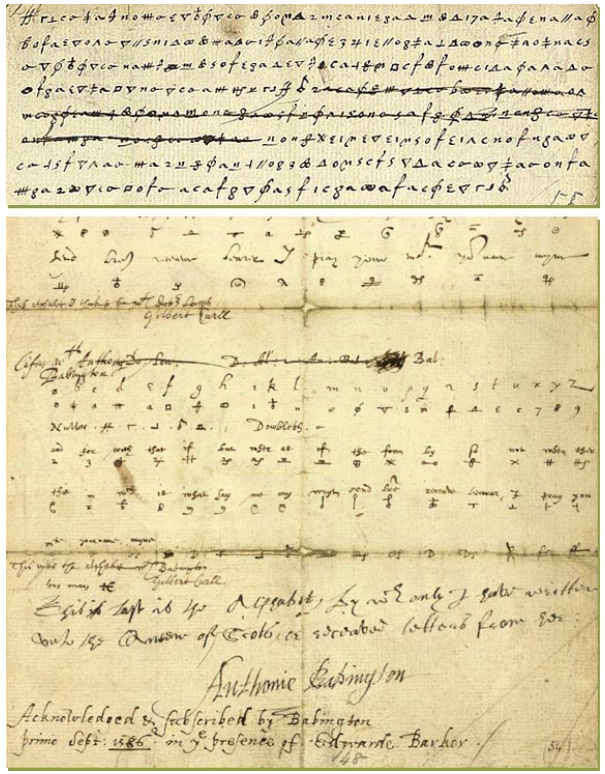
Securely Encrypting and Decrypting Data

Josh Cummings
PRINCIPAL SOFTWARE ENGINEER

@jzheaux blog.jzheaux.io



In 1586, Queen Mary hatches a plot, sends encoded messages

Thomas Phelippes intercepts and decodes messages

And he even adds his own instructions into the message

Mary is found out, charged with treason, and beheaded

“Few false ideas have more firmly gripped the minds of so many intelligent men than the one that, if they just tried, they could invent a cipher that no one could break.”

David Kahn

Goals

Cipher

Secrecy,
Authenticity,
Integrity

Google Tink

Signing vs. Encryption

Signing	Encryption
Integrity	Secrecy
Symmetric/ Asymmetric	Symmetric/ Asymmetric
Irreversible	Reversible

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding");
cipher.init(Cipher.ENCRYPT_MODE, key, spec);
byte[] encrypted = cipher.doFinal(plaintext);
```

Cipher

Call getInstance, indicating the algorithm, block cipher, and padding

Initialize with key, similar to initSign and initVerify

Some algorithms require more inputs, via spec classes

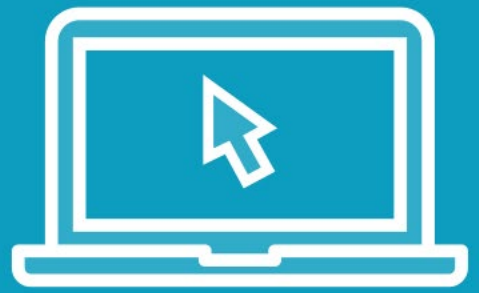
byte[] update(byte[])

