It’s a long way to the auth if you want a SIGv4 roll

Authenticating applications with AWS SIGv4
https://github.com/dushan-talks/bsides-melb-2019

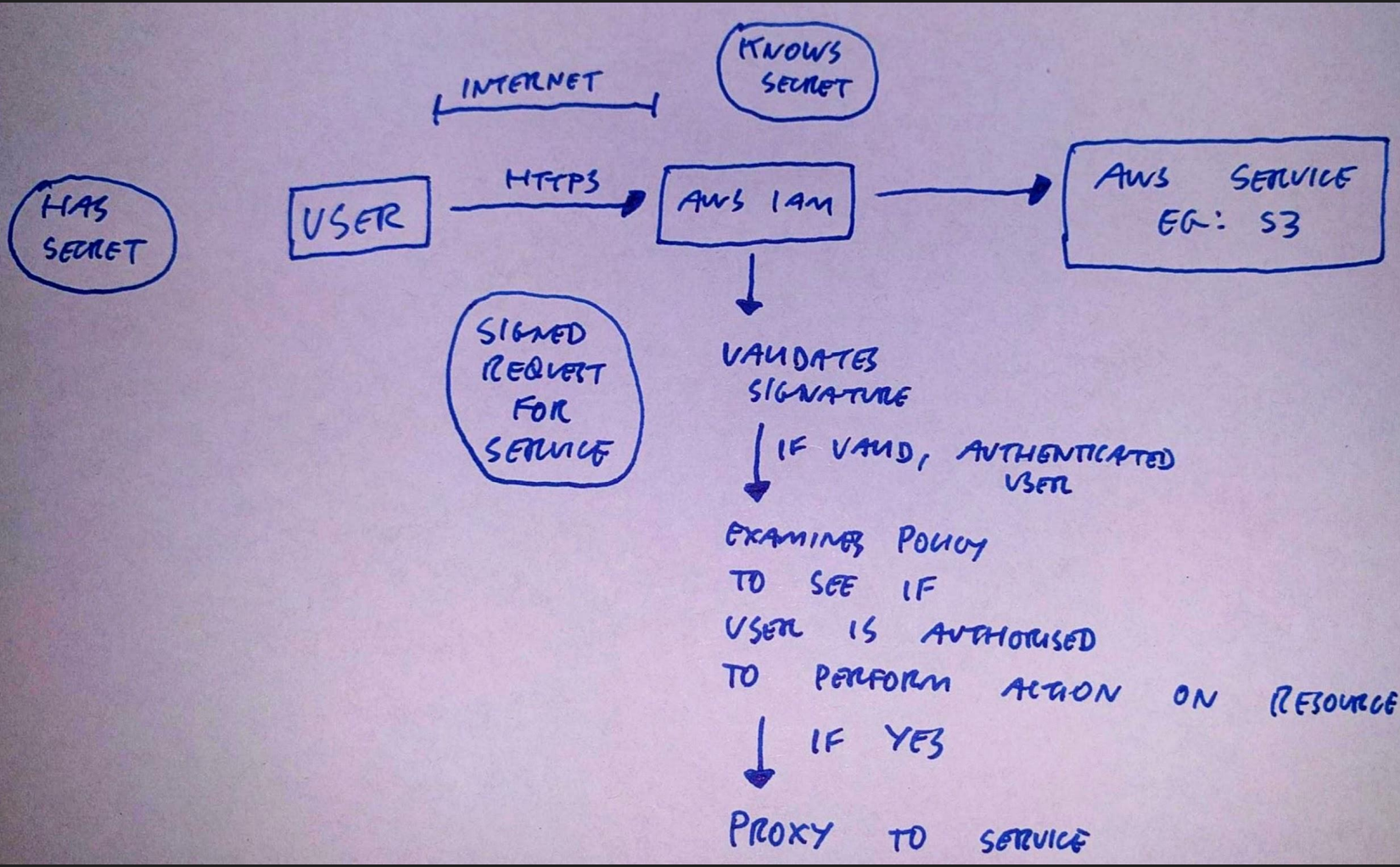
Authentication is challenging

Especially with shared secrets

Why not use a PKI?

