Chaser

IAM Roles Anywhere – now for everyone with Let's Encrypt

fwd:cloudsec ‘25 Denver

Dhruv AHUJA @new23d www.new23d.com

1 the problem
2 iamra
3 lets encrypt
4 private key

Chaser

Assume an AWS IAM Role:

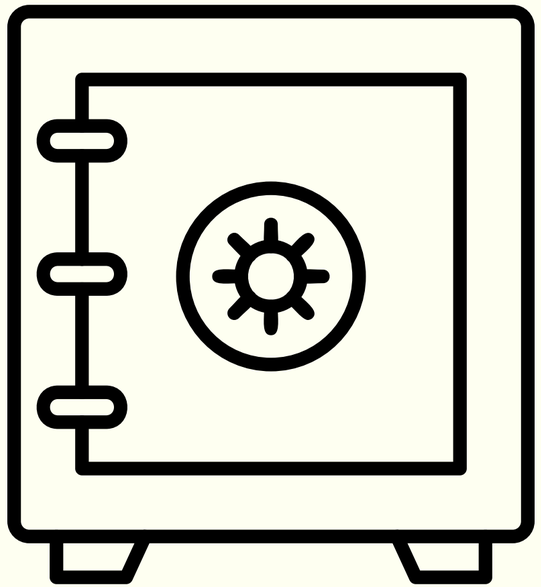
- not on AWS
  - on-prem
  - GPU specialist
  - legacy CI/CD
  - servers
  - workstations
  - laptops
- no static creds
- no OIDC
- no HSM, TPM, etc

solution: IAM Roles Anywhere

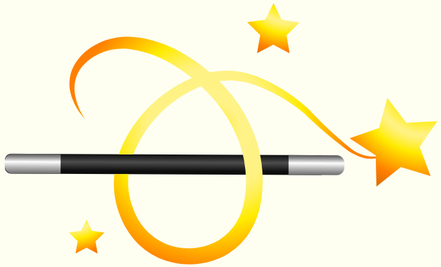
solution: IAM Roles Anywhere

challenges:

- no existing PKI
- no SPIFFE
- AWS Private CA cost
- Cert Authority experience
- cert distribution
- private key protection



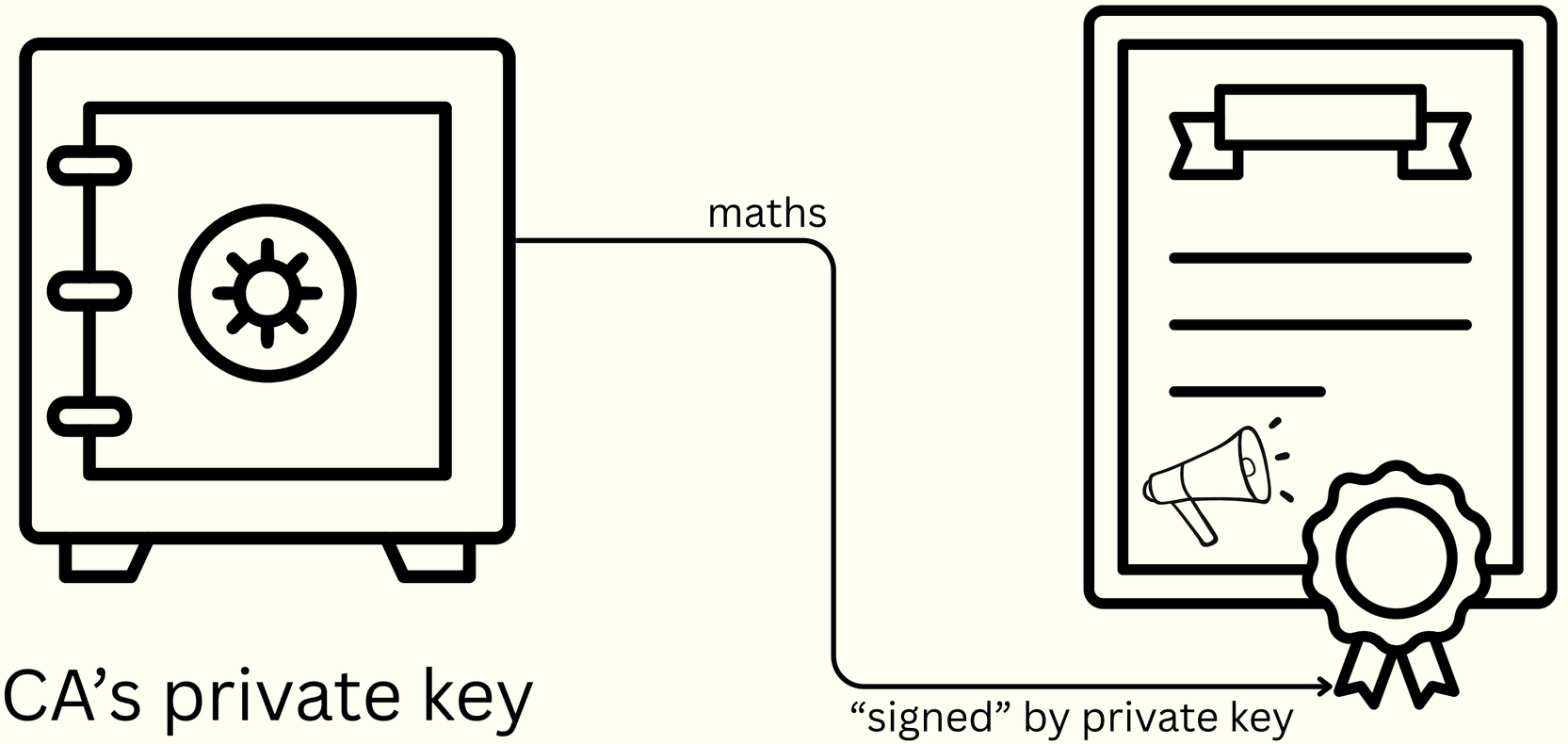
private key



maths

public key

certificate (public)



certificate (public)

