# SECRETS AT SCALE

**Manish Mehta**

Security Engineer

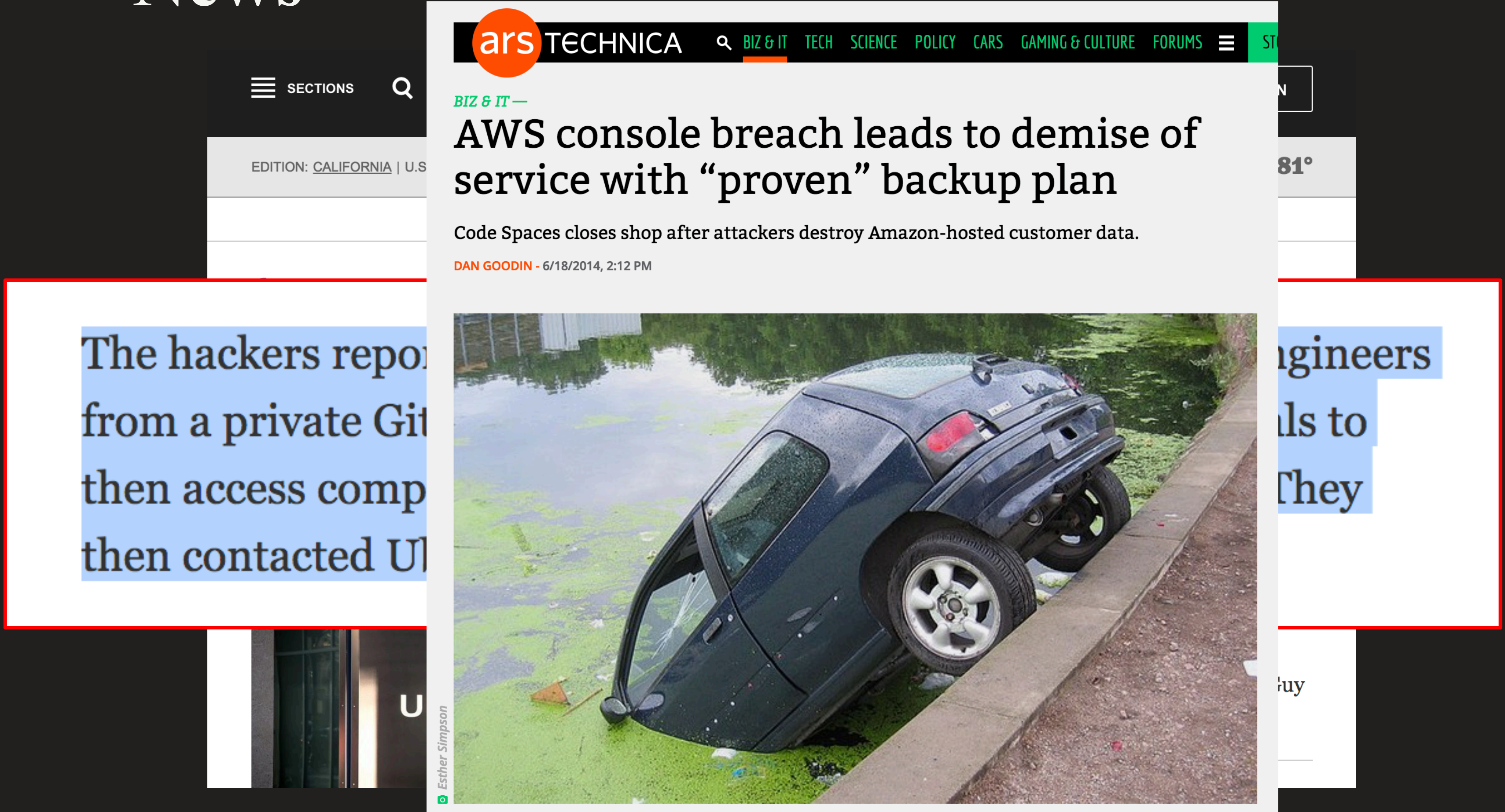
Jan 11, 2018 @ RWC 2018

NETFLIX

# Disclaimer

- Design discussions and statements in this presentation do not necessarily reflect Netflix’s future business plans
- Parts of this presentation are under a US patent (pending)

