
AWS
re:Invent
NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

SEC404

# A day in the life of a billion requests

Eric Brandwine (he/him)

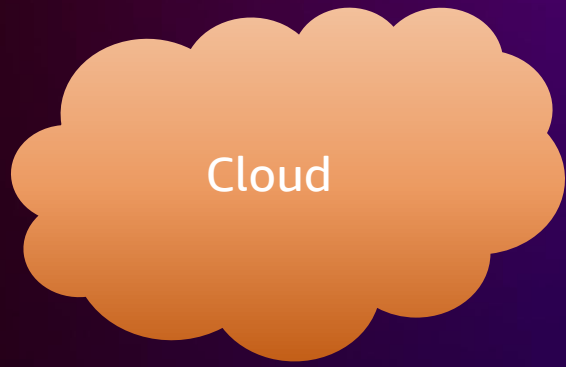
Vice President and Distinguished Engineer
Amazon Security

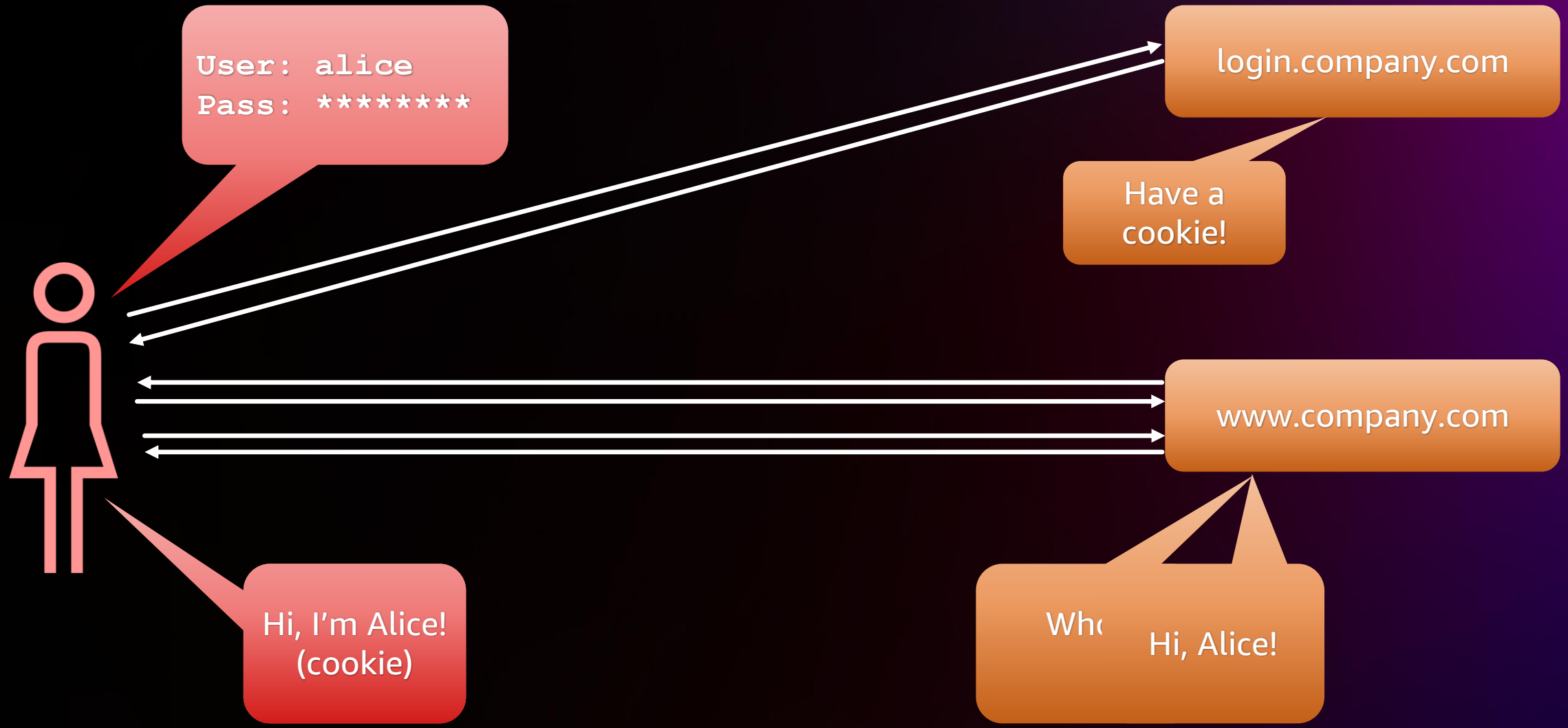
# Authentication and Authorization

# Cryptographic Protocol


https://xkcd.com/2691/

Alice





# TLS was expensive and not widespread

No iPhone or Android

Rickrolling hadn’t been invented

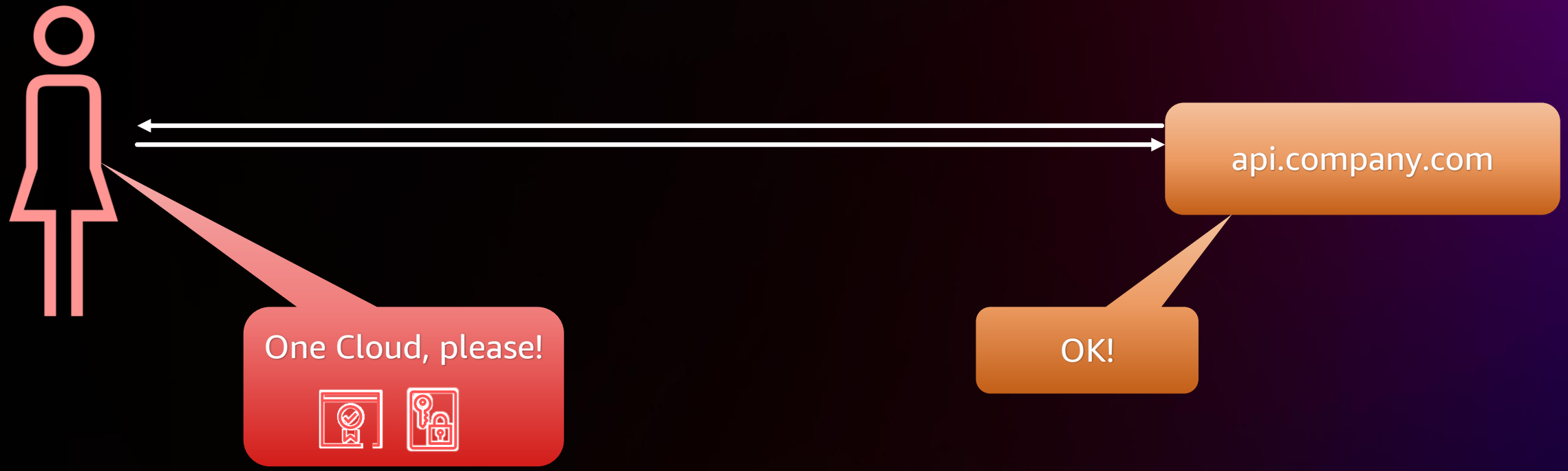
# 2006

Dinosaurs roamed the Earth