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            05:de:1d:ce:4b:da:07:d9:ce:77:9e:44:fc:3a:51:e9:15:3c
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, O = Let's Encrypt, CN = R11
        Validity
            Not Before: May 24 13:01:18 2025 GMT
            Not After : Aug 22 13:01:17 2025 GMT
        Subject: CN = www.chasersystems.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:b6:81:c3:f5:a7:0a:44:44:07:bb:ed:31:3b:7c:
                    59:e5:51:0f:9e:17:fc:35:78:fa:42:8c:af:e4:27
```

certificate (public)

```
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Subject Alternative Name:
                DNS:chasersystems.com, DNS:www.chasersystems.com
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        75:15:bd:5d:2d:a6:da:42:dd:06:f1:7a:f1:f6:20:50:e1:67:
        0d:44:c1:de:9d:89:43:31:e5:98:2e:b5:4d:90:74:b9:58:45
```

web browsing

1. web browser or OS preloaded with CAs public certs: Let’s Encrypt, GlobalSign, Sectigo, GoDaddy, DigiCert, etc
2. you visit something.com, it presents a public certificate allegedly issued by a preloaded CA
3. public key maths, signatures, etc check out
4. certificate data such as expiry date, subject, alternative names, etc also check out
5. remote server something.com is authenticated

~~web browser~~ IAMRA

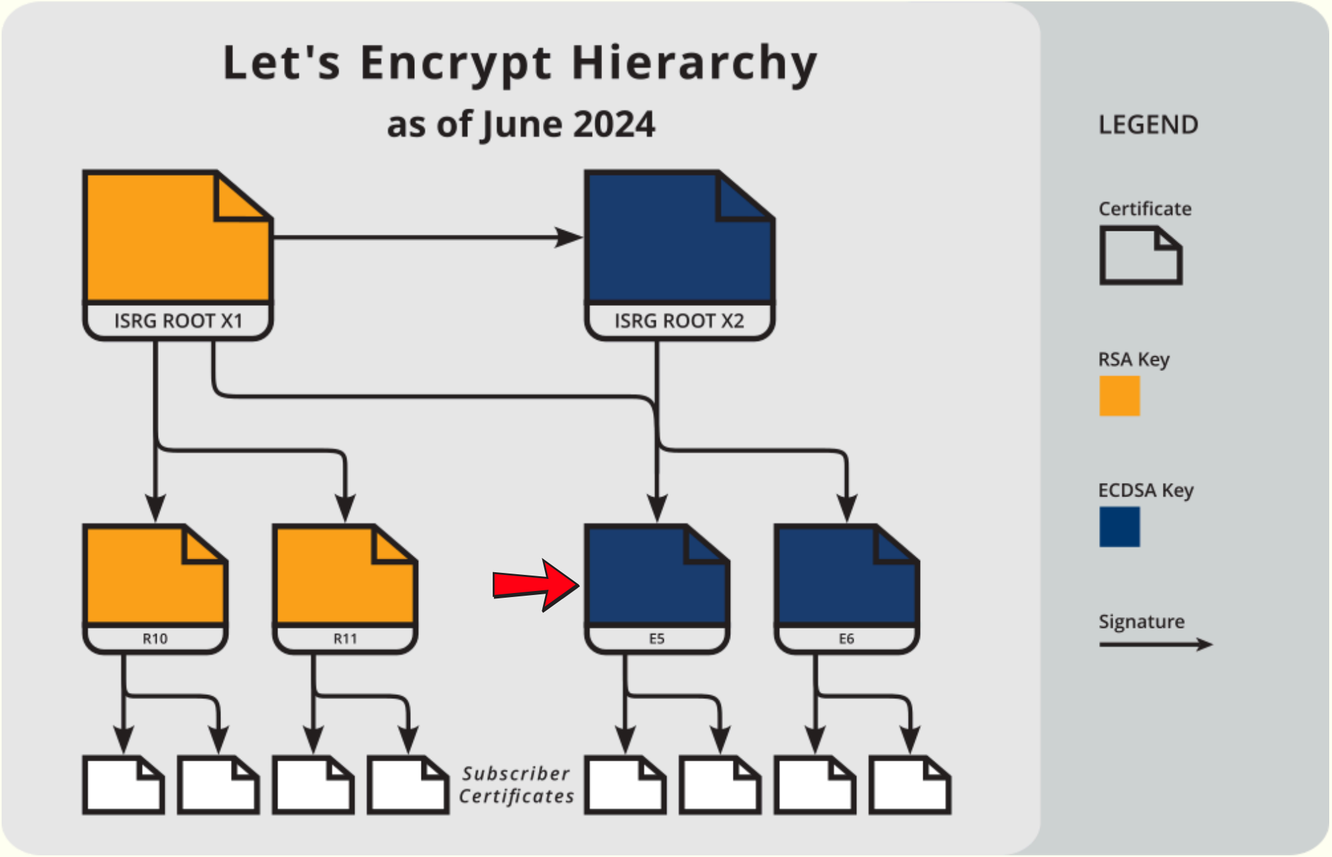
1. load CA certs as IAMRA Trust Anchors
2. create IAM Roles that have the Trust Anchors' ARNs in their Trust Relationships
3. create an IAMRA Profile and add the IAM Roles to it
4. workload generates Private Key
5. workload gets signed Cert (public) from CA
6. workload uses IAMRA Signing Helper with following to call the CreateSession API
    a. Public Cert
    b. Private Key
    c. IAMRA Trust Anchor ARN
    d. IAMRA Profile ARN
    e. IAM Role ARN
7. IAMRA does the key maths, cert dates, checks Trust Policy, Roles in Profile, etc