# News



The hackers repo ngineers

from a private Git ls to

then access comp They

then contacted U

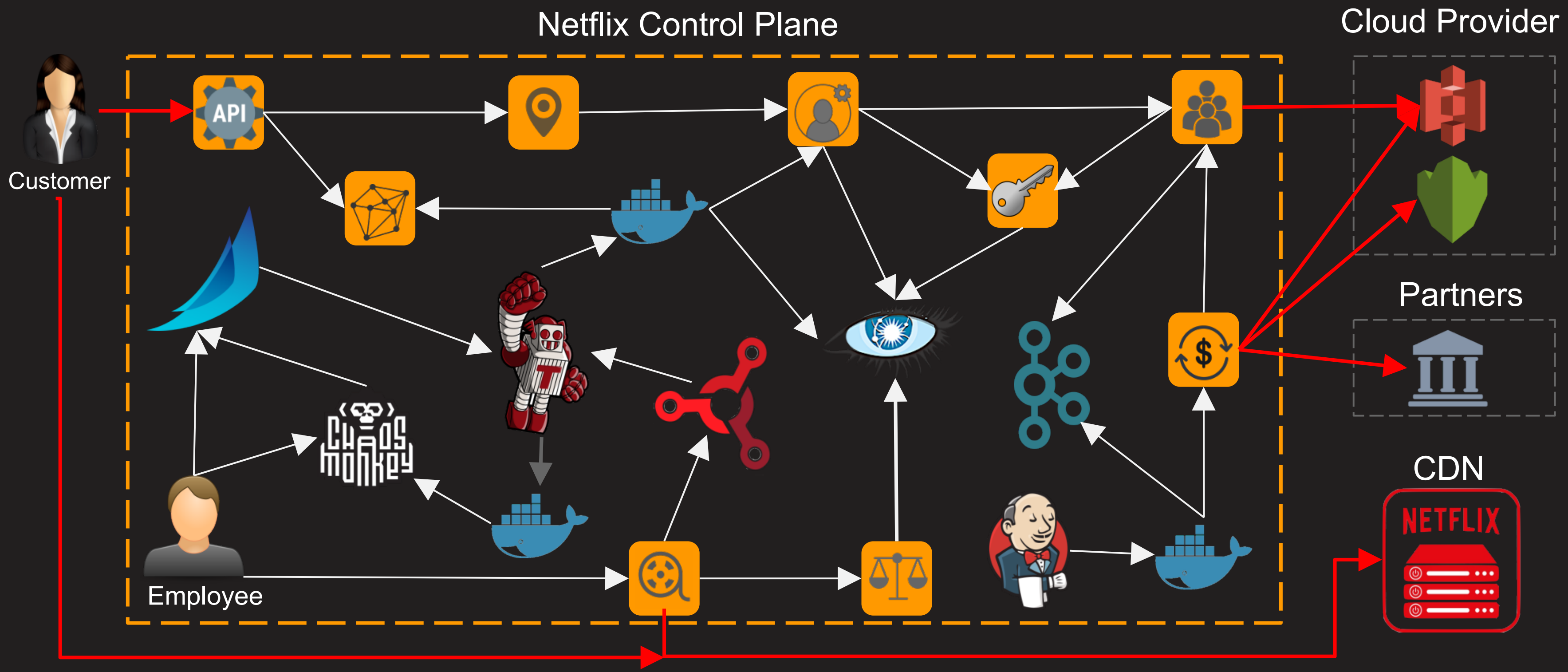
BIZ & IT —

AWS console breach leads to demise of service with “proven” backup plan

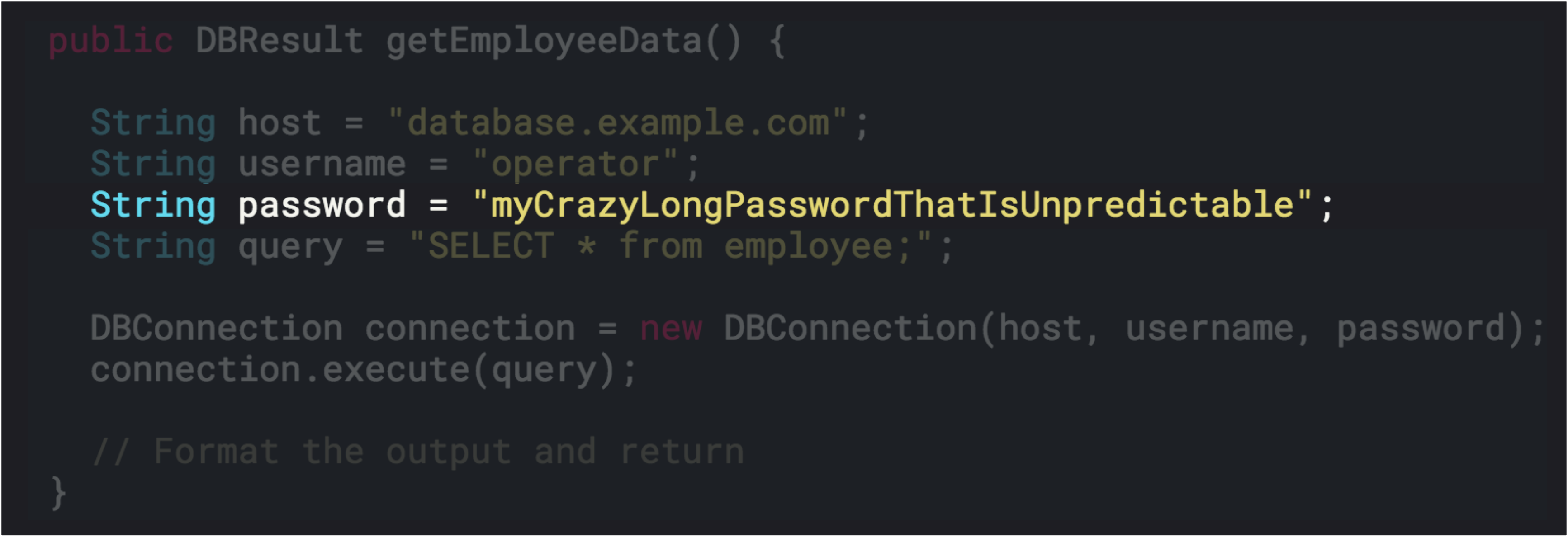
Code Spaces closes shop after attackers destroy Amazon-hosted customer data.