Netflix delivered via the postman

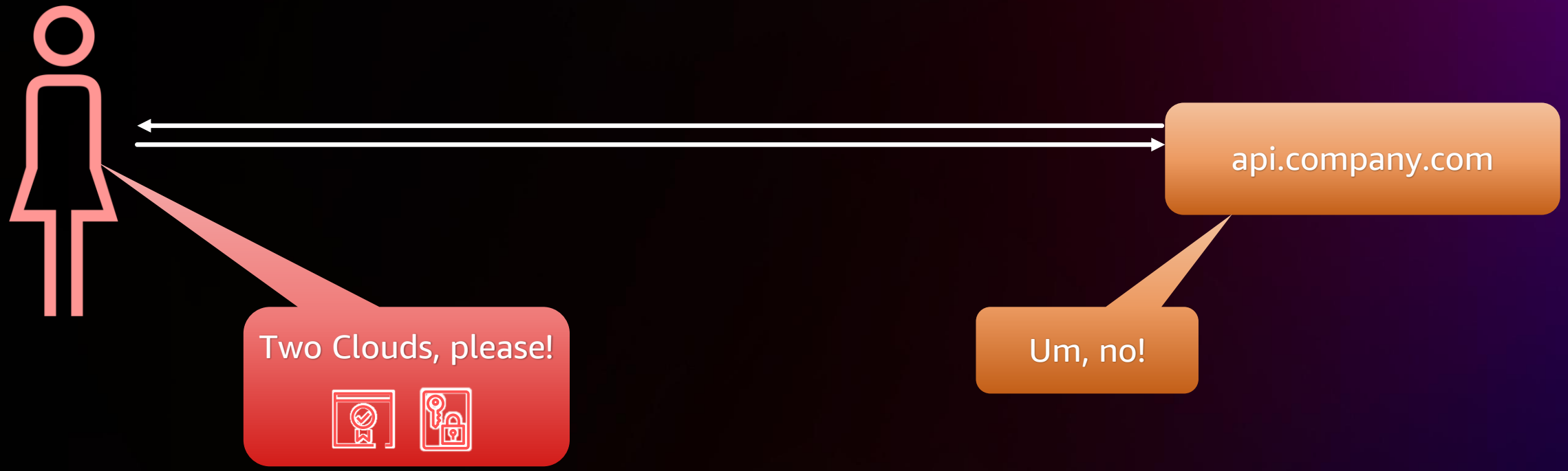
sign( , “One Cloud, please!”) →

verify(“One Cloud, please!”, , ) → Yup!



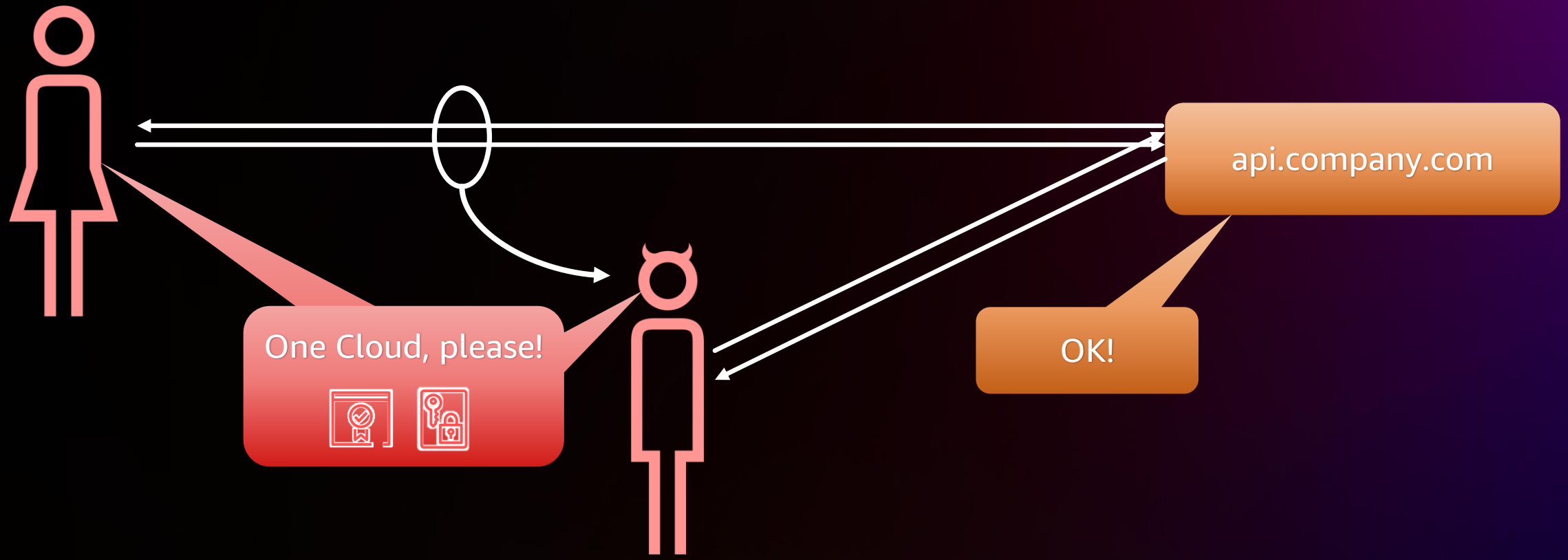
sign( , “One Cloud, please!”) →

verify(“Two Clouds, please!”, , ) → Nope!



sign( , “One Cloud, please!”) →

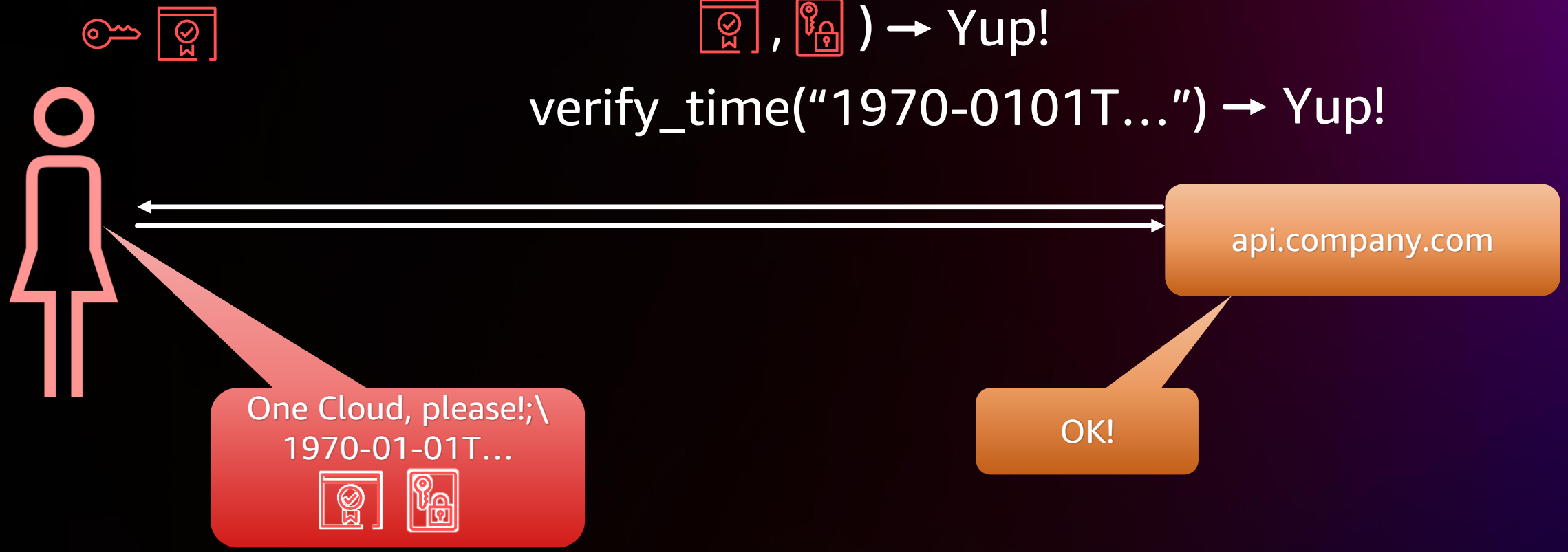
verify(“One Cloud, please!”, , ) → Yup!



sign( , “One Cloud, please!;1970-01-01T…”) → 

verify(“One Cloud, please!;1970-0101T…”,
 , ) → Yup!

verify_time(“1970-0101T…”) → Yup!