    a. Private Key never leaves the workload, only used for "signing"
8. if all checks out, temporary Session Token is granted - just like Assume Role!

Let’s Encrypt in IAMRA

1. load Intermediate CA certs as IAMRA Trust Anchors

> **Important**
>
> Certificates issued from public CAs cannot be used as trust anchors.

source: https://docs.aws.amazon.com/rolesanywhere/latest/userguide/getting-started.html



source: https://letsencrypt.org/certificates/

Let’s Encrypt in IAMRA

1. load Staging Intermediate CA certs as IAMRA Trust Anchors

Subordinate (Intermediate) CAs

The staging environment has intermediate certificates that mimic production, issued from the untrusted roots detailed above. Like in production, not all are in use at any time. The full list of current intermediates is:

- (STAGING) Pseudo Plum E5
- (STAGING) False Fennel E6
- (STAGING) Puzzling Parsnip E7
- (STAGING) Mysterious Mulberry E8
- (STAGING) Fake Fig E9
- (STAGING) Counterfeit Cashew R10
- (STAGING) Wannabe Watercress R11
- (STAGING) Riddling Rhubarb R12
- (STAGING) Tenuous Tomato R13
- (STAGING) Not Nectarine R14

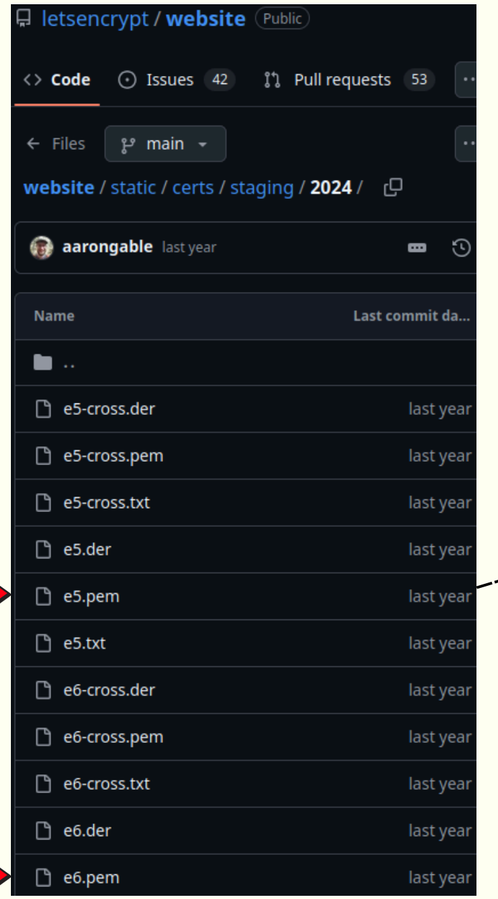
source: https://letsencrypt.org/docs/staging-environment/

because:

- sufficient for use case
- no need for browser-trusted
- we import whatever we want in IAMRA

Let’s Encrypt in IAMRA

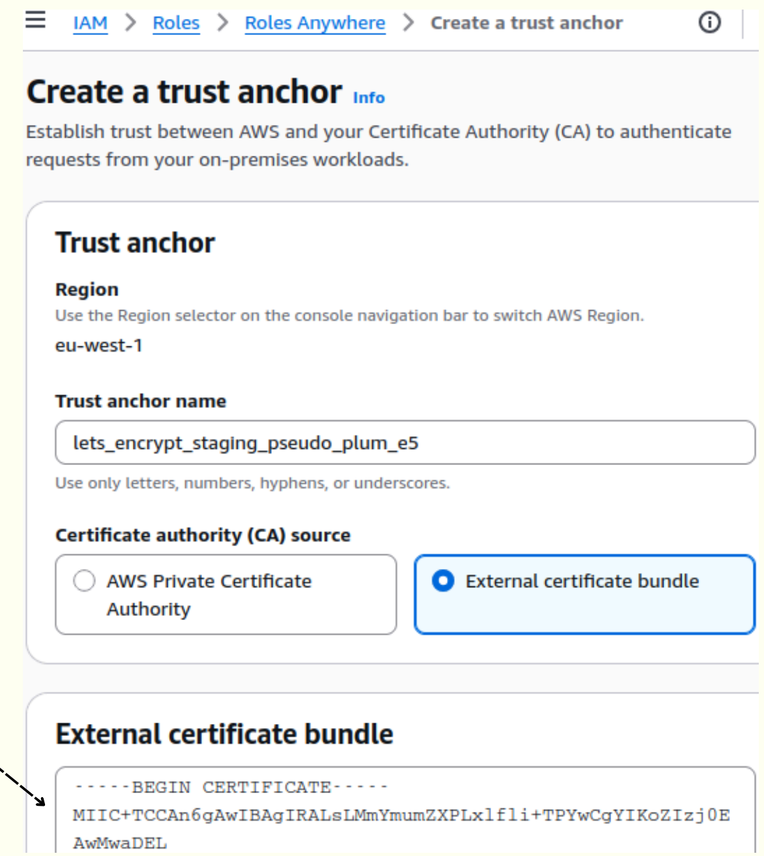
1. load Staging Intermediate CA certs as IAMRA Trust Anchors



source: https://github.com/letsencrypt/website/tree/main/static/certs/staging/2024

e5.pem
e6.pem

concatenate



Let’s Encrypt ACME Challenge Types

HTTP-01*

DNS-01 ✓

TLS-ALPN-01*

TLS-SNI-01*

* needs an open port and a listening process

Let’s Encrypt ACME DNS-01

1. Buy an inconspicuous domain name: not-your-co.com
2. Host it in Route 53
    a. exclusive AWS Account
    b. save Zone ID
3. Create IAM User with a tight policy (next slide)
4. Obtain static credentials 🫣