Could we leverage the fact that both parties trust AWS?

Authentication in AWS

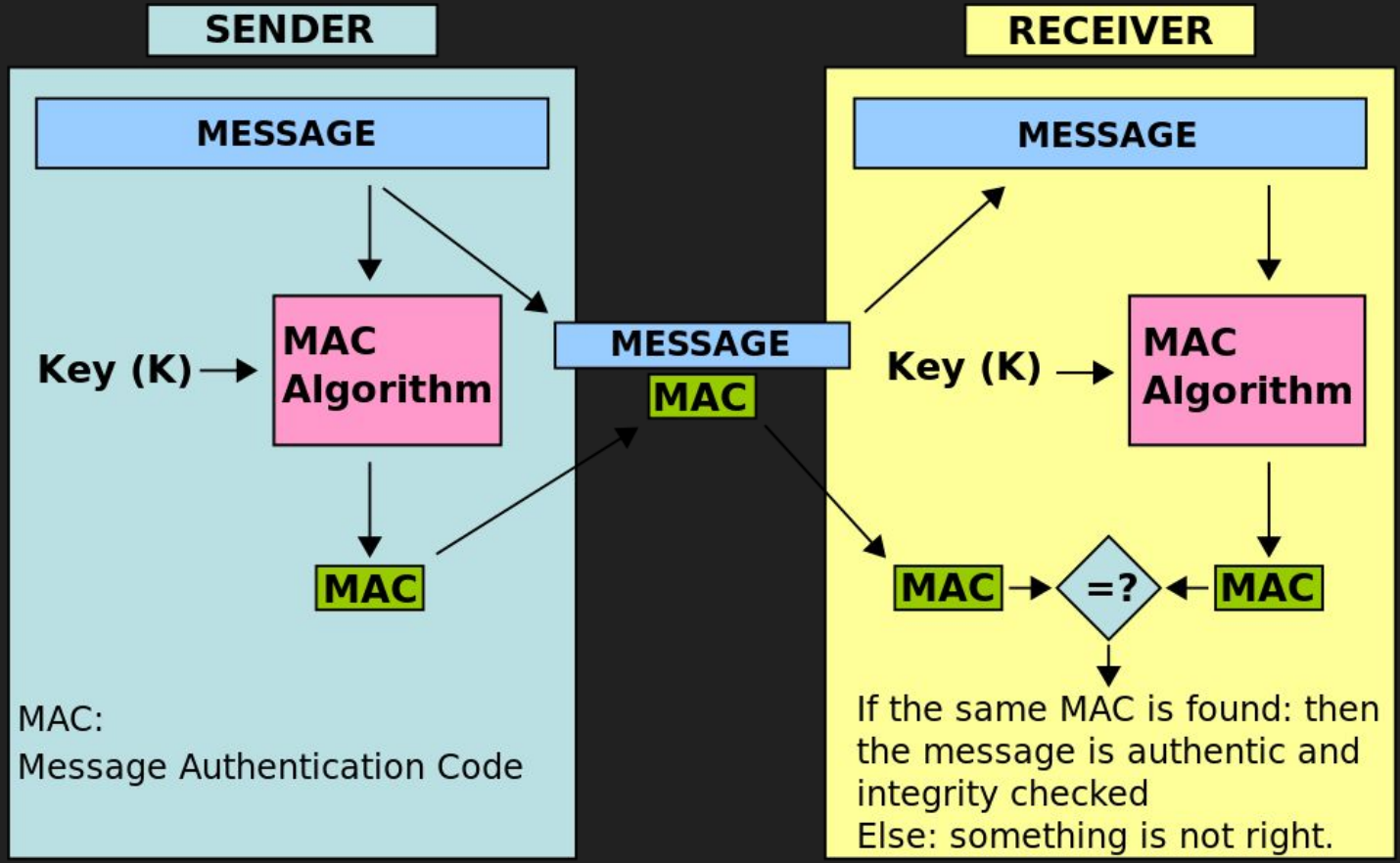

INTERNET
KNOWS SECRET
HAS SECRET
USER
HTTPS
AWS IAM
AWS SERVICE
EG: S3
SIGNED REQUEST FOR SERVICE
VALIDATES SIGNATURE
IF VALID, AUTHENTICATED USER
EXAMINES POLICY TO SEE IF USER IS AUTHORISED TO PERFORM ACTION ON RESOURCE
IF YES
PROXY TO SERVICE