DAN GOODIN - 6/18/2014, 2:12 PM

# Netflix Architecture

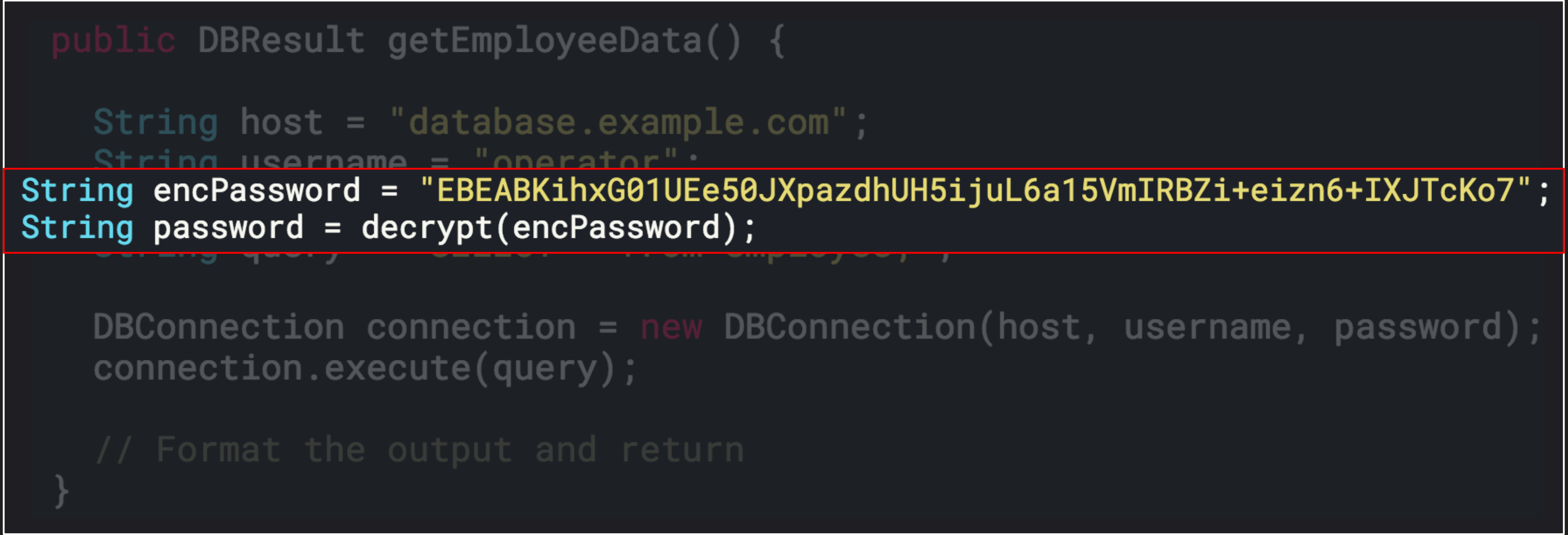


# Let’s build a story



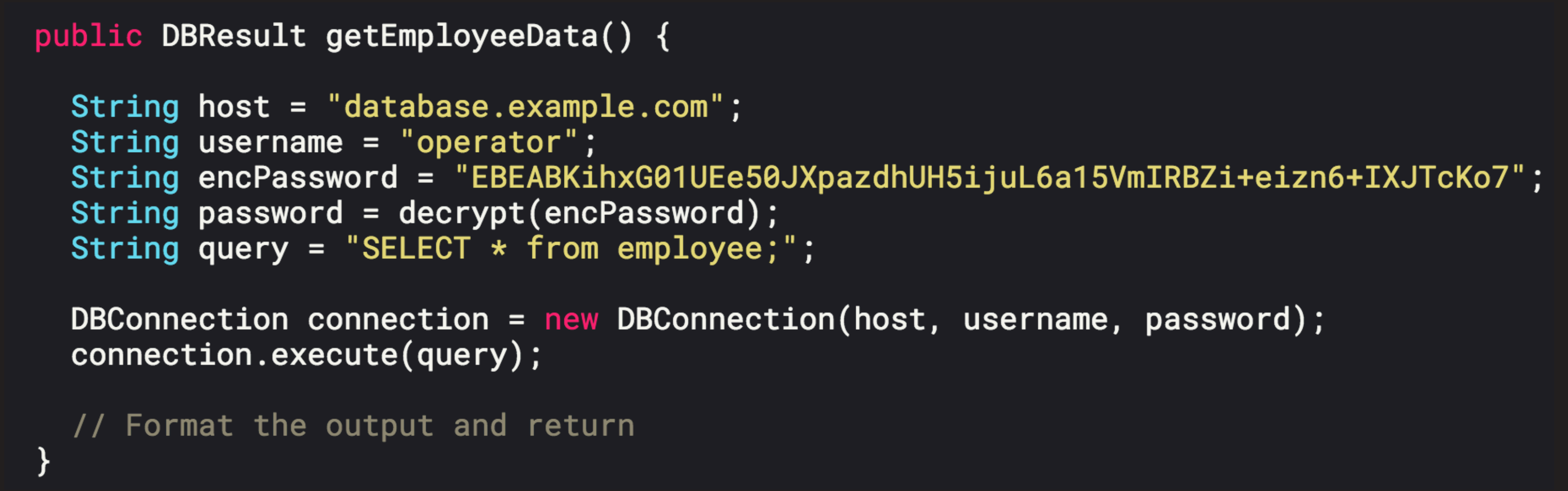
```
public DBResult getEmployeeData() {

  String host = "database.example.com";
  String username = "operator";
  String password = "myCrazyLongPasswordThatIsUnpredictable";
  String query = "SELECT * from employee;";

  DBConnection connection = new DBConnection(host, username, password);
  connection.execute(query);

  // Format the output and return
}
```