```
{ "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "route53:GetChange",
      "Resource": "arn:aws:route53:::change/*"
    },
    {
      "Effect": "Allow",
      "Action": "route53:ListHostedZonesByName",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": ["route53:ListResourceRecordSets"],
      "Resource": ["arn:aws:route53:::hostedzone/Z111222333"]
    },
    {
      "Effect": "Allow",
      "Action": ["route53:ChangeResourceRecordSets"],
      "Resource": ["arn:aws:route53:::hostedzone/Z111222333"],
      "Condition": {
        "ForAllValues:StringEquals": {
          "route53:ChangeResourceRecordSetsNormalizedRecordNames": [
            "_acme-challenge.not-your-co.com"
          ],
          "route53:ChangeResourceRecordSetsRecordTypes": ["TXT"]
}}}]}
```

limited blast radius

can’t do much except:

- get “change”
- list zones
- list records of specific Zone
- make TXT record change in specific Zone

source: https://go-acme.github.io/lego/dns/route53/index.html

Let’s Encrypt ACME DNS-01

On workload foo, using **Lego** Let’s Encrypt ACME client:
https://go-acme.github.io/lego/

With Environment Variables: AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY, AWS_DEFAULT_REGION (scoped to ‘service’ on a multi-user system)

On bootstrap (with SystemD), generate key and obtain new certificate:

```
lego --path /etc/lego --accept-tos --email random@not-your-co.com
--dns route53 --domains foo.not-your-co.com --server=https://acme-
staging-v02.api.letsencrypt.org/directory run
```

Then, renew it weekly with Cron or SystemD Service & Timer:

```
lego --path /etc/lego --accept-tos --email random@not-your-co.com
--dns route53 --domains foo.not-your-co.com --server=https://acme-
staging-v02.api.letsencrypt.org/directory renew
```

workload identity

workload foo now has

Private Key: /etc/lego/certificates/foo.not-your-co.com.key

Certificate: /etc/lego/certificates/foo.not-your-co.com.crt

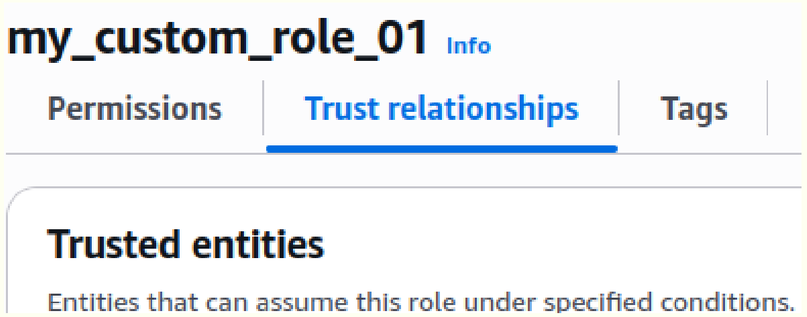
workload identity

workload foo now has

Private Key: /etc/lego/certificates/foo.not-your-co.com.key

Certificate: /etc/lego/certificates/foo.not-your-co.com.crt

IAM



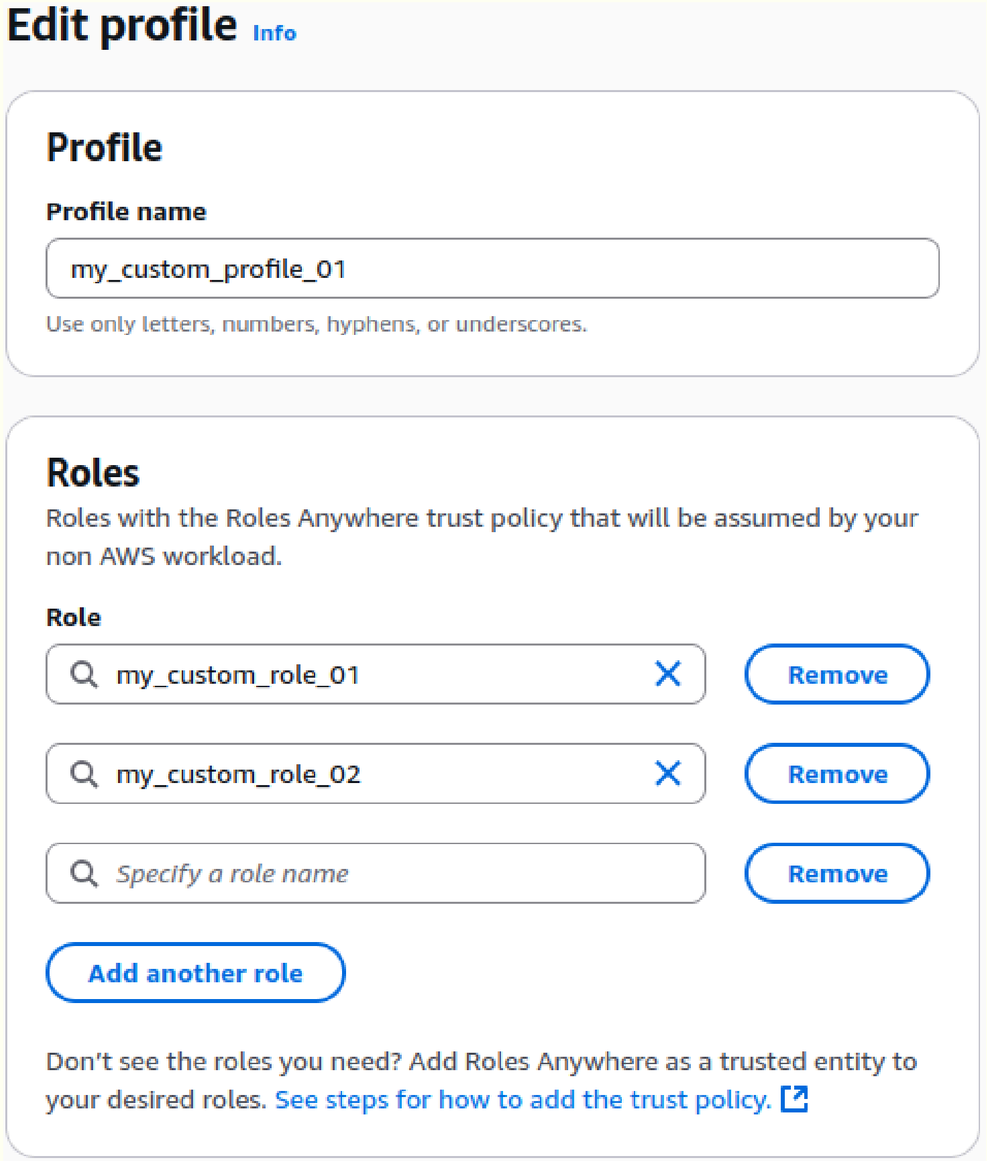
1. create a new IAM Role
2. establish trust between IAMRA and IAM Role
3. conditional on domain name matching in certificate subject - preventing **any** Let's Encrypt cert from working