# TLS was expensive and not widespread

# 2006

# So, where are we?

- ✓ Every request is signed
- ✓ Requests can’t be tampered
- ✓ Requests can’t be replayed
- ✓ Protocol is stateless
- ✗ Crypto is expensive

# Meet the hash!

Cryptographic Hash Algorithm:

- Also known as a “digest”
- Maps arbitrary length input to fixed length output
- Has the avalanche effect
- Is “hard” to reverse

*Fast!*

# Bonus! We're already hashing!

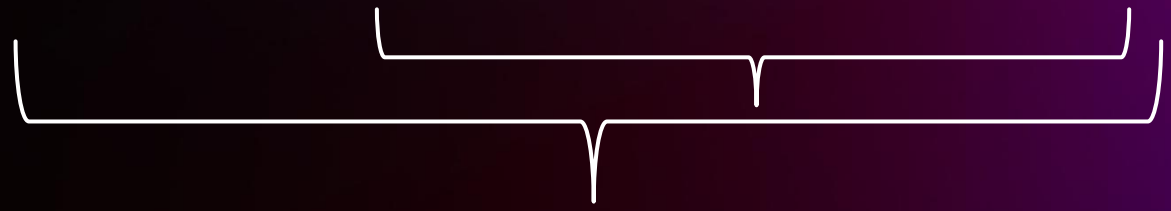
~~sign(🔑, "Really long message that keeps going…")~~ → 

sign(🔑, H("Really long message that keeps going…")) → 

# Hash-based Message Authentication Code

# Meet the HMAC!

HMAC(key, message) = H(key || H(key || message))



hmac-sha256( , “Shhh!  Don't tell!”) =
`362d571721e65959e4202253a2ff068c8dbeb974ba3c3776f8881a4d95fbfcf5`

hmac-sha256( , “Shhhh!  Don't tell!”) =
`e5248d8d5dfd0897c02f3e6088de46d0f4858973cf2cab67e6c08080a107db40`

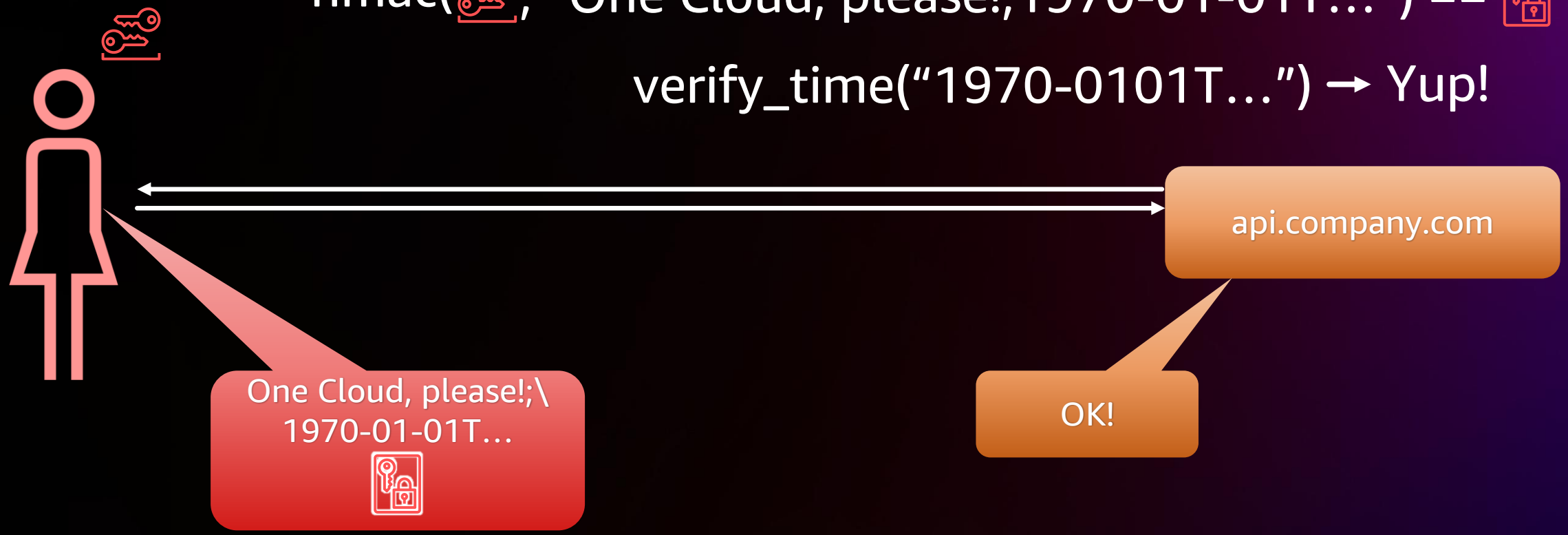
hmac-sha256( , “Shhhh!  Don't tell!”) =
`717d8051682eeb25566812a59f54448c192f26ca8fa3d60f9887b3ef7bce86f0`

hmac-sha256( , “Rhhhh!  Don't tell!”) =
`fdce6af8037a6b3465fd95204d772e09e020767ef5c2001531e1bc17c7679450`

hmac( , “One Cloud, please!;1970-01-01T…”) →

hmac( , “One Cloud, please!;1970-01-01T…”) ==

verify_time(“1970-0101T…”) → Yup!



# So, NOW where are we?

- ✓ Every request is signed
- ✓ Requests can’t be tampered
- ✓ Requests can’t be replayed
- ✓ Protocol is stateless
- ✓ Crypto is cheap
- ✓ Largely unaffected by quantum
- ✗ Keys are symmetric

signin

console

User: alice
Pass: ********

s3.us-east-1

Who?
Go sign in!