# Let’s build a story



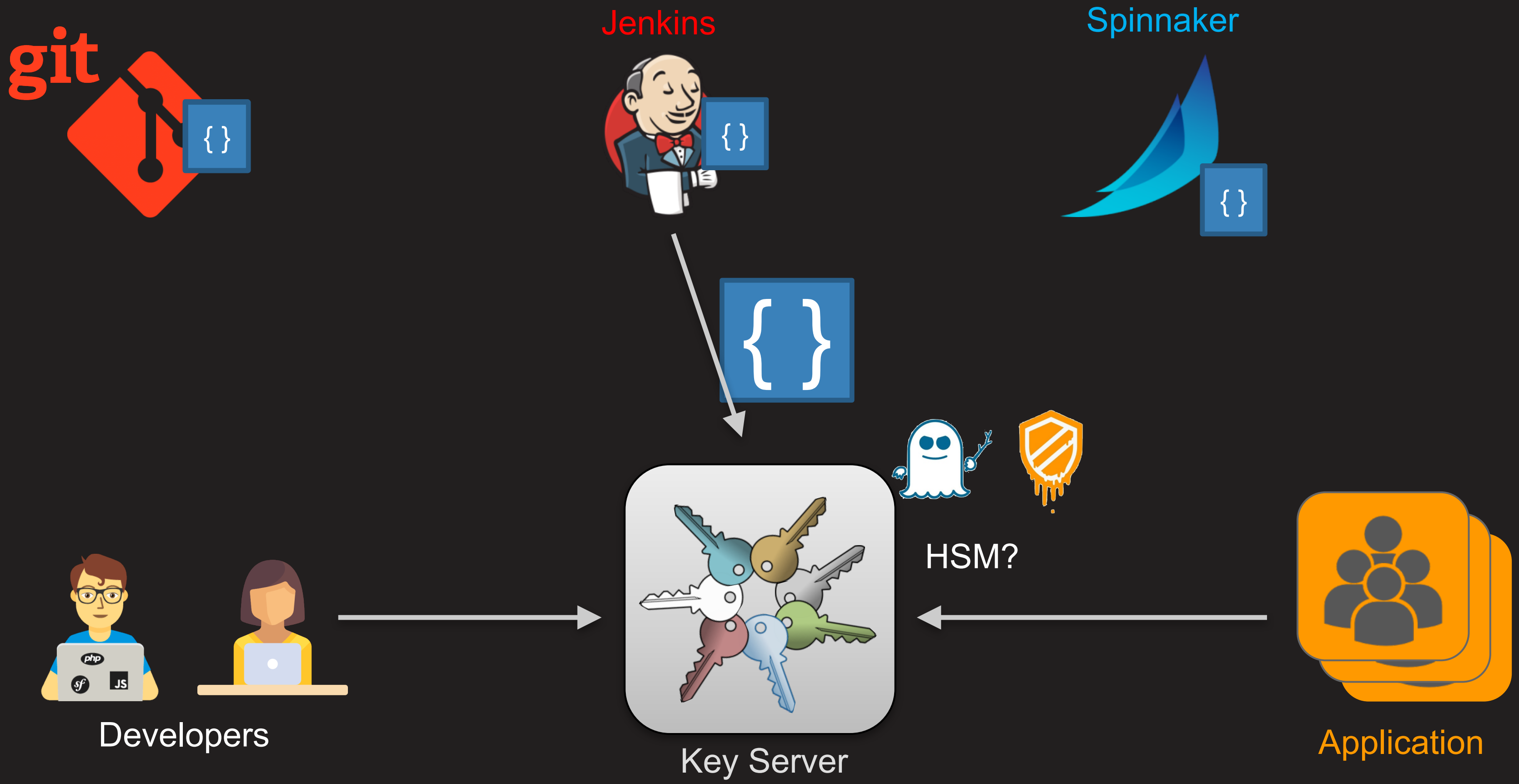
```
public DBResult getEmployeeData() {

    String host = "database.example.com";
    String username = "operator";
String encPassword = "EBEABKihxG01UEe50JXpazdhUH5ijuL6a15VmIRBZi+eizn6+IXJTcKo7";
String password = decrypt(encPassword);
    String query = "SELECT * from employees";

    DBConnection connection = new DBConnection(host, username, password);
    connection.execute(query);

    // Format the output and return
}
```

# Let’s build a story

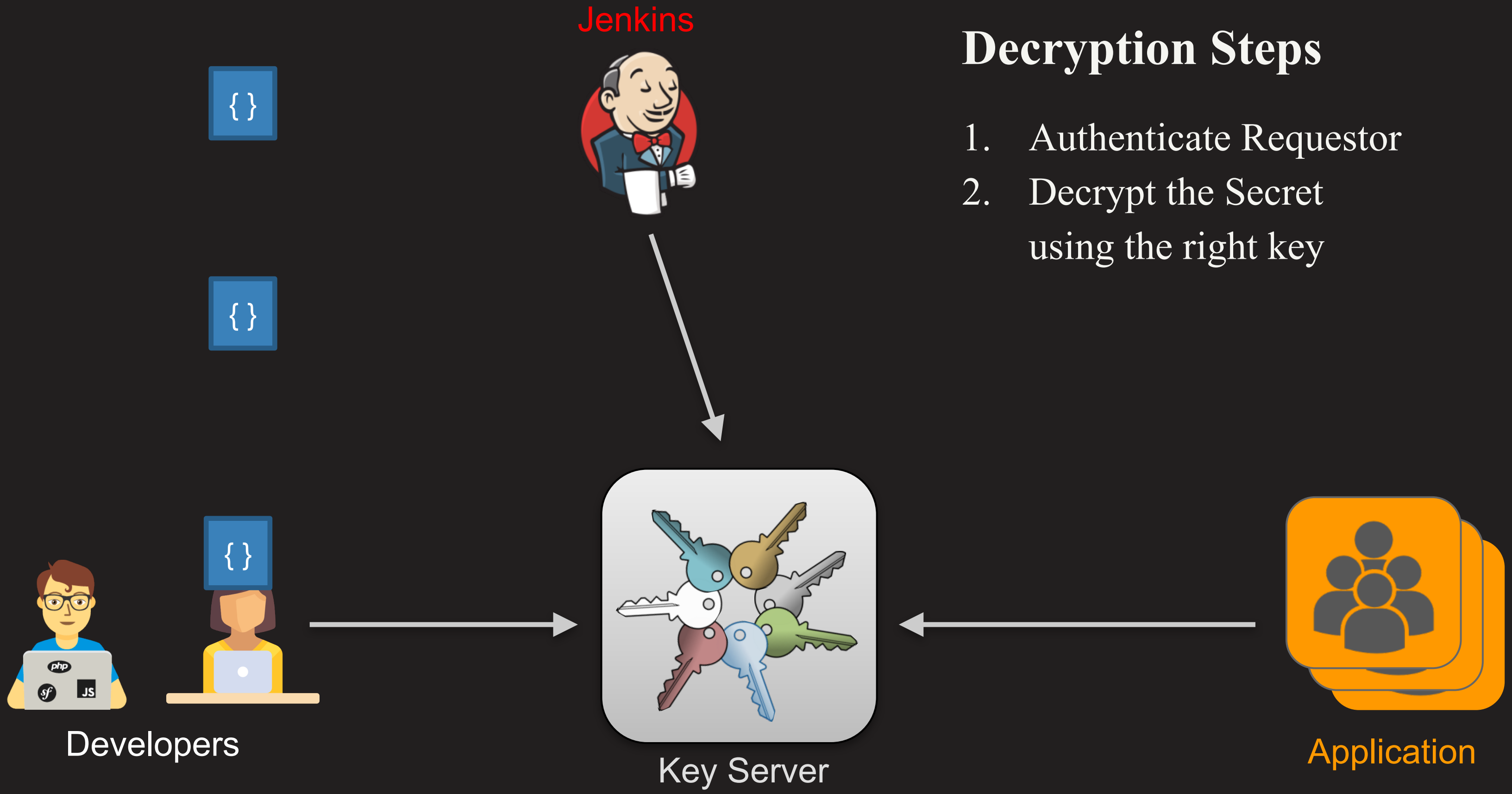


```
public DBResult getEmployeeData() {

  String host = "database.example.com";
  String username = "operator";
  String encPassword = "EBEABKihxG01UEe50JXpazdhUH5ijuL6a15VmIRBZi+eizn6+IXJTcKo7";
  String password = decrypt(encPassword);
  String query = "SELECT * from employee;";

  DBConnection connection = new DBConnection(host, username, password);
  connection.execute(query);

  // Format the output and return
}
```