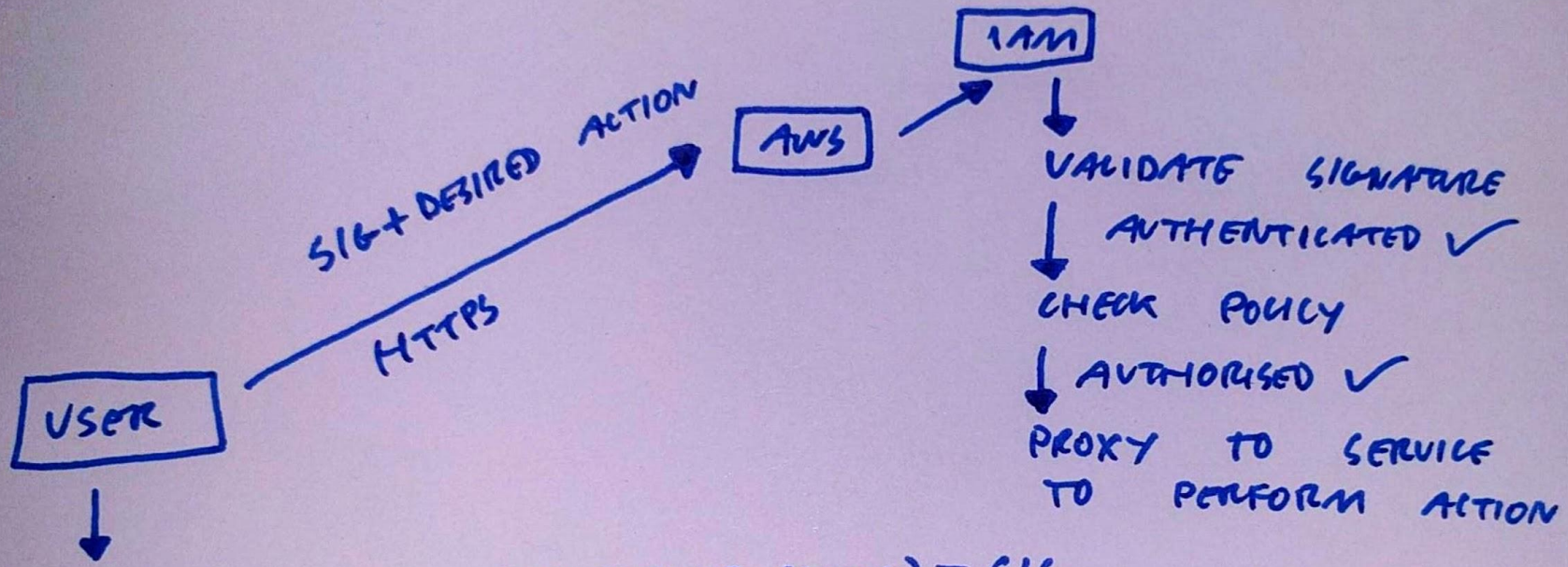
But how is the request authenticated?

Signature Version 4

https://docs.aws.amazon.com/general/latest/gr/signature-version-4.html

Which is really just a HMAC


SENDER
RECEIVER
MESSAGE
MESSAGE
Key (K)
MAC Algorithm
MESSAGE
MAC
Key (K)
MAC Algorithm
MAC
MAC
=?
MAC
MAC:
Message Authentication Code
If the same MAC is found: then the message is authentic and integrity checked
Else: something is not right.