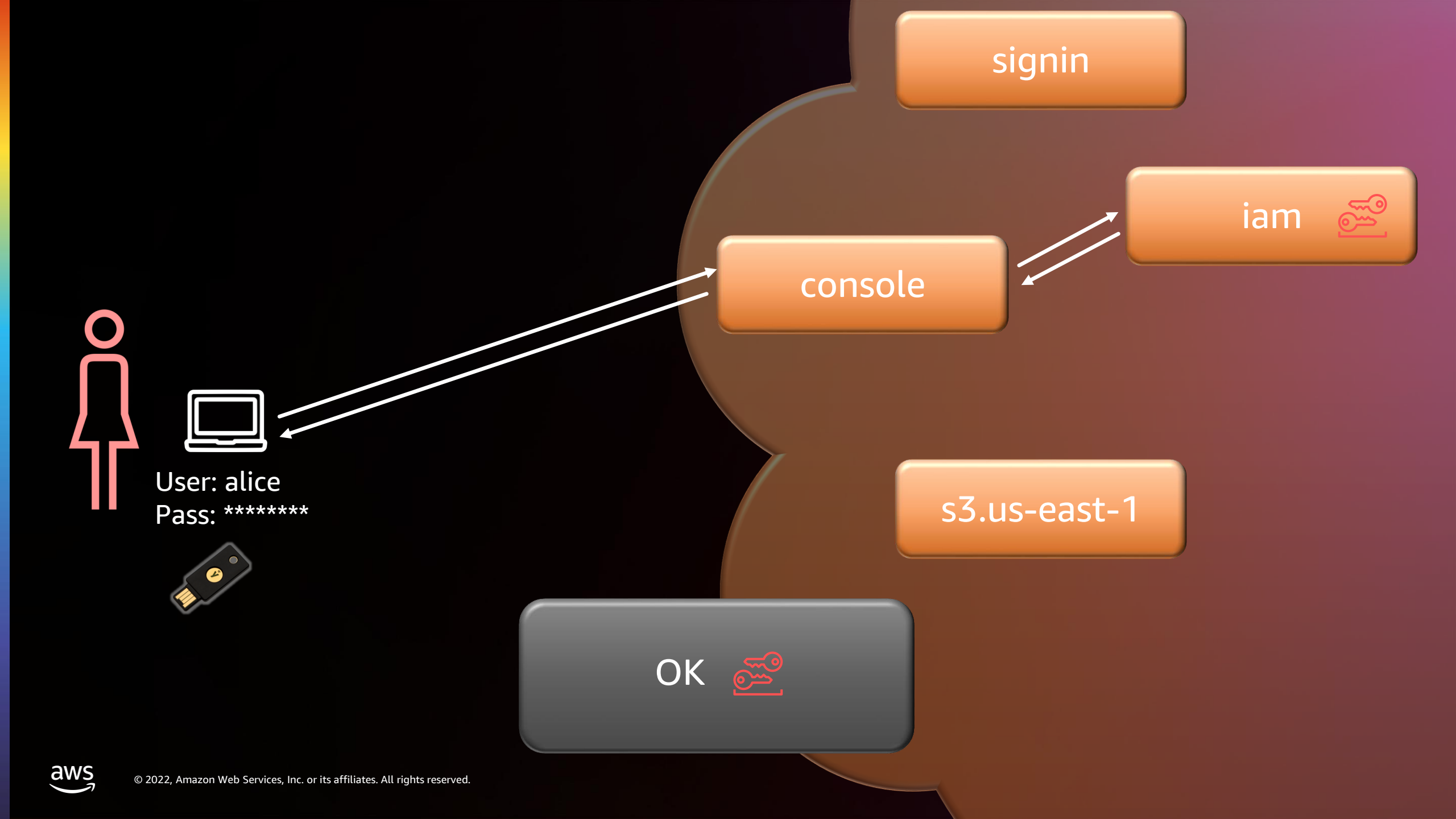
User: alice
Pass: ********

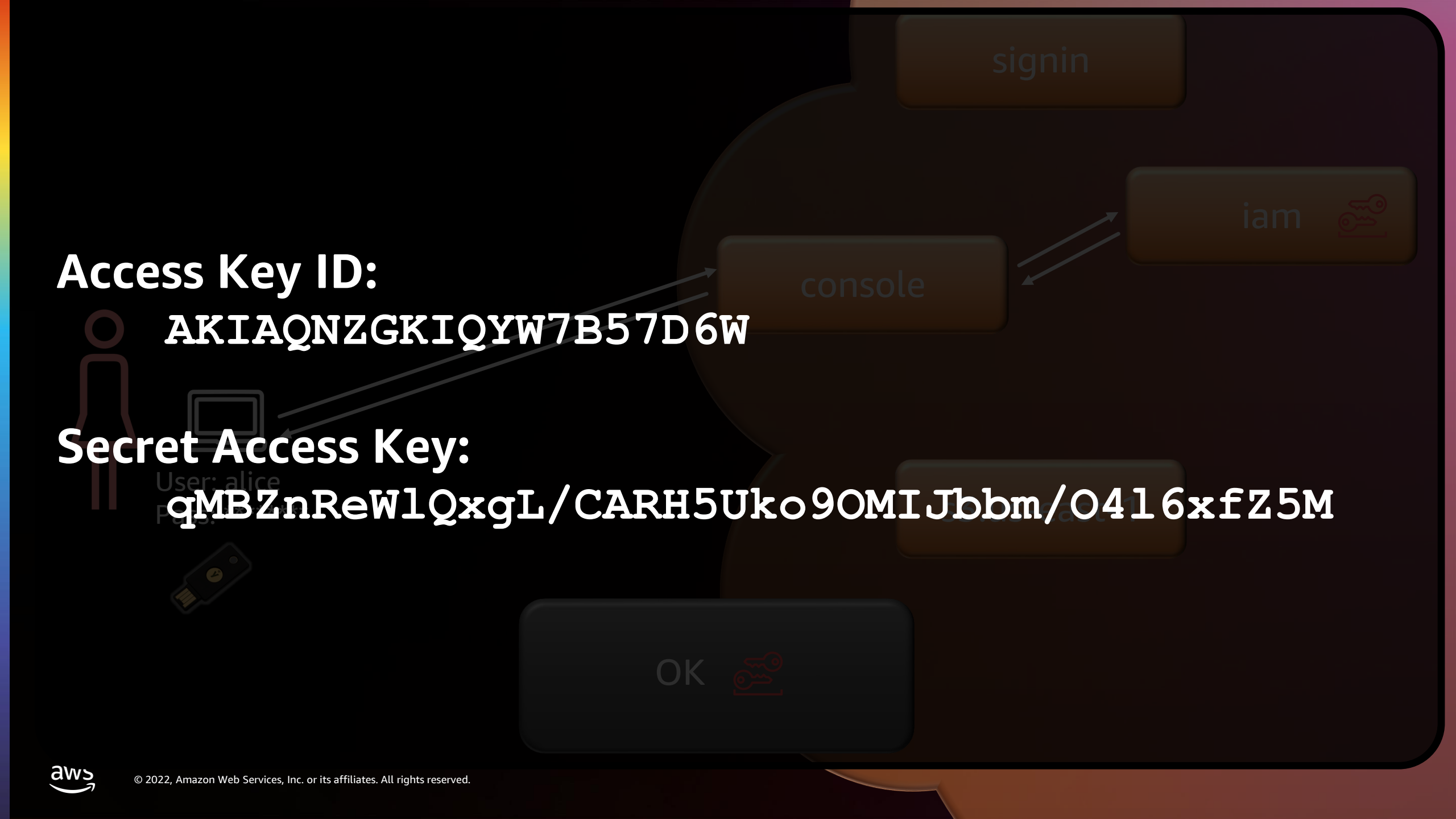
signin

console

s3.us-east-1

Have a cookie!