# Story at Netflix



# Story at Netflix



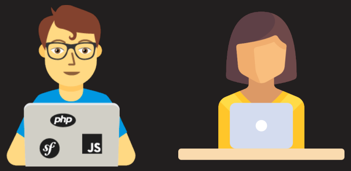
## Decryption Steps

1. Authenticate Requestor
2. Decrypt the Secret using the right key

# Step 1: Authenticate Requestor

## Requestor’s Identity

1. Users
   - mTLS or Oauth
   - Identity Bootstrapped thru User Identity Provider



Developers

2. Applications (AWS VMs/Containers)
   - mTLS
   - Identity Bootstrapped thru AWS Metadata service

Jenkins

Application

# Step 1: Authenticate Requestor

## Identity Bootstrapping for Applications (AWS VMs)

- Use AWS Metatdata Service as Root-of-Trust

  http://169.254.169.254/latest/dynamic/instance-identity/rsa2048

MIAGCSqGSIb3DQEHAqCAMIACAQExDzANBglghkgBZQMEAgEFADCABgkqhkiG9w0BBwGggCSABIIBsnsKICAiZGV
2cGF5UHJvZHVjdENvZGVzIiA6IG51bGwsCiAgInByaXZhdGVJcCIgOiAiMTAwLjY2LjQzLjI0NCIsCiAgImF2YWlsYWJpb
Gl0eVpvbmUiIDogInVzLWVhc3QtMWUiLAogICJhY2NvdW50SWQiIDogIjE3OTcyNzEwMTE5NCIsCiAgInZlcnNpb24iIDo
gIjIwMTAtMDgtMzEiLAogICJpbnN0YW5jZUlkIiA6ICJpLTBmODM5MmJjNTk4N2MwOGIxIiwKICAiYmlsbGluZ1Byb2R1Y
3RzIiA6IG51bGwsCiAgImluc3RhbmNlVHlwZSIgOiAibTMuMnhsYXJnZSIsCiAgImltYWdlSWQiIDogImFtaS1lNjBjOTVmM
SIsCiAgInBlbmRpbmdUaW1lIiA6ICIyMDE2LTA4LTEyVDIyOjI4OjA5WiIsCiAgImFyY2hpdGVjdHVyZSIgOiAieDg2XzY0Ii
wKICAia2VybmVsSWQiIDogbnVsbCwKICAicmFtZGlza0lkIiA6IG51bGwsCiAgInJlZ2lvbiIgOiAidXMtZWFzdC0xIgp9AAA
AAAAAMYIB/zCCAfsCAQEwaTBcMQswCQYDVQQGEwJVUzEZMBcGA1UECBMQV2FzaGluZ3RvbiBTdGF0ZTEQMA
4GA1UEBxMHU2VhdHRsZTEgMB4GA1UEChMXQW1hem9uIFdlYiBTZXJ2aWNlcyBMTEMCCQCxacxAFVmkGTANBg
lghkgBZQMEAgEFAKBpMBgGCSqGSIb3DQEJAzELBgkqhkiG9w0BBwEwHAYJKoZIhvcNAQkFMQ8XDTE2MDgxMjIy
MjgyM1owLwYJKoZIhvcNAQkEMSIEIOPIgCnFPPH6XRU4IJt3Vt2PhdbTthPhZUdqtEQhOf0YMA0GCSqGSIb3DQEBA
QUABIIBAFiNhtqwvLEAGwoLgqjE2lrnoFl0LFPSuduCV9Rh8X6xcw2vCPVwj2JP4jvMao0N1mkFiRY2m+URlBrZr+Tsxg
QWu1z/yGNaJ/ausBzlNuyBqNwQiHTSF6X8GtUH2tuBXN2jYsfHIU72xX1XD4njoCBxZz3XRC3Ltyl6yvPBzZdtKYcqmPs
3Jx43JnqvnauZBUARYZX20WE0TdHa+KPHY2nbMPLkIkN/3TIstUvx9YfeCXT2lwVNRF6BYv+MqM2+cWSbt3arEK7gU/
B0cDETmiaIlBHfNb51etQ2/3kOxuOqBx17hhxD9k25qKjJbxDiNb3UBqVy56yHfjj/BEpkt04AAAAAAAA=

# Step 1: Authenticate Requestor

AWS Metadata Service Output

```
{
 “data” : {
          "devpayProductCodes" : null,
          "privateIp" : "100.66.43.244",
          "availabilityZone" : "us-east-1e",
          "
          "
          "
          "
          "
          "
          "
          "kernelId" : null,
          "ramdiskId" : null,
          "region" : "us-east-1”
 },
 “signature” : “DqktfKuv2r8j …..
                    JqIYWS0aMoFjZhYMg4G”
}
```

AWS describeInstance Output

```
{
   architecture: "x86_64",
   class: "com.amazonaws.services.ec2.model.Instance",
   imageId: "ami-e60c95f1",

    {
      aws:autoscaling:groupName: ”infocrypt-v002",
    }
   ],
   vpcId: "vpc-12345"
}
```

Details on this in

1. Enigma 2017 Conference
2. Future:NET 2017 Conference

# Step 2: Decrypt

## Requirement

Each Group of User(s) and Application(s) MUST have at least one unique key

For e.g.