IAM
AWS
SIG + DESIRED ACTION
HTTPS
USER
VALIDATE SIGNATURE
AUTHENTICATED ✓
CHECK POLICY
AUTHORISED ✓
PROXY TO SERVICE TO PERFORM ACTION
SIGV4(SECRET, METADATA, DESIRED ACTION) = SIG

In AWS everything has a globally unique “Amazon Resource Name”

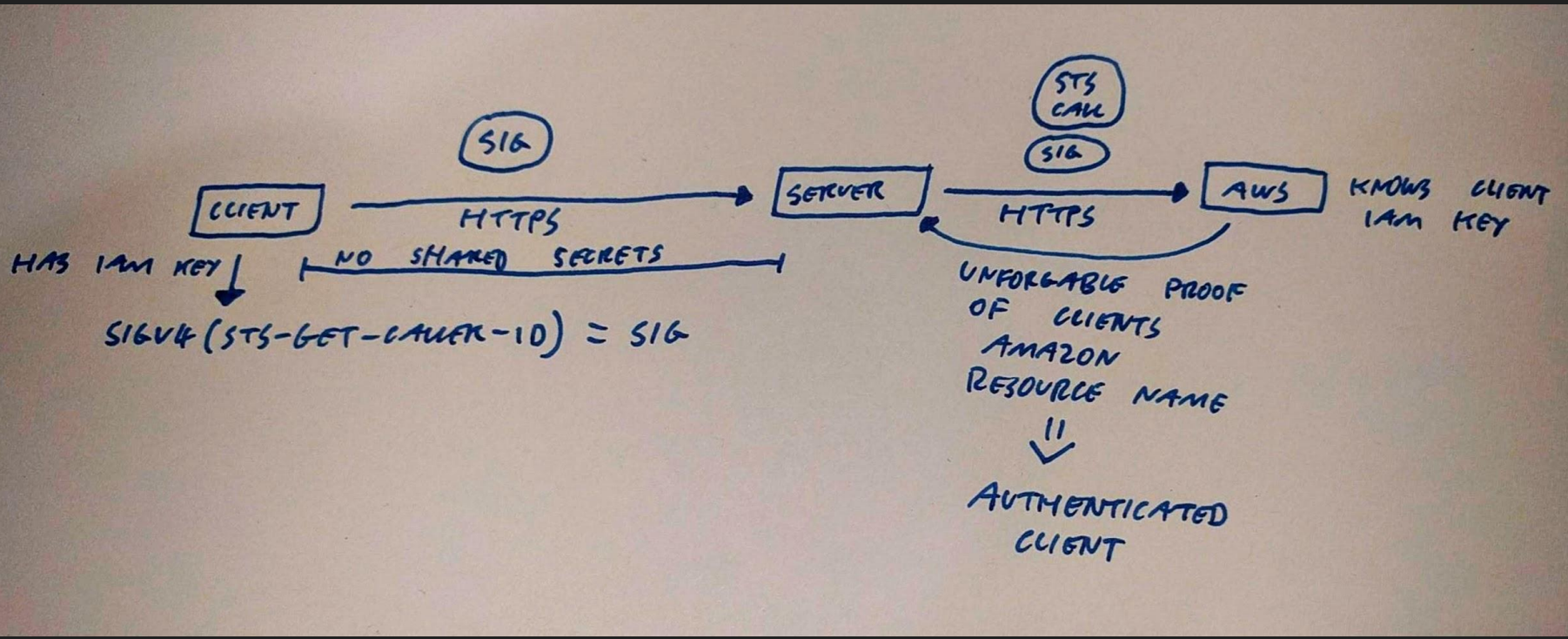
There’s an API to query yours

```
$ aws sts get-caller-identity
{
    "Account": "123456789012",
    "UserId": "foo",
    "Arn": "arn:aws:iam::123456789012:user/foo"
}
```

This is very useful from an authentication perspective

As is the fact that a SIGv4 can be computed locally and transmitted later

With this knowledge and the above primitives, we can build our own authentication protocol


SIG
CLIENT
HTTPS
SERVER
STS CALL
SIG
AWS
HTTPS
KNOWS CLIENT IAM KEY
HAS IAM KEY
NO SHARED SECRETS
SIGV4(STS-GET-CALLER-ID) = SIG
UNFORGABLE PROOF OF CLIENTS AMAZON RESOURCE NAME
AUTHENTICATED CLIENT

Thanks, questions?