```
InputStream is = new CipherInputStream(
      new Base64InputStream(
            request.getInputStream()), cipher);
// or
baos.write(cipher.update(somedata));
baos.write(cipher.update(moredata));
baos.write(cipher.doFinal());
```

void update(byte[])

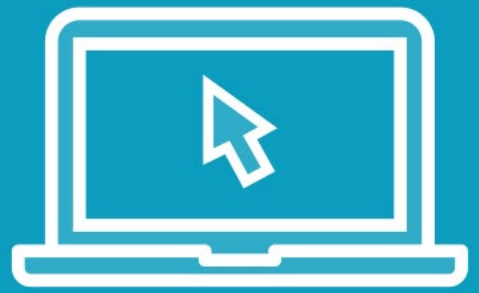
~~byte[] body = readFromRequest();~~
~~cipher.update(body);~~
~~cipher.doFinal();~~

Demo



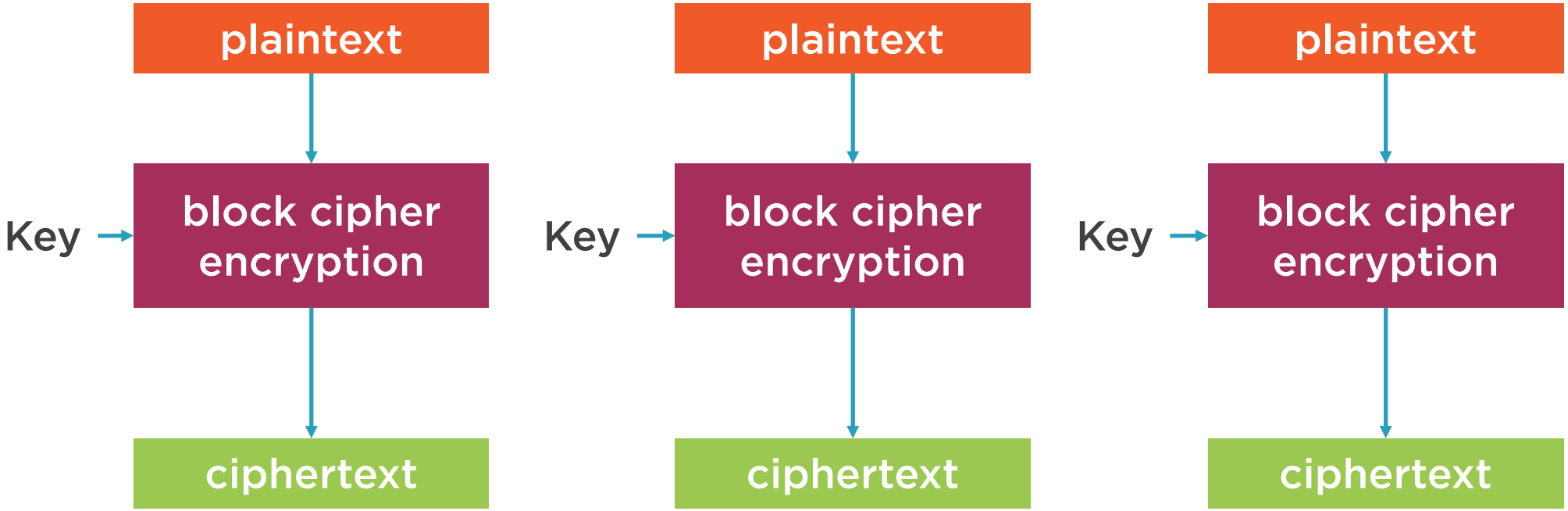
Cipher - AES

Demo



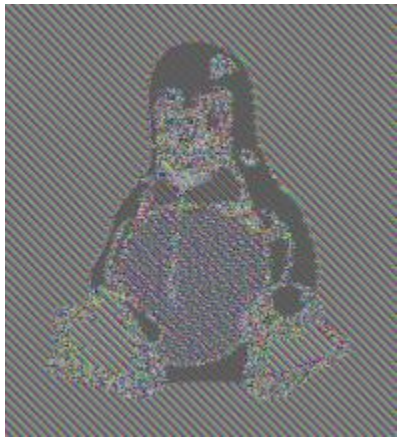
VA

AES Overview