$K_1$ for $G_1$ = [ Alice , Bob , Application$_1$ , Jenkins$_1$ ]

$K_2$ for $G_2$ = [ Eve , Application$_2$ , Application$_3$ ]

…

# Let’s talk scale

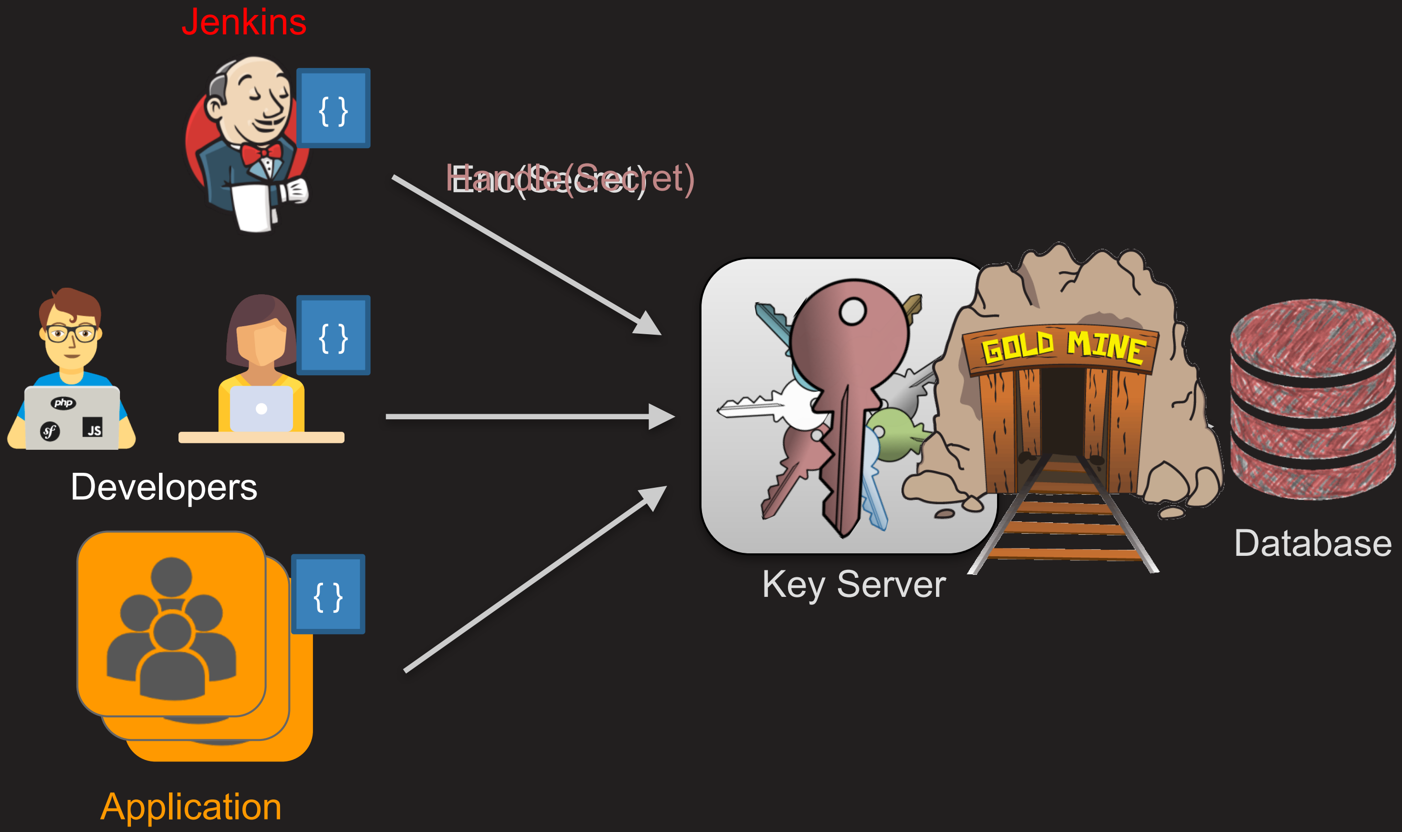
If we have $N$ Users and $M$ Applications, maximum # of groups is …

$$\sum_{k=1}^{M+N} \binom{M+N}{k} = 2^{(M+N)} - 1$$

For $N = 10$ and $M = 10$, the number is 1 Million+

For $N = 12$ and $M = 12$, the number is 16 Million+

# But, why complicate?



Jenkins

Enc(Secret)

Handle(Secret)

Developers

Application

Key Server

Database

# Define our Goals

## Goal

- Secret MUST NOT ever be readable in clear except for the creator and intended consumers (Not even the Decryption Service)