```
{ "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": ["rolesanywhere.amazonaws.com"]
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringLike": {"aws:PrincipalTag/x509Subject/CN": "*.not-your-co.com"},
        "ArnEquals": {
          "aws:SourceArn": [
            "arn:aws:rolesanywhere:region:account:trust-anchor/TA1_ID",
            "arn:aws:rolesanywhere:region:account:trust-anchor/TA2_ID"
]}}}]}
```

ref: https://docs.aws.amazon.com//rolesanywhere/latest/userguide/getting-started.html

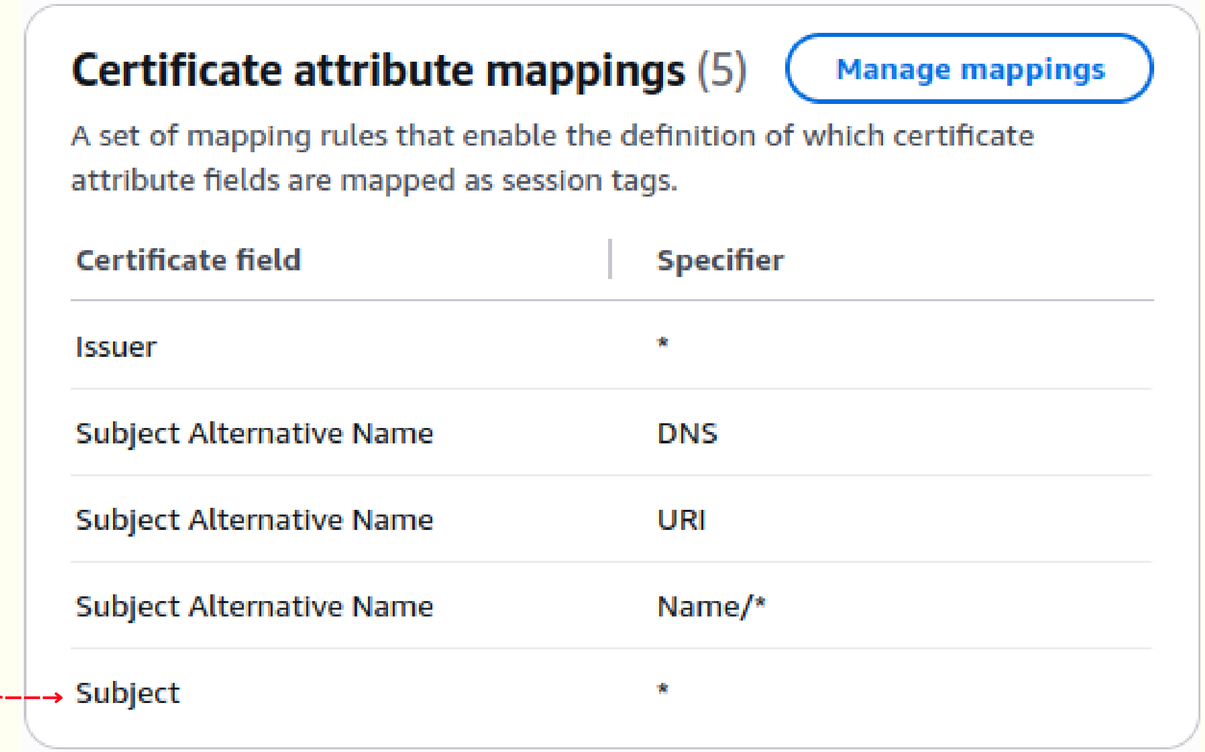
IAMRA Profile

1. create new Profile in IAMRA
2. add all IAM Roles to it you'd want workloads to be able to Assume
3. only IAM Roles with Trust established to rolesanywhere.amazonaws.com will be available



IAMRA Profile

1. "aws:PrincipalTag/x509Subject/CN" was available in Trust Policy from this mapping
2. may as well delete unused mappings



source: https://docs.aws.amazon.com/rolesanywhere/latest/userguide/attribute-mapping.html

test on workload

file: ~/.aws/credentials