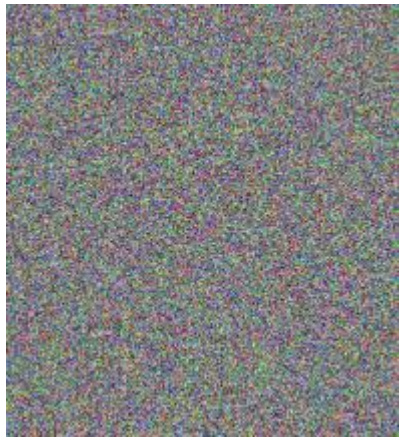
Block Cipher Comparison

No Encryption



Electronic Codebook

First-generation AES



Cipher Block Chaining

Uses previous block as input to next block

Images courtesy Larry Ewing, lewing@isc.tamu.edu, using GIMP

`Cipher.getInstance("AES")`
== weaker block cipher

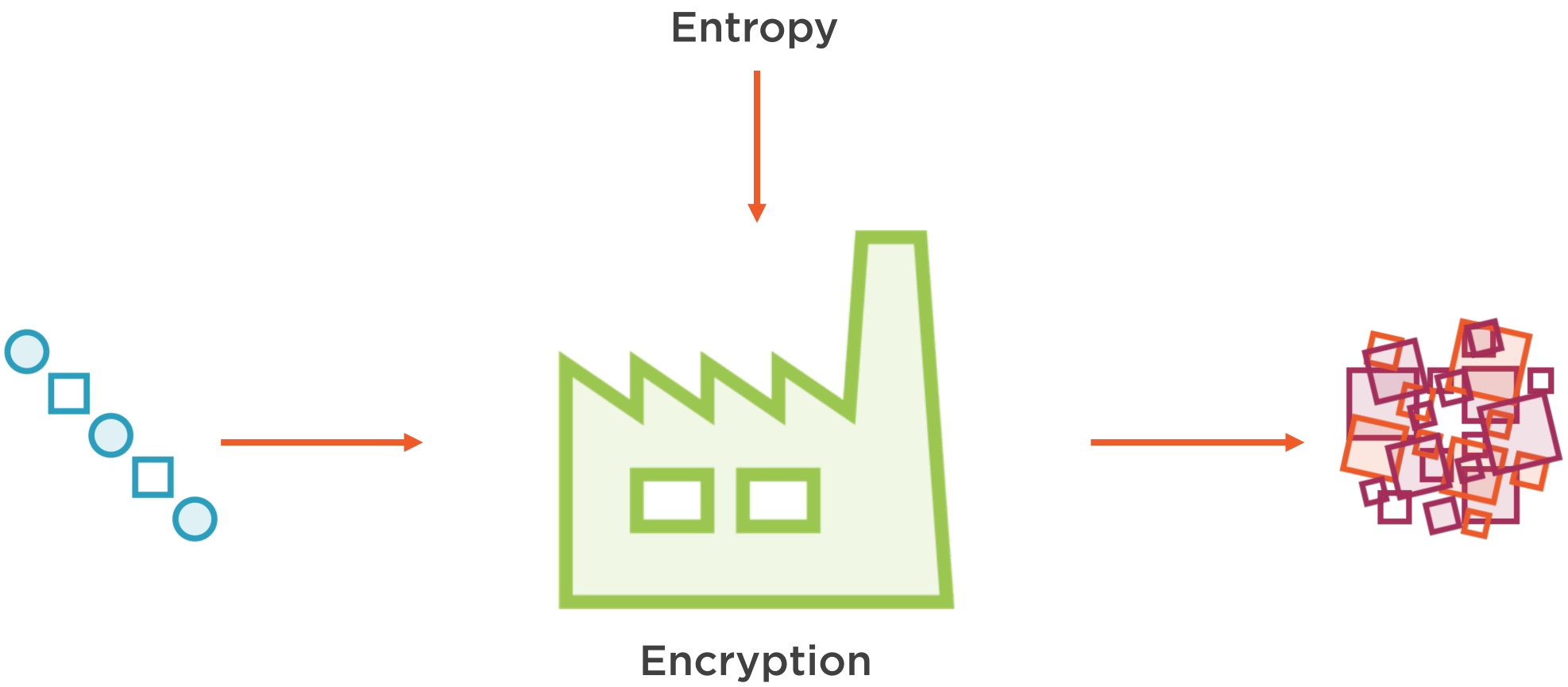
```
Cipher c = Cipher.getInstance
    ("AES/CBC/PKCS5Padding");
```

- Always supply block cipher, even when using ECB
- CBC requires an initialization vector