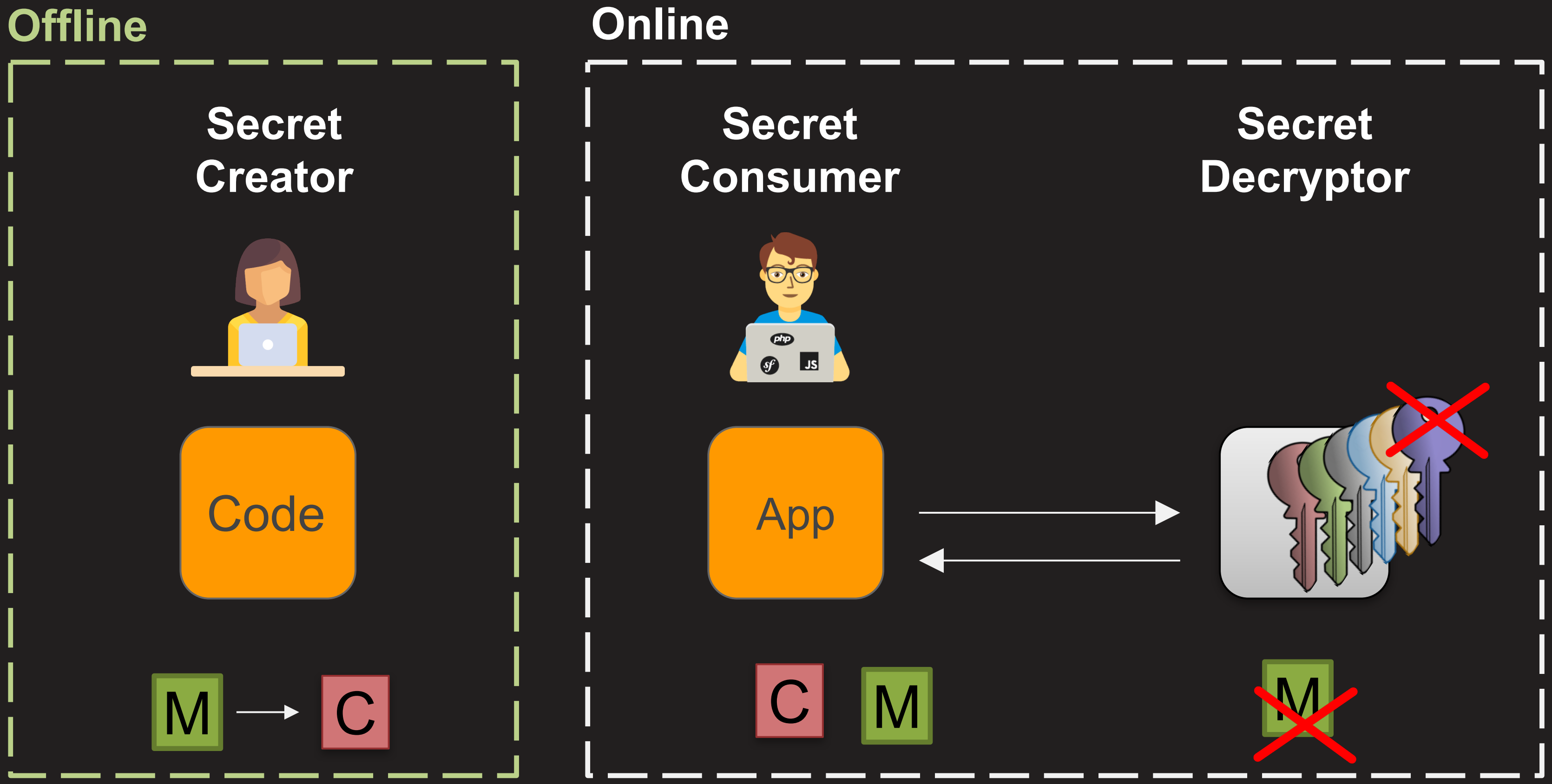
## Stretch Goals

- Offline Encryption of Secrets SHOULD BE supported
- Decryption Service's ability to observe usage pattern SHOULD BE limited

## Constraints

- # of Keys should scale
- # of Request should scale

# Goals - Visually



Offline

Secret Creator

Code

M → C

Online

Secret Consumer

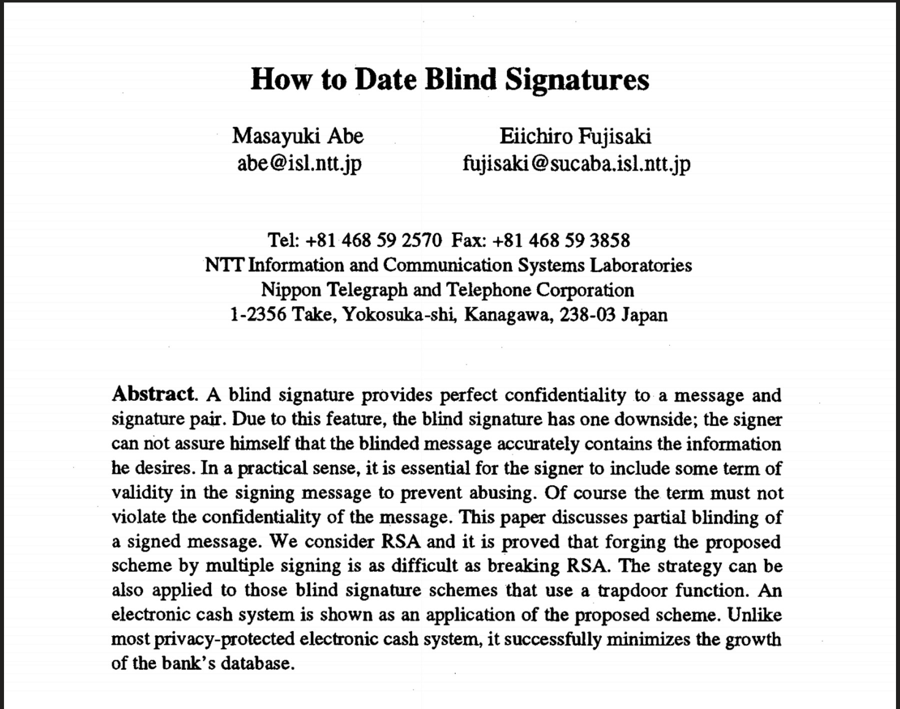
App

C M

Secret Decryptor

M

# Our Solution - Inspiration



## How to Date Blind Signatures

Masayuki Abe
abe@isl.ntt.jp

Eiichiro Fujisaki
fujisaki@sucaba.isl.ntt.jp

Tel: +81 468 59 2570 Fax: +81 468 59 3858
NTT Information and Communication Systems Laboratories
Nippon Telegraph and Telephone Corporation
1-2356 Take, Yokosuka-shi, Kanagawa, 238-03 Japan

**Abstract**. A blind signature provides perfect confidentiality to a message and signature pair. Due to this feature, the blind signature has one downside; the signer can not assure himself that the blinded message accurately contains the information he desires. In a practical sense, it is essential for the signer to include some term of validity in the signing message to prevent abusing. Of course the term must not violate the confidentiality of the message. This paper discusses partial blinding of a signed message. We consider RSA and it is proved that forging the proposed scheme by multiple signing is as difficult as breaking RSA. The strategy can be also applied to those blind signature schemes that use a trapdoor function. An electronic cash system is shown as an application of the proposed scheme. Unlike most privacy-protected electronic cash system, it successfully minimizes the growth of the bank's database.

Abe M., Fujisaki E., *How to date blind signatures*, ASIACRYPT '96. LNCS, Vol 1163. Springer, Berlin.