```
[app_profile]
credential_process=/path/to/gen-creds.sh
```

file: gen-creds.sh

```
#!/bin/bash
aws_signing_helper credential-process \
  --region eu-west-2 \
  --certificate /etc/lego/certificates/foo.not-your-co.com.crt \
  --private-key /etc/lego/certificates/foo.not-your-co.com.key  \
  --trust-anchor-arn <ARN_OF_IAMRA_TRUST_ANCHOR> \
  --profile-arn <ARN_OF_IAMRA_PROFILE> \
  --role-arn <ARN_OF_IAM_ROLE_TO_ASSUME>
```

https://github.com/aws/rolesanywhere-credential-helper

test on workload

```
export AWS_PROFILE=app_profile

aws sts get-caller-identity
```

key protection

```
aws_signing_helper credential-process \
  --region eu-west-2 \
  --certificate /etc/lego/certificates/foo.not-your-co.com.crt \
  --private-key /etc/lego/certificates/foo.not-your-co.com.key \
  --trust-anchor-arn <ARN_OF_IAMRA_TRUST_ANCHOR> \
  --profile-arn <ARN_OF_IAMRA_PROFILE> \
  --role-arn <ARN_OF_IAM_ROLE_TO_ASSUME>
```

key in hsm

using Yubikey's ykman CLI or Yubico Authenticator GUI

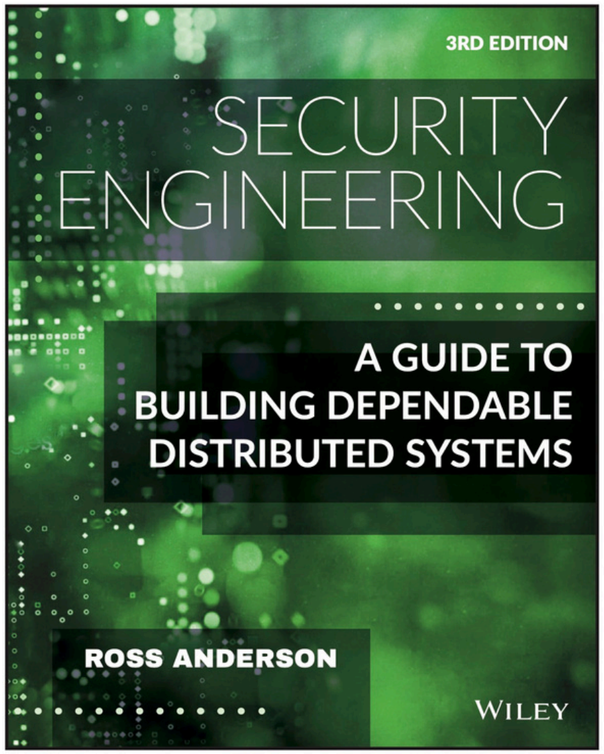
```
ykman piv keys export 9c --format pem foo.public

ykman piv certificates request --subject CN=foo.not-your-co.com 9c
foo.public foo.csr