```
byte[] iv = new byte[16];

SecureRandom rand =
    new SecureRandom();

rand.nextBytes(iv);

c.init(Cipher.ENCRYPT_MODE, key,
    new IVParameterSpec(iv));
```

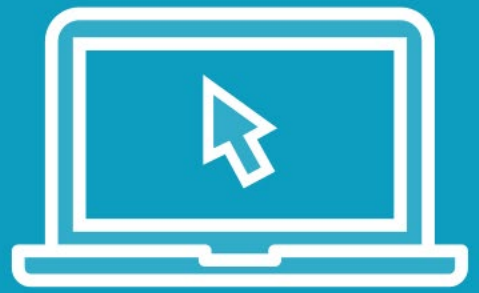
- For CBC, the initialization vector is the same size as the blocks
- Generate a new one each time, like salts in passwords

Random Inputs Introduce Entropy



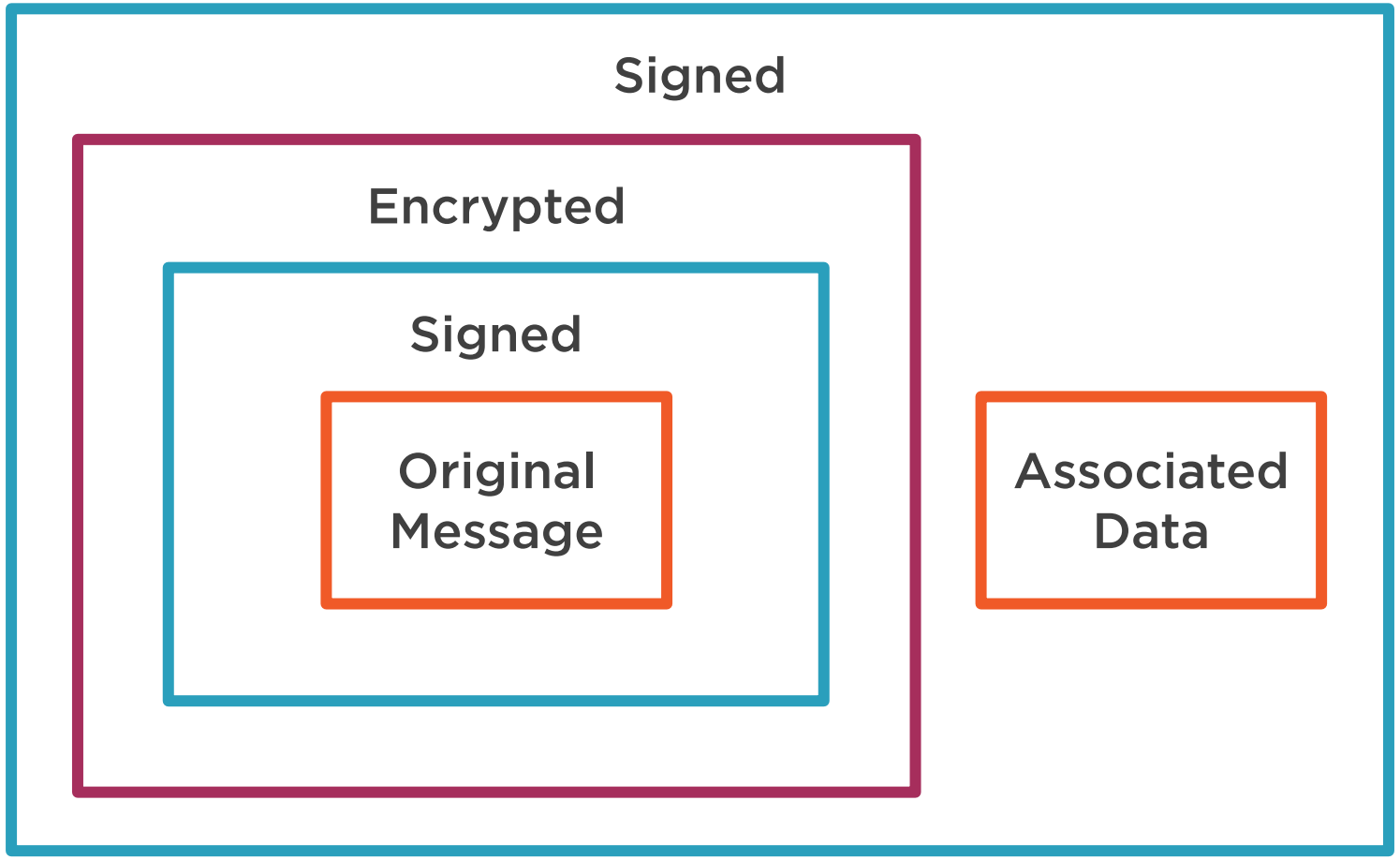
How will the recipient know the IV?

Demo

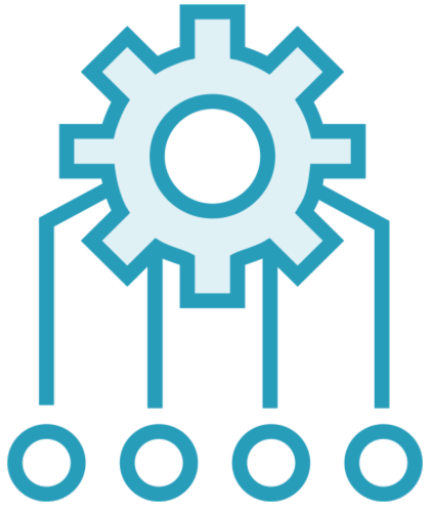


With IV

Sending Associated Data



Can We Do Better Than CBC?



Can’t be parallelized

Needs additional signing

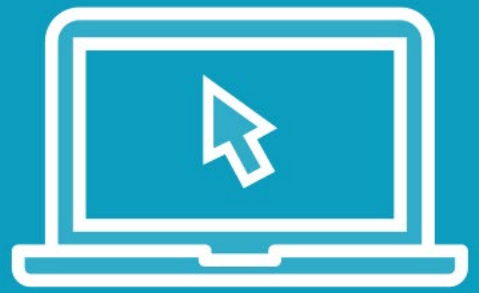
```
Cipher c = Cipher.getInstance
    ("AES/GCM/NoPadding");