# Our Solution - Setup

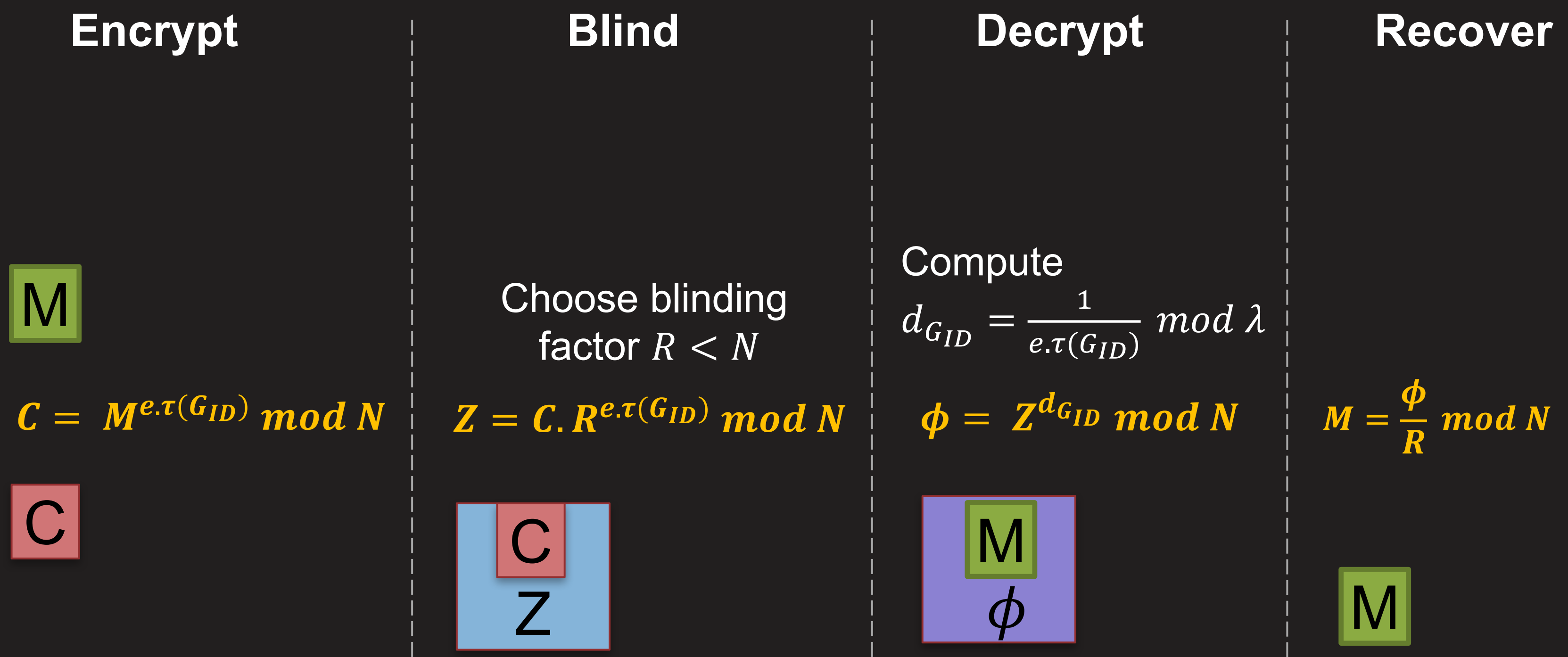
Let $G_{ID}$ be group ID with length $(k-2)$ bits.
Let $\boldsymbol{\tau(G_{ID}) = 2^{k-1} + 2G_{ID} + 1}$

That is, $\tau(G_{ID_i})$ does not divide $\tau(G_{ID_j})$ where $i \neq j$

Choose two large primes $p$ and $q$ such that
$\boldsymbol{s_i \nmid \lambda}$ for all prime $s_i$ $(3 \leq s_i \leq 2^{k-1} - 1)$
Where $\lambda$ is the LCM of $p-1$ and $q-1$

Choose public prime exponent $\boldsymbol{e \geq 2^k - 1}$
Compute $d$ such that $ed = 1 \; mod \; \lambda$

# Our Solution – In Action



## Encrypt

$C = M^{e.\tau(G_{ID})} \ mod \ N$

## Blind

Choose blinding factor $R < N$

$Z = C.R^{e.\tau(G_{ID})} \ mod \ N$

## Decrypt

Compute

$d_{G_{ID}} = \frac{1}{e.\tau(G_{ID})} \ mod \ \lambda$

$\phi = Z^{d_{G_{ID}}} \ mod \ N$

## Recover

$M = \frac{\phi}{R} \ mod \ N$

# Padding

- OAEP, KEM

- Since the Decryption step is after Authentication, it is not practical for attacker to use it as Decryption Oracle without getting noticed.

# Our Solution vs. Goals

**Goal**

- Secret MUST NOT ever be readable in clear except for the creator and intended consumers (Not even the Decryption Service)

✓ Blind Decryption Service behind Authentication

**Stretch Goals**

- Offline Encryption of Secrets SHOULD BE supported
- Decryption Service’s ability to observe usage pattern is limited

✓ Asymmetric system provides offline Encryption and Blinding limits Decryption Service’s visibility

**Constraint**

- # of Keys should scale
- # of Request should scale

✓ Stateless system with only 1 private key - Scalable

# Taking it a step further

- $G_{ID}$ is just a positive integer of $(k-2)$ bits
- It does not have to look like

  $G_1$ = [ Alice , Bob , $\text{Application}_1$ , $\text{Jenkins}_1$ ]
- Instead, it can look something like

  $G_1$ = <signed policy document with ID>

# Other Constructions

- Aware of

  Jaimee Brown, Juan Manuel Gonzalez Nieto, and Colin Boyd. *Efficient CCA-Secure Public-Key Encryption Schemes from RSA-Related Assumptions*, pages 176–190. Springer BerlinHeidelberg, Berlin, Heidelberg, 2006.

- Other suggestions are welcome !

# Next Steps

Keep looking for better underlying scheme

- Better Provable Security Guarantees
- Multi-party Blind Decryption
- PQ-resistant scheme

# Resources

- Enigma 2017 Talk on Bootstrapping Identities https://www.youtube.com/watch?v=15H5uCj1hlE

- Future:NET 2017 Talk on Application Identity https://www.youtube.com/watch?v=g2efknf-HXQ

- Abe M., Fujisaki E. (1996) *How to date blind signatures*. In: Kim K., Matsumoto T. (eds) Advances in Cryptology — ASIACRYPT '96. ASIACRYPT 1996. Lecture Notes in Computer Science, vol. 1163. Springer, Berlin, Heidelberg https://doi.org/10.1007/BFb0034851

# Thank you.

*(we are hiring)*

mmehta@netflix.com

NETFLIX