU3W+RA1vF9LVsVhSQCX3OinbRavCH0BK3VSZSkAU0YPAsVSzjcoGFiMVQyVte/tTXIFERLnmCRQ==", "key":
"ZTViUx5iRkBmyU3qe3mpUdIuoLP2S3marTOUOu9FHdz5w+eCVK7BcGAboZsqbGYN2aXmf2ZTC74vgGV8I1u2fEC
J1ZupVNW0L8Mp410eVY8eVJK/cCekzEajMFSFfjn45V4rLvLYUldoM6WGi2/mRBvLieUb3q6Jn5P4ppXPcJk="}'
, 'action': 'forward', 'dest': '3'}
Unwrapped2:  <class 'dict'>
Forwarding to:  3

Decrypted message:  {'content': "This is a long Tweet. It will by the end become the lon
gest Tweet I'm allowed to Tweet, and now I am going to wrap it up real tight and send it
 off to bed. It'll dream of sheep or some shit, but who cares, this was just a test for
onion encryption anyways. Is it too long to work?", 'action': 'publish', 'dest': 'n/a'}
Unwrapped3:  <class 'dict'>
Publishing!!!
```

Infrastructure & Automation: Terraform

- Terraform "allows infrastructure to be expressed as code"
- Consistency is security, so why not code the blueprint?
- Use cases:
  - Destroy and recreate node instantly in case of compromise
  - Recreate entire infrastructure regularly
  - Can create configuration that allows for scalability and custom configs
- End goal: deploy code that anyone can stand up themselves

```
provider "aws" {
  region = "us-east-1"
}

resource "aws_vpc" "tweeters_vpc" {
  cidr_block = "10.0.0.0/16"
  tags = {
      Name : "tweeters_anon"
  }
}
```

Example: Defining a Lambda

```
resource "aws_lambda_function" "ta_lambda_entry_node" {
  function_name = var.lambda_names[0]
  description   = "Test Python Lambda Function that does god knows what..."
  filename      = "node.zip" # zip file stored in same dir, or can be pulled from S3
  runtime       = "python3.6"
  handler       = "node.lambda_handler"  # handler is 'main' for script

  role = aws_iam_role.ta_node_one_role.arn

  tags = {
      Name = "tweeters_anon"
  }
```

Example: Role for the Lambda

```
# IAM role for Lambda, with two policies
resource "aws_iam_role" "ta_node_one_role" {
  name = "ta_node_one_role"
  assume_role_policy = <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Action": "sts:AssumeRole",
        "Principal": {
            "Service": "lambda.amazonaws.com"
        },
        "Effect": "Allow",
        "Sid": ""
        }
    ]
}
EOF

   inline_policy {
        name = "dydb_policy"
        policy = data.aws_iam_policy_document.lambda_policy_dydb.json
   }
```

- Roles can be changed and applied as the program is built
- Can write in plain text, import from file, or define through Terraform's config language
- Key idea: consistency and control

Summary & Next Steps

- This project isn't done yet!
- Onion Encryption – applied theory to practice
- Anonymity is only as secure as the system itself
- Terraform allows for thoughtful, design-focused implementation
- Build process is both:
  - Top-down: designing infrastructure from high level to the nitty gritty
  - Bottom-up: Python code establishes low-level basics of onion encryption
- Goal: Finish by meeting in the middle for an end-to-end product!

ANONYMIZE
ALL THE THINGS

Cassandra Young
Pronouns: she/her
Twitter: @muteki_rtw