byte[] iv = entropy();

c.init(Cipher.ENCRYPT_MODE, key,
    new GCMParameterSpec(iv));

c.updateAAD(version);

byte[] ciphertext =
    c.doFinal(plaintext);

// now, to decrypt

c.updateAAD(version)

byte[] plaintext =
    c.doFinal(ciphertext);
```

- GCM can be parellized
- It also adds a MAC automatically

- Add associated data to the MAC
- Encrypt as normal

- If the MAC fails, decryption fails

Demo



GCM

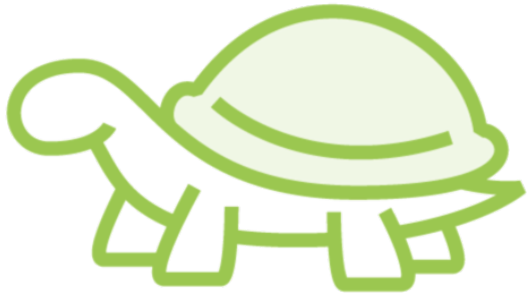
Public-key Encryption

	Sender	Recipient
Signing		
Encryption		

Asymmetric Encryption

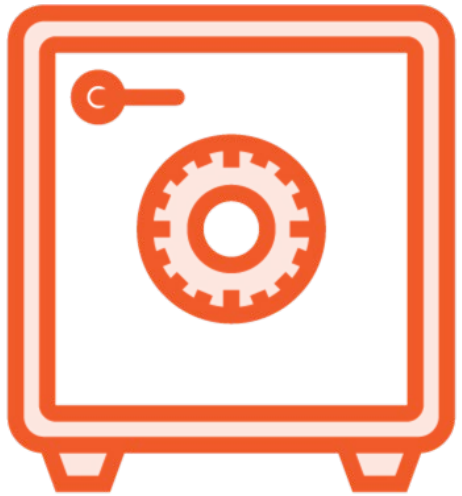
No Authenticity

You can't know who encrypted the payload



Slower

Taxes computational resources, generally impractical for large messages



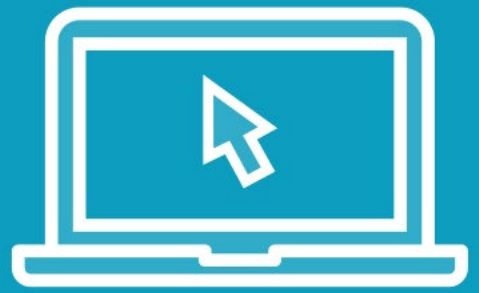
Convenient

You don't need to send anyone your secrets