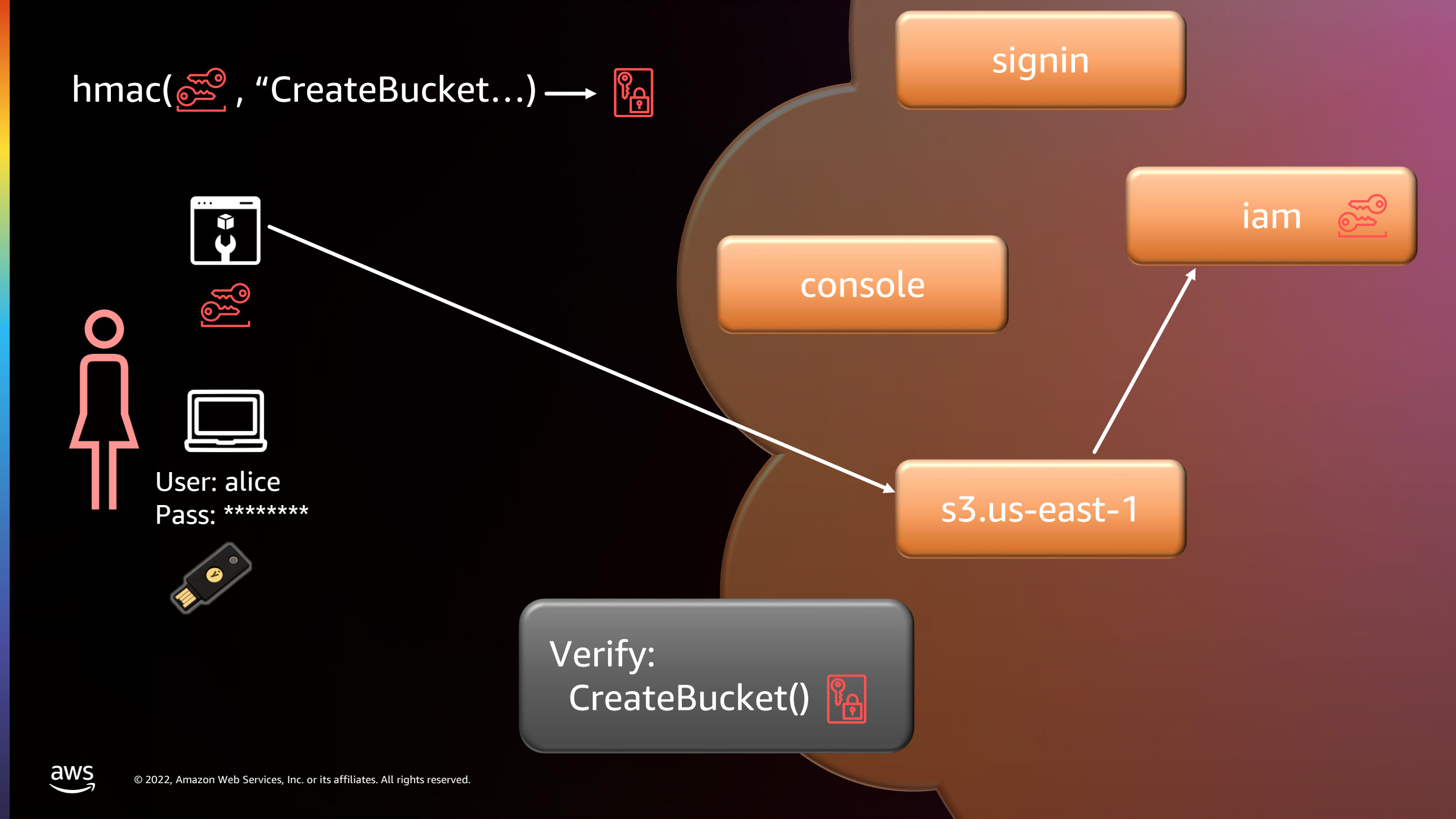



**Access Key ID:**

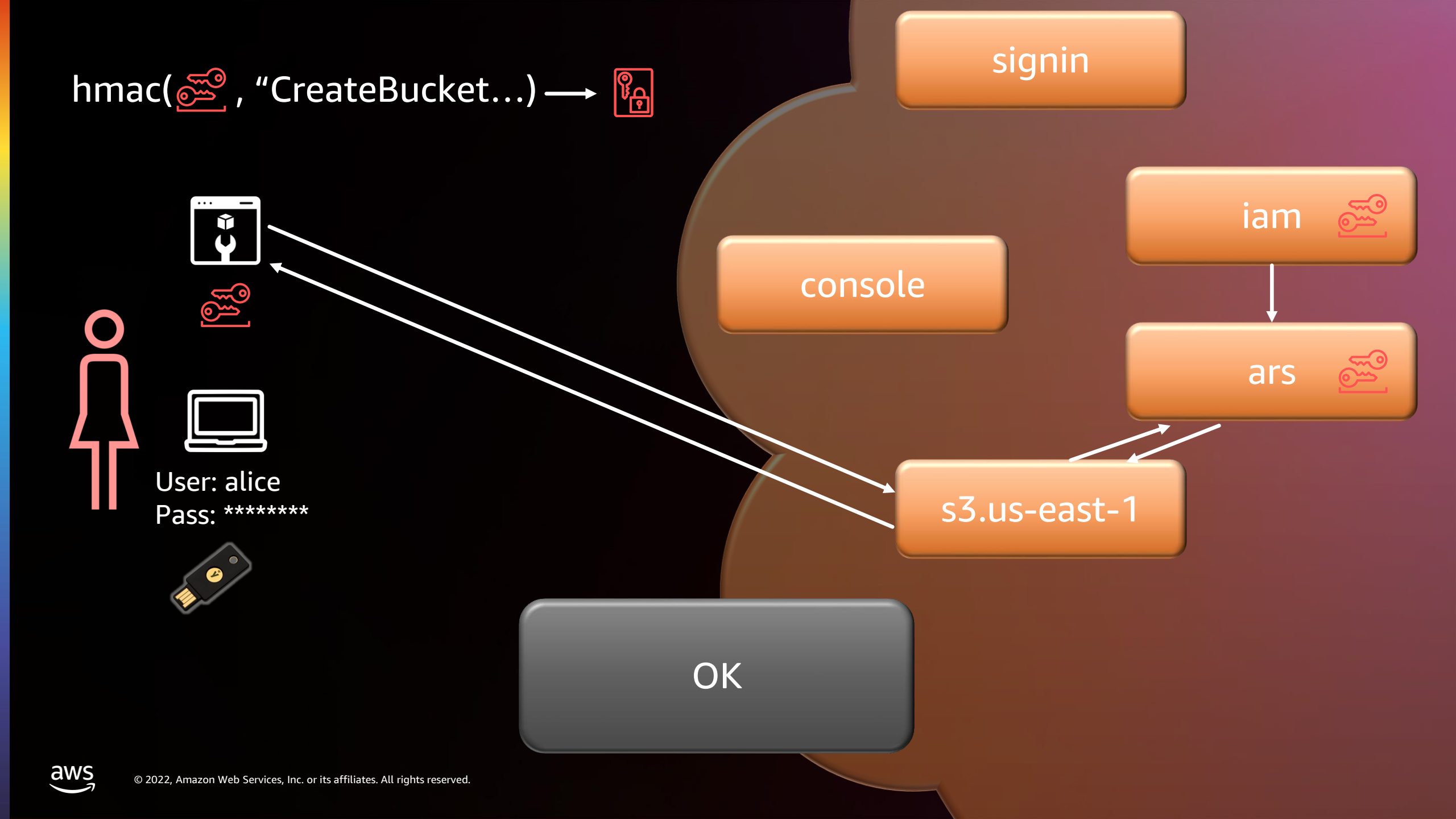
`AKIAQNZGKIQYW7B57D6W`

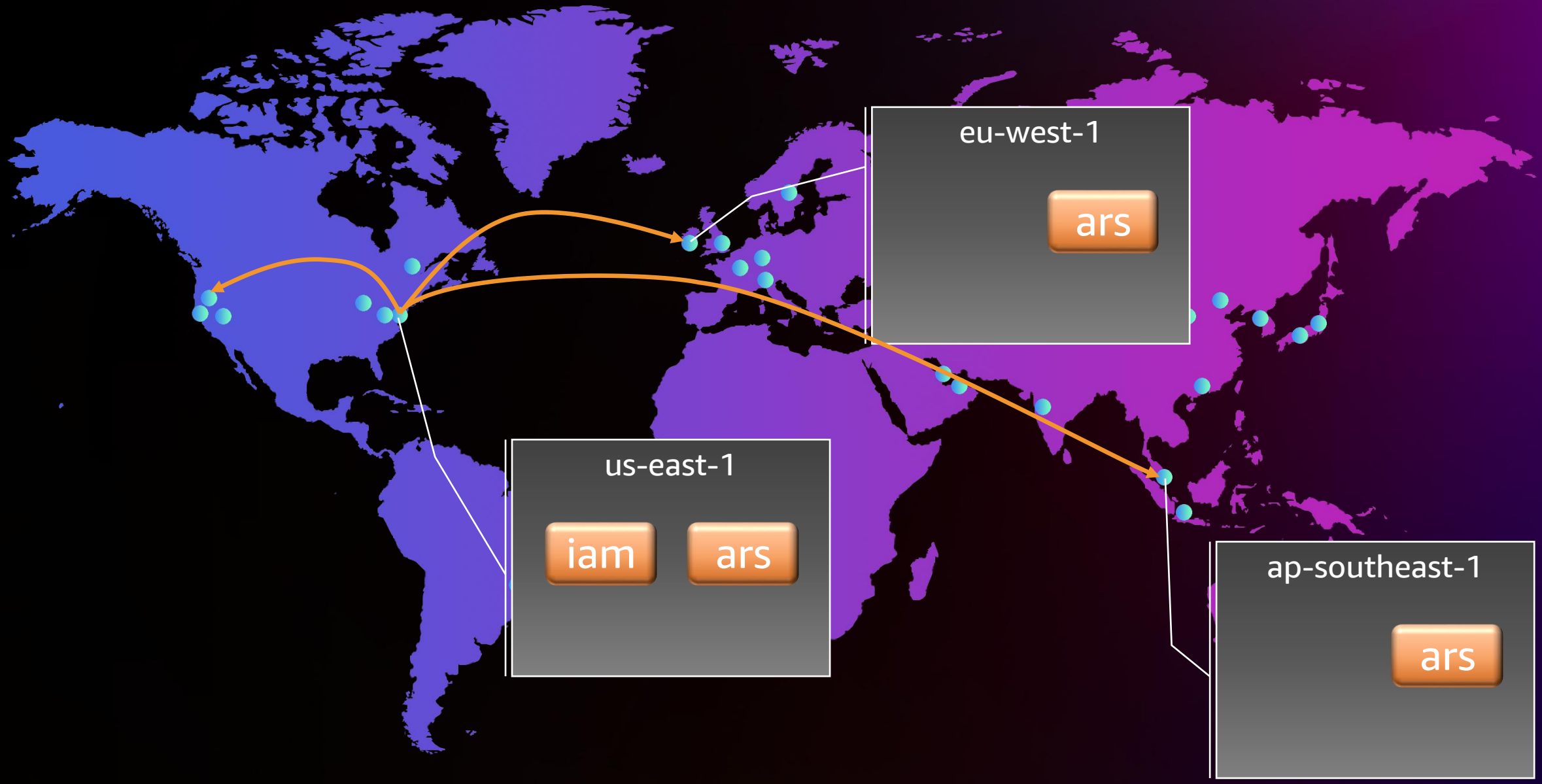
**Secret Access Key:**

`qMBZnReWlQxgL/CARH5Uko9OMIJbbm/O4l6xfZ5M`



# Auth Runtime Service





# AWS Signing History

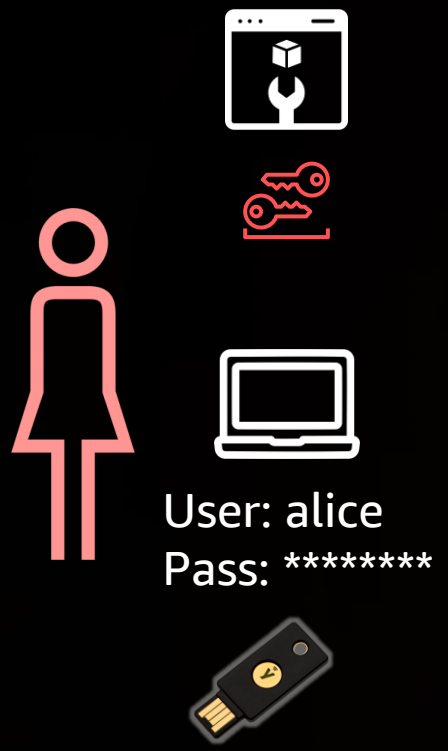
| | |
|---|---|
| AWS SigV0 | Largely internal |
| AWS SigV1 | Canonicalization issue 2008 |
| AWS SigV2 | No known issues |

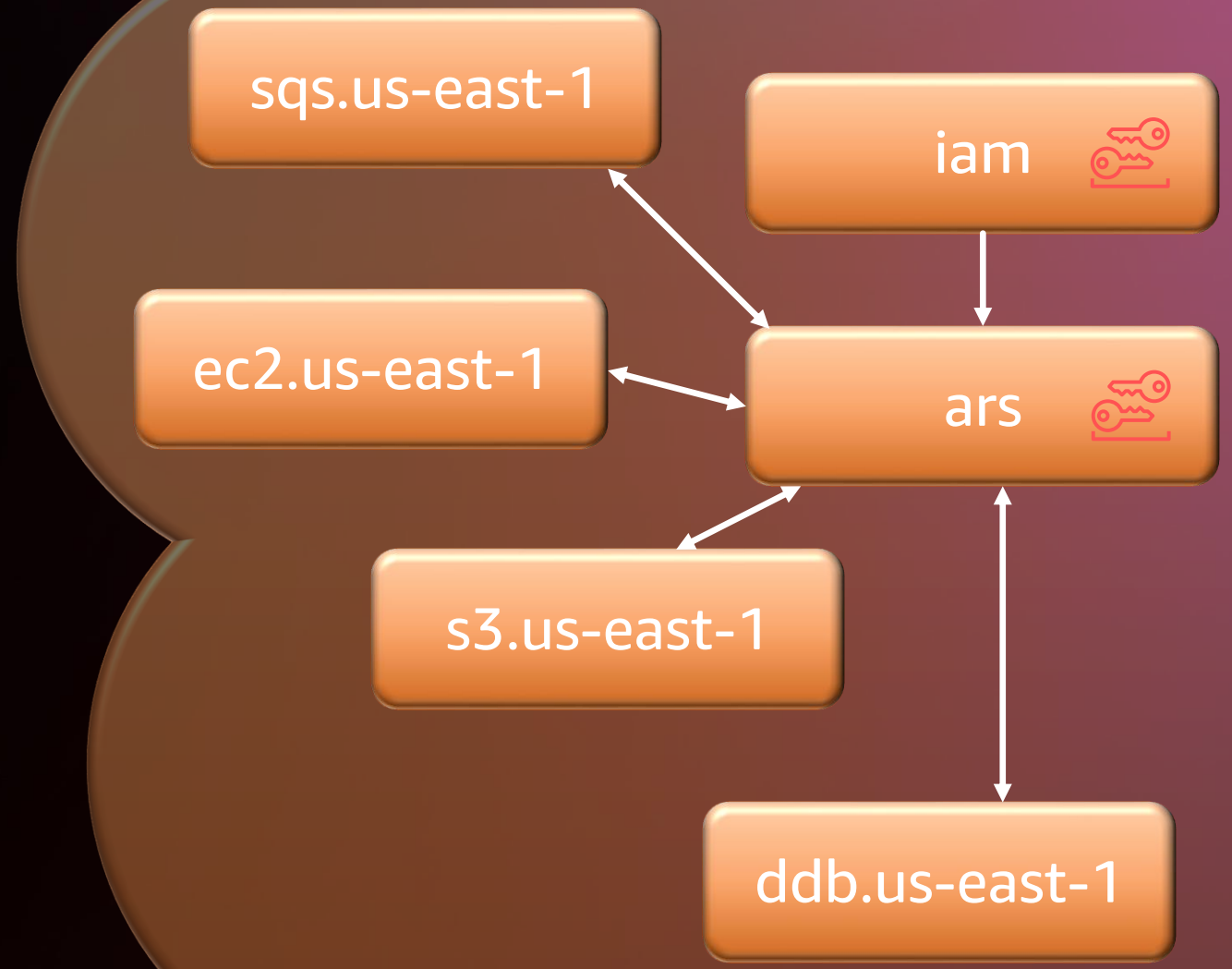
hmac(  , “One Cloud, please!;1970-01-01T…”) → 

Canonicalization

# AWS Signing History

| | |
|---|---|
| AWS SigV0 | Largely internal |
| AWS SigV1 | Canonicalization issue 2008 |
| AWS SigV2 | No known issues |



# So, now what?

- Literally millions of HMAC keys
- Asymmetric Cryptography is still slow
- HMAC is symmetric
- HMAC is still fast

# AWS Signature Version 4

# Hey! What happened to AWS SigV3?

What happened to IPv5?

# If one is good…

Long-term customer secret


HMAC("AWS4" ||  , Date)


HMAC( 

, Region)

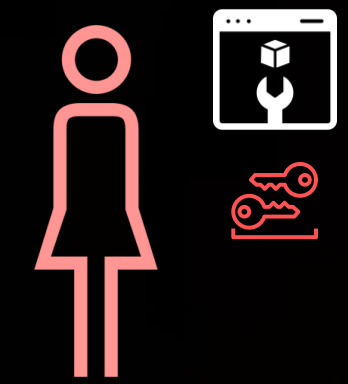

HMAC( 

, Service)