lego --path /etc/lego --accept-tos --email random@not-your-co.com --dns
route53 --csr foo.csr --server=https://acme-staging-
v02.api.letsencrypt.org/directory run
```

AWS Signing Helper has excellent docs on use of Private Key from PKCS11, TPMs, HSMs, etc

environmental sensing



Ross Anderson
15 September 1956 – 28 March 2024

Professor of Security Engineering at the Department of Computer Science and Technology, University of Cambridge

Security Engineering book download: https://www.cl.cam.ac.uk/archive/rja14/book.html

environmental sensing

Chapter: Nuclear Command and Control

*The basic principle was that **a unique aspect of the environment had to be sensed before the weapon would arm**. For example, missile warheads and some free-fall bombs had to experience zero gravity, while artillery shells had to experience an acceleration of thousands of G.*

*Environment: the **weapon must have sensed the appropriate aspect of the environment**. (With atomic demolition munitions, this **requirement is replaced by the use of a special container**.)*

source: Security Engineering third edition

environmental sensing

1. pick invariant(s) from the environment
    - static assets
    - platform hints
        - /sys/*, /proc/*
        - SPIFFE: node/workload attestation

environmental sensing

1. pick invariant(s) from the environment
    - static assets
    - platform hints
        - /sys/*, /proc/*
        - SPIFFE: node/workload attestation
    - NFTs

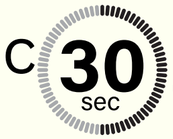
environmental sensing

1. pick invariant(s) from the environment
    - static assets
    - platform hints
        - /sys/*, /proc/*
        - SPIFFE: node/workload attestation
    - NFTs
2. use fingerprint as seed for TOTP
3. store SHA1* checksum for detection side logic
4. pass in to Custom Role Session Name from workload

**or SHA256*

environmental sensing

1. pick invariant(s) from the environment
    - static assets
    - platform hints
        - /sys/*, /proc/*
        - SPIFFE: node/workload attestation
    - NFTs
2. use fingerprint as seed for TOTP
3. store SHA1* checksum for detection side logic
4. pass in to Custom Role Session Name from workload



221250
191482
319897
270861
592194

**or SHA256*

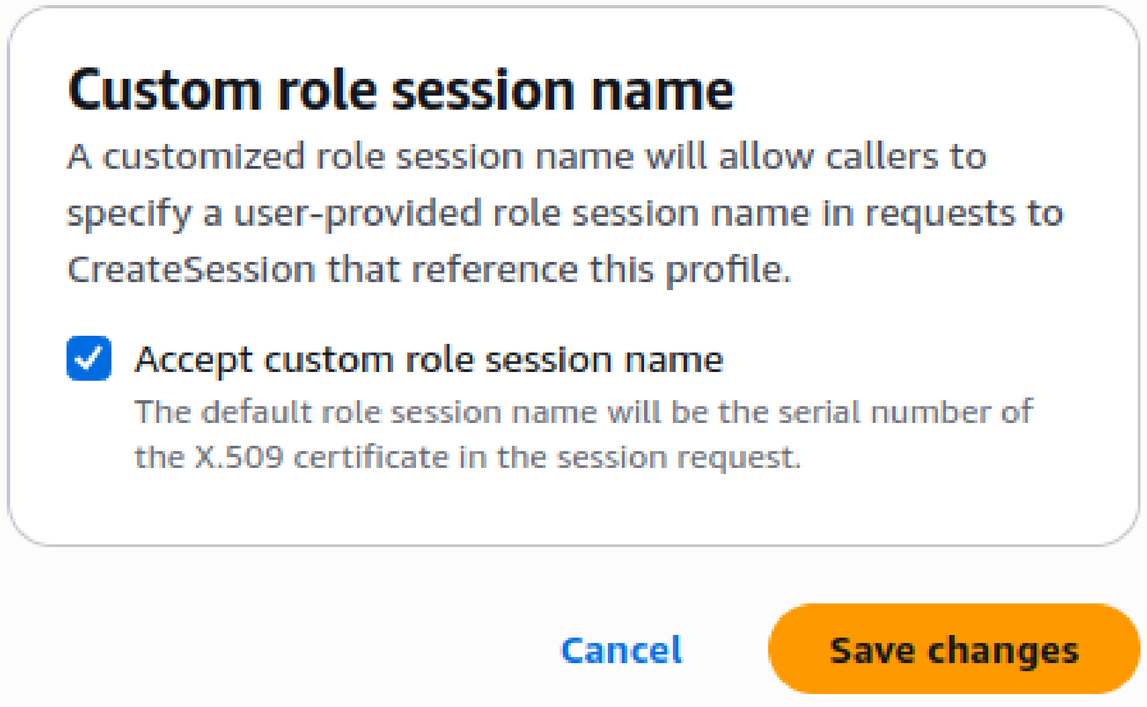
DB of environmental checksums

`env-sha1sum-db.json`

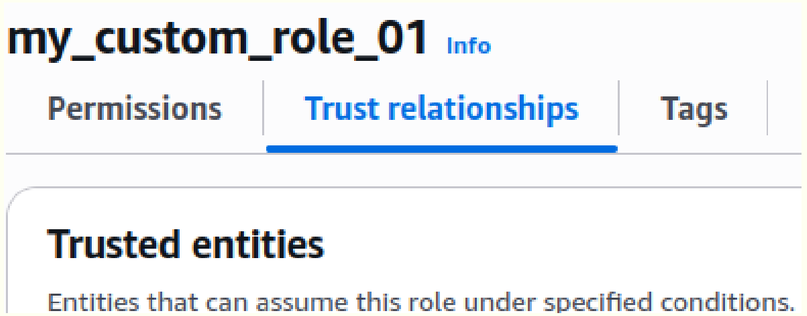
```
[
  {
    "CN=foo.not-your-co.com": "fd59103ec3ab3c7ac7f53550ec526648b91d7676"},
  {
    "CN=bar.not-your-co.com": "d22b880fe75d3940678a7337dbd33947588a5362"},
  {
    "CN=egg.not-your-co.com": "026c748e1934627865a2af27ab05e8a02d350c59"},
  {
    "CN=spa.not-your-co.com": "6bb345934e84464ce012c4703414b374789b4d51"}
]
```

perhaps store as a Secret in Secrets Manager;
important that you don't supply these and they are fingerprints of the env

IAMRA Profile



IAM



force supply of a 6 char session name

```
{ "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": ["rolesanywhere.amazonaws.com"]
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringLike": {"aws:PrincipalTag/x509Subject/CN": "*.not-your-co.com",
                       "sts:RoleSessionName": "??????"}
        "ArnEquals": {
          "aws:SourceArn": [
            "arn:aws:rolesanywhere:region:account:trust-anchor/TA1_ID",
            "arn:aws:rolesanywhere:region:account:trust-anchor/TA2_ID"
]}}}]}
```

key protection

file: gen-creds.sh