```
publicKey ←
    recipientPublicKey();

secretKey ← generate();

encryptedKey ←
    encrypt(secretKey,publicKey);

aad ← encryptedKey

ciphertext ←
    encrypt(plaintext,
        secretKey, aad);

send(ciphertext,encryptedKey);
```

- Generate a random symmetric key
- Encrypt the symmetric key with the recipient's public key
- Encrypt plaintext with secret key, including the encrypted symmetric key as associated data
- Send ciphertext and encrypted key

Demo

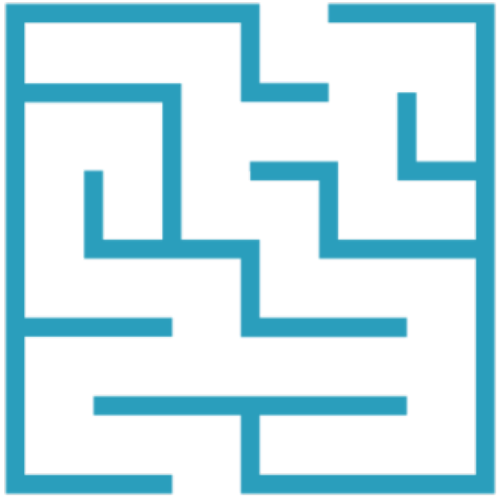


Hybrid Encryption

Where Can We Improve?

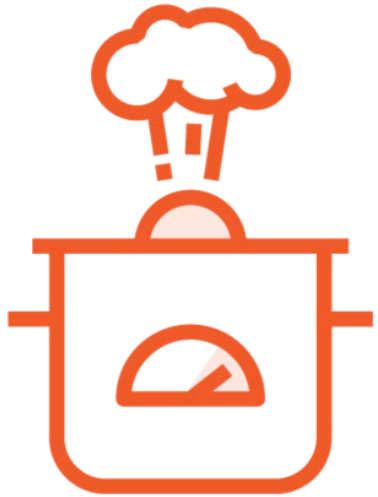
Weak Defaults

ECB by default, key sizes



Unclear APIs

Wait, I have to call `init`, too?



Boilerplate

It takes a lot of code to say a little

Show me the library!

```
HybridConfig.register(); // initialize the Hybrid primitive

KeyTemplate implementation = // use this Hybrid algorithm
HybridKeyTemplates.ECIES_P256_HKDF_HMAC_SHA256_AES128_GCM;
```

Google Tink

Cryptographic primitives, one for each type of cryptographic operation

Multiple implementations for each primitive

KeysetHandles

```
KeysetHandle handle =
    KeysetHandle.generateNew(implementation);
```

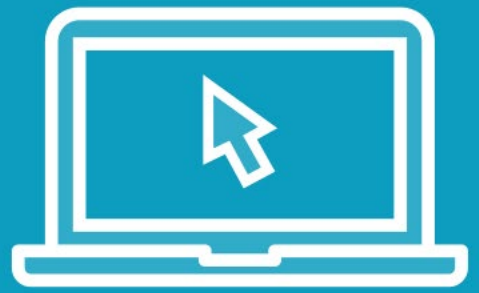
Built-in Key Rotation

No working directly with keys

```
HybridConfig.register();

KeysetHandle handle =
    CleartextKeysetHandle
        .read(keysetReader);


PublicKeysetHandle pub =
    handle.getPublic…

HybridEncrypt encrypt =
    HybridEncryptConfig
        .getPrimitive(pub);

byte[] ciphertext =
    pub.encrypt(plaintext);
```

- Initialize primitive, one-time
- Load key pair from the file system or remotely (or generate with `generateNew`)
- Load the appropriate primitive; hybrid encryption with a public key, in this case
- Perform encryption – look Ma, no keys!

Demo



Hybrid with Tink

Encryption and Decryption

Encryption is about secrecy - remember entropy

Cipher is Java's API for encryption

For symmetric encryption, use AES GCM to additionally achieve integrity and authenticity

Favor hybrid encryption over plain asymmetric encryption

Google Tink addresses some of the JCA's API limitations

SSL