HMAC( 

, "aws4_request")


hmac( , “CreateBucket…”)



signin

console

iam

ars

s3.us-east-1

OK


hmac( , “CreateBucket…”)

signin

console

iam

ars

s3.us-east-1

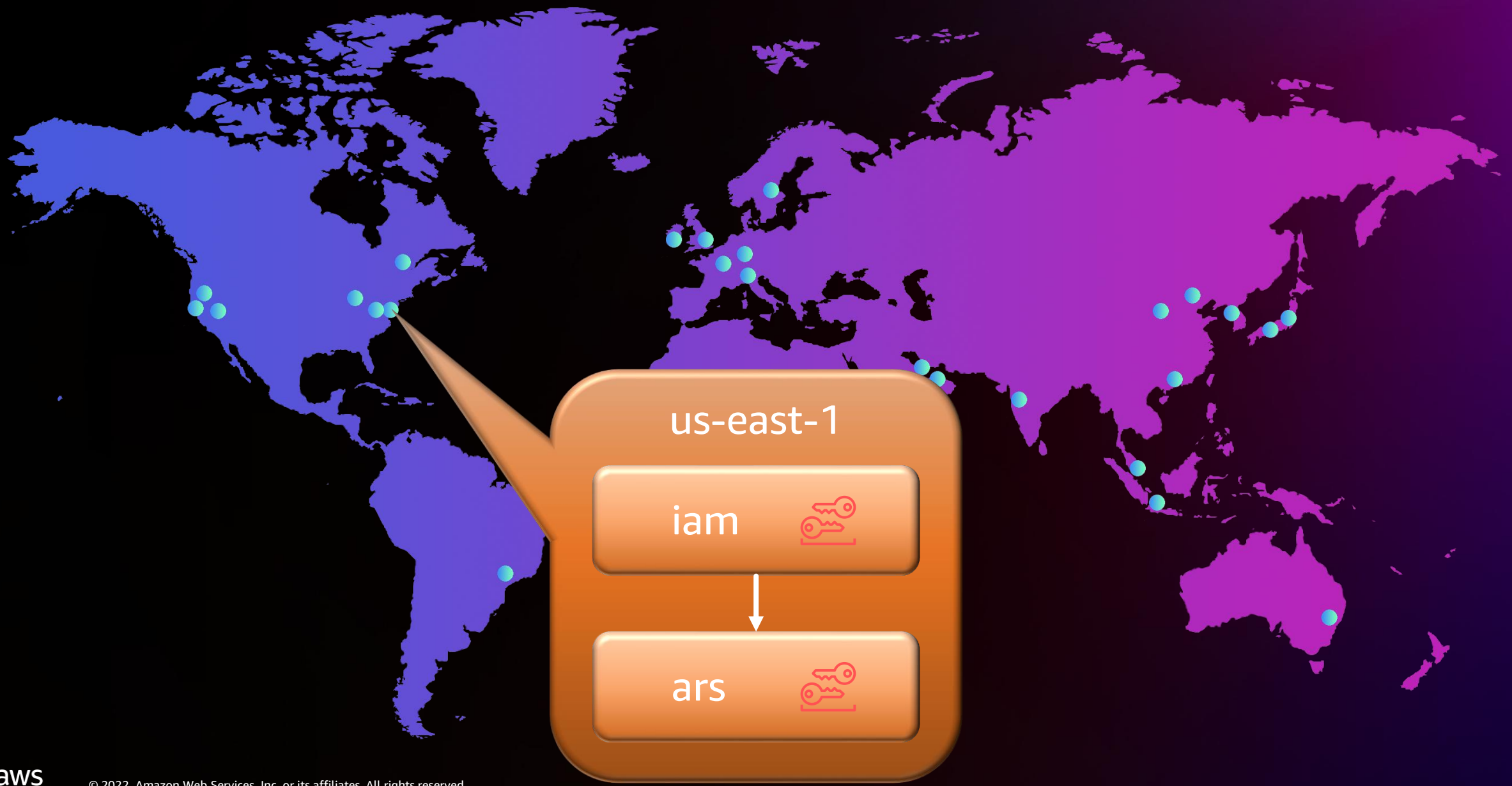
OK

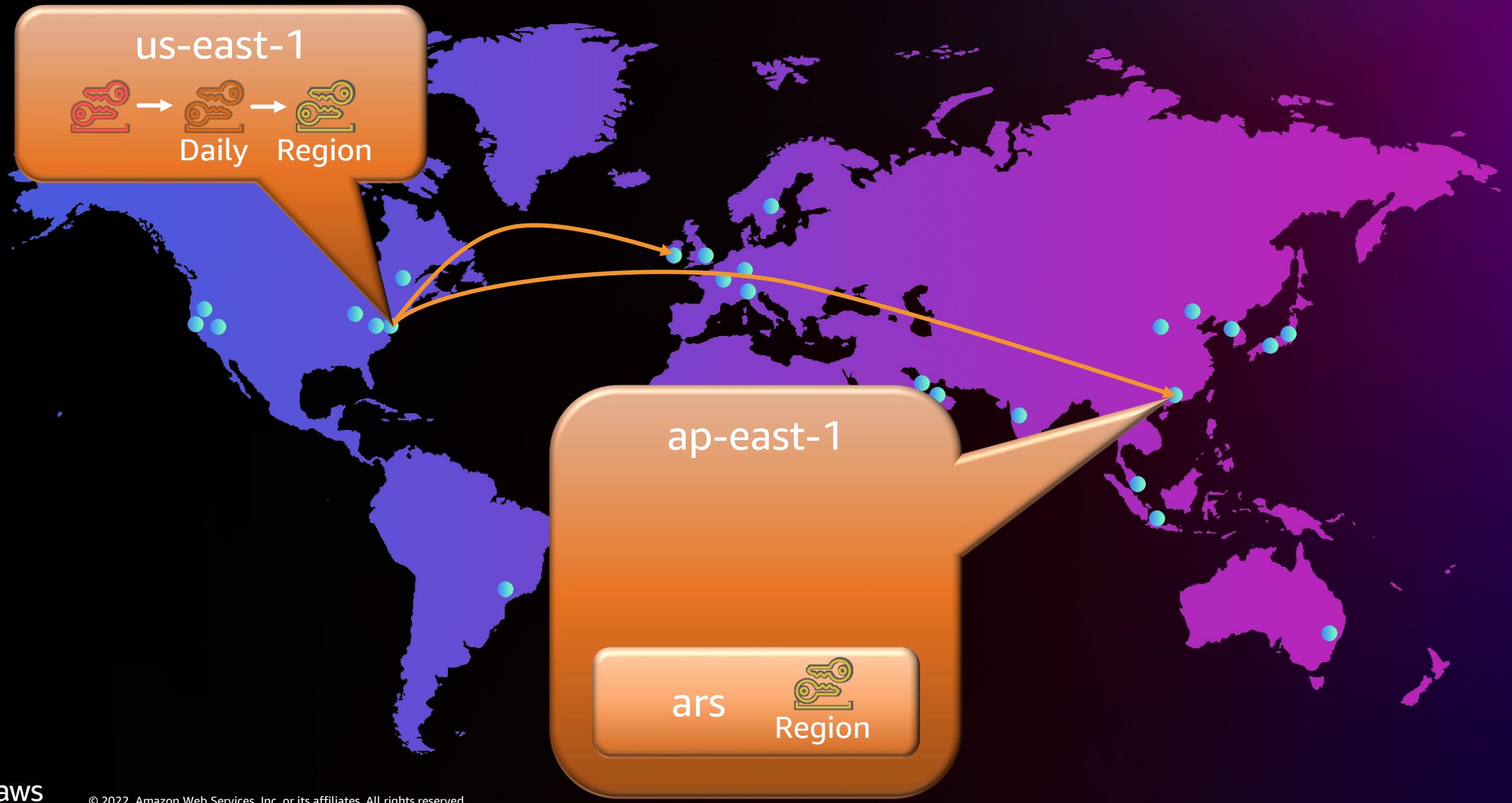
# AWS Signing History

| | |
|---|---|
| AWS SigV0 | Largely internal |
| AWS SigV1 | Canonicalization issue 2008 |
| AWS SigV2 | No known issues |
| AWS SigV3 | Why are you still asking? |
| AWS SigV4 | June 2012 |

# Other Milestones

- October 2014: AWS Germany (Frankfurt) Region launch
  - 100% SigV4 for all services
- April 2019: AWS Asia Pacific (Hong Kong) Region launch
  - 1st opt-in Region





```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnableDisableHongKong",
            "Effect": "Allow",
            "Action": [
                "account:EnableRegion",
                "account:DisableRegion"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {"account:TargetRegion": "ap-east-1"}
            }
        },
        {
            "Sid": "ViewConsole",
            "Effect": "Allow",
            "Action": [
                "aws-portal:ViewAccount",
                "account:ListRegions"
            ],
            "Resource": "*"
        }
    ]
}
```