```
#!/bin/bash
aws_signing_helper credential-process \
  --region eu-west-2 \
  --certificate /etc/lego/certificates/foo.not-your-co.com.crt \
  --private-key /etc/lego/certificates/foo.not-your-co.com.key  \
  --trust-anchor-arn <ARN_OF_IAMRA_TRUST_ANCHOR> \
  --profile-arn <ARN_OF_IAMRA_PROFILE> \
  --role-arn <ARN_OF_IAM_ROLE_TO_ASSUME> \
  --role-session-name <ENVIRONMENTAL_CHECKSUM>
```

TOTP

oathtool

```
oathtool --totp fd59103ec3ab3c7ac7f53550ec526648b91d7676
123456

oathtool --totp fd59103ec3ab3c7ac7f53550ec526648b91d7676 --now
"2025-06-25T11:18:59Z"
227749

oathtool --totp fd59103ec3ab3c7ac7f53550ec526648b91d7676 --now
"2025-06-25T11:18:59Z" --window 1
227749
880050
```

apt install oathtool (Debian & derivatives, Ubuntu, etc)
apk add oath-toolkit-oathtool (Alpine)
dnf install oathtool (EPEL for Red Hat & derivatives, CentOS, Rocky, etc)

key protection

file: gen-creds.sh

```
#!/bin/bash
TOTP=$(oathtool --totp $(sha1sum /opt/monkey.jpg | cut -d' ' -f1))
aws_signing_helper credential-process \
  --region eu-west-2 \
  --certificate /etc/lego/certificates/foo.not-your-co.com.crt \
  --private-key /etc/lego/certificates/foo.not-your-co.com.key  \
  --trust-anchor-arn <ARN_OF_IAMRA_TRUST_ANCHOR> \
  --profile-arn <ARN_OF_IAMRA_PROFILE> \
  --role-arn <ARN_OF_IAM_ROLE_TO_ASSUME> \
  --role-session-name $TOTP
```

very important that the environment is checksummed on every invocation
don't embed the checksum here
*therefore, can't use **aws_signing_helper serve***
app should cache creds; app may renew creds ahead of time

cloudtrail log event


```
{ "eventTime": "2025-06-25T11:18:59Z",
  "eventSource": "rolesanywhere.amazonaws.com",
  "eventName": "CreateSession",
  "sourceIPAddress": "203.0.113.44",
  "requestParameters": {
    "profileArn": "...",
    "roleArn": "...",
    "trustAnchorArn": "...",
    "cert": "...base64 cert...",
    "durationSeconds": 3600,
    "roleSessionName": "444444"
  },
  "responseElements": {
    "subjectArn": "...certificate arn...",
    "x509Subject": "CN=foo.not-your-co.com"
}}
```

mismatch detection script

```
#!/bin/bash

set -uo pipefail

 who=$(jq -r .responseElements.x509Subject "$1")
when=$(jq -r .eventTime "$1")
with=$(jq -r .requestParameters.roleSessionName "$1")

env_db_file="env-sha1sum-db.json"

seed=$(jq -r ".[] | .\"$who\" // empty" $env_db_file)

when_minus_30s=$(date -R -u -d "$when - 30 seconds")

found=$(oathtool --window 1 --now "$when_minus_30s" \
        --totp $seed $with)
```

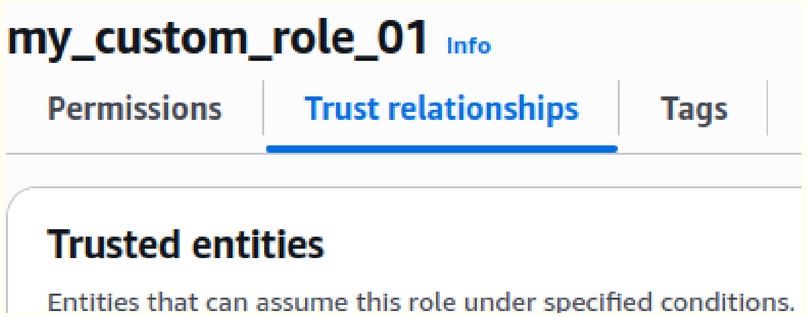
TOTP: RFC6238; network latency and time windows

```
if [ "$found" -le 1 ]; then
 echo "legit"
 _ec=0
else
 echo "oh no"
 _ec=1
fi

exit $_ec
```

caught in one misstep

IAM



revocation alternative

1. IAMRA support CRLs but Let's Encrypt only updates them once a week or so
2. In other cases, it's not generated at all
3. Deny immediately in Trust Policy of the Role

```
{ "Effect": "Deny",
  "Principal": {
    "Service": "rolesanywhere.amazonaws.com"
  },
  "Action": "sts:AssumeRole",
  "Condition": {
    "StringEqualsIgnoreCase": {
      "aws:PrincipalTag/x509Subject/CN": "bar.not-your-co.com"
}}}
```

challenge	solution
no existing PKI	Let’s Encrypt
no SPIFFE	ACME
AWS Private CA cost	Let’s Encrypt
Cert Authority experience	Let’s Encrypt
cert distribution	ACME
private key protection	2FA Detection

Also Consider

- IAMRA is regional
- Lowering Session Duration to 15 minutes
- Using the upcoming Let’s Encrypt 6-day valid certs
  - future/TBA “shortlived” profile
- Monitoring the Certificate Transparency log
- ACME alternatives to Let’s Encrypt
  - zerossl.com
  - buypass.com
  - self-host: https://github.com/letsencrypt/boulder

CHASER

Thank You

Q&A

Dhruv AHUJA @new23d www.new23d.com