# Short-Term Keys

Long-Term Keys:
Valid until explicitly revoked

Short-Term Keys:
Automatically expire

# Why?

- Identity Federation
- Role Assumption
- Roles for *

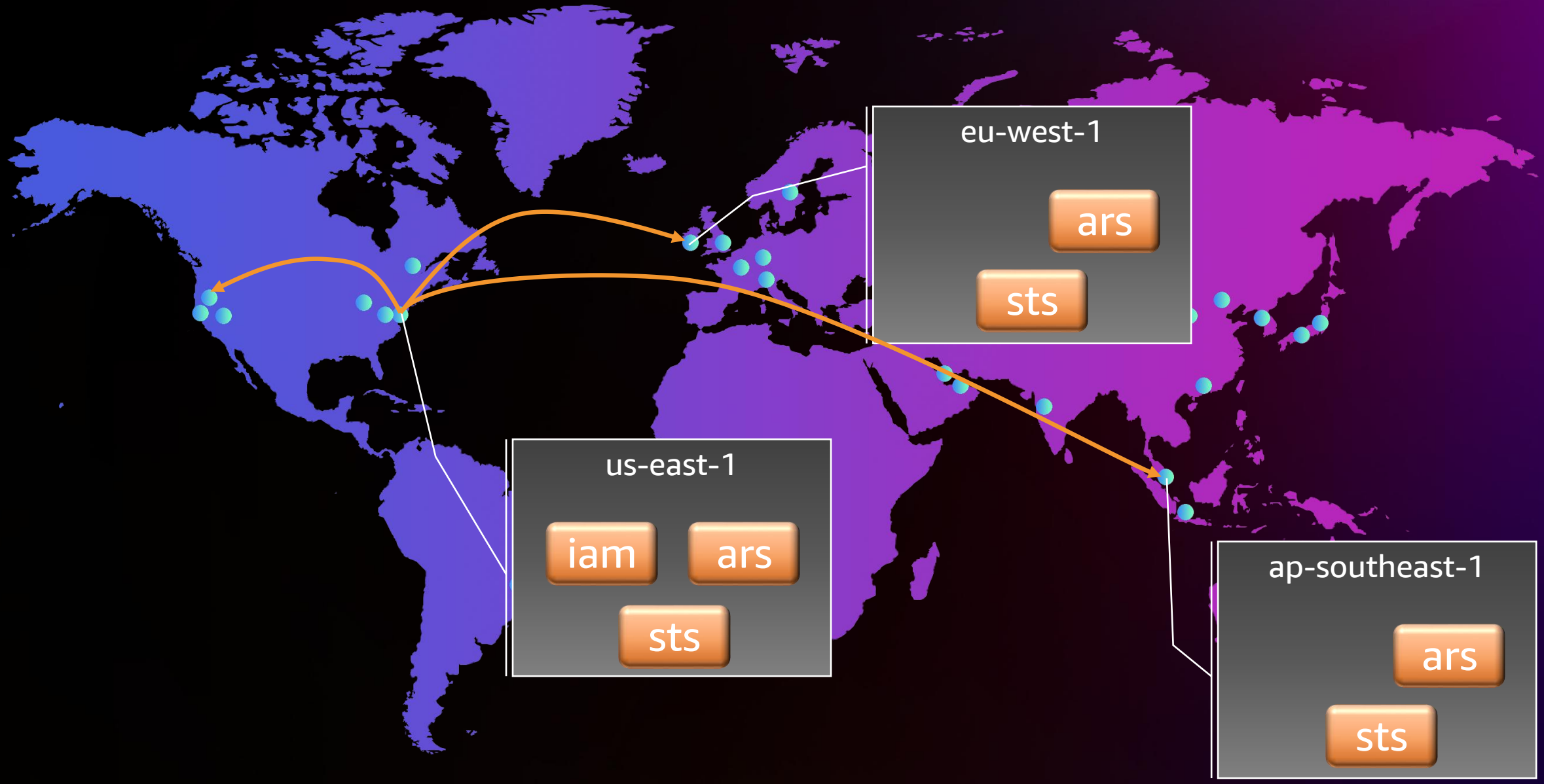
```
TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token"\
  -H "X-aws-ec2-metadata-token-ttl-seconds: 30"` \
&& curl -H "X-aws-ec2-metadata-token: $TOKEN" \
  http://169.254.169.254/latest/meta-data/iam/security-credentials/TestRole
```

```
{
  "Code" : "Success",
  "LastUpdated" : "2022-11-03T23:03:04Z",
  "Type" : "AWS-HMAC",
  "AccessKeyId" : "ASIAQNZGKIQY56JQ7WML",
  "SecretAccessKey" : "x+0Bra63Fr+cER48CUtkHpCxLk8gFV8MawMS0RRF",
  "Token" : "IQoJb3JpZ2luX2VjEOf/////////wEaCXVzLWVhc3QtMSJI <snip>",
  "Expiration" : "2022-11-04T05:28:38Z"
}
```

# Challenges with Long-Term Keys

# Secure Token Service

# Secure Token Service



```
TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token"\
  -H "X-aws-ec2-metadata-token-ttl-seconds: 21600"` \
&& curl -H "X-aws-ec2-metadata-token: $TOKEN" \
  http://169.254.169.254/latest/meta-data/iam/security-credentials/TestRole
```

```
{
  "Code" : "Success",
  "LastUpdated" : "2022-11-03T23:03:04Z",
  "Type" : "AWS-HMAC",
  "AccessKeyId" : "ASIAQNZGKIQY56JQ7WML",
  "SecretAccessKey" : "x+0Bra63Fr+cER48CUtkHpCxLk8gFV8MawMS0RRF",
  "Token" : "IQoJb3JpZ2luX2VjEOf/////////wEaCXVzLWVhc3QtMSJI <snip>",
  "Expiration" : "2022-11-04T05:28:38Z"
}
```

```
TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token"\
-H "X-aws-ec2-metadata-token-ttl-seconds: 21600"` \
&& curl -H "X-aws-ec2-metadata-token: $TOKEN" \
http://169.254.169.254/latest/meta-data/iam/security-credentials/TestRole
```

```
Session token version: one
Role Name: "TestRole"
account_id: "029608264753"
issuer: "AIDADEKZJMX2ZKUJ2YNT4"
encryptionKeyId: -47
CreateDate: 2022-11-03T23:03:04Z
Seconds: 21600
AccessKeyId: "ASIAQNZGKIQY56JQ7WML"
SecretAccessKey: "x+0Bra63Fr+cER48CUtkHpCxLk8gFV8MawMS0RRF"
policy: ...
Signed: sts.us-east-1
```


```
Session token version: one
Role Name: "S3Admin"
account_id: "029608264753"
issuer: "AIDADEKZJMX2ZKUJ2YNT4"
encryptionKeyId: -47
CreateDate: 2022-11-03T23:03:04Z
Seconds: 21600
AccessKeyId: "ASIAQNZGKIQY56JQ7WML"
SecretAccessKey: "x+0Bra63Fr+cER48CUtkHpCxLk8gFV8MawMS0RRF"

                Signed: sts.us-east-1
```


hmac( , "CreateBucket...") →

```
Session token version: one
Role Name: "S3Admin"
account_id: "029608264753"
issuer: "AIDADEKZJMX2ZKUJ2YNT4"
encryptionKeyId: -47
CreateDate: 2022-11-03T23:03:04Z
Seconds: 21600
AccessKeyId: "ASIAQNZGKIQY56JQ7WML"
SecretAccessKey: "x+0Bra63Fr+cER48CUtkHpCxLk8gFV8MawMS0RRF"

                Signed: sts.us-east-1
```

# TLS was expensive and not widespread

# 2006

# Here Come The ⊕ Ninjas

Thai Duong Juliano Rizzo

May 13, 2011

**Abstract**

This paper introduces a fast blockwise chosen-plaintext attack against SSL 3.0 and TLS 1.0. We also describe one application of the attack that allows an attacker to efficiently decrypt and obtain authentication tokens embedded in HTTPS requests[1] The resulting exploits work for major web browsers at the time of writing.

# What just happened?

- Scale and customer requirements lead us to an interesting design
- Scale and growth broke our design
  - Innovation and careful cryptography gave us even greater scale
- Short-lived keys and SigV4 allow us to support new use cases
- Scale cost us simplicity, but not elegance

# Thank you!

Eric Brandwine
@ebrandwine