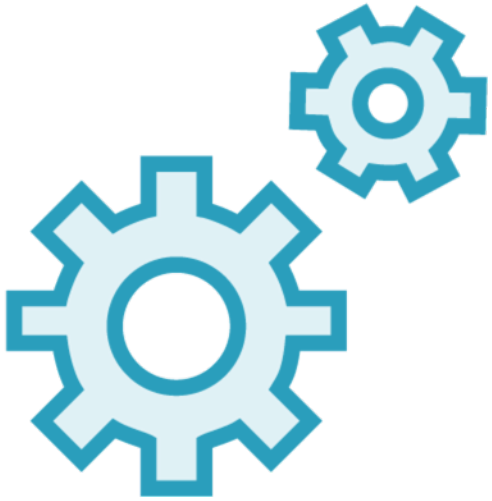
Securely Data Over the Network

Josh Cummings
PRINCIPAL SOFTWARE ENGINEER

@jzheaux blog.jzheaux.io

Challenges with JSSE

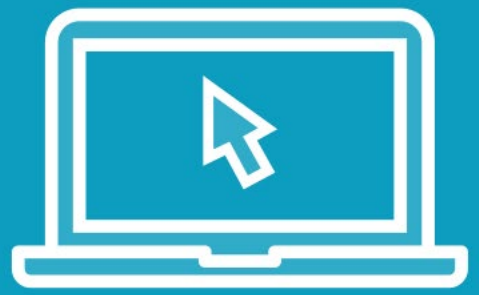


Tricky to Configure

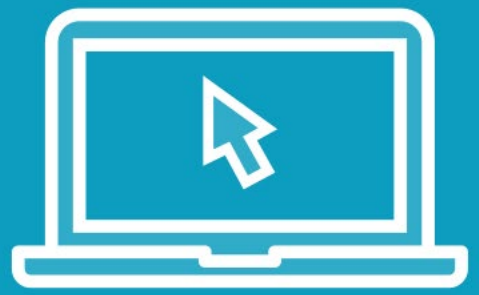
Tricky to Extend

Blissfully Silent When Misconfigured

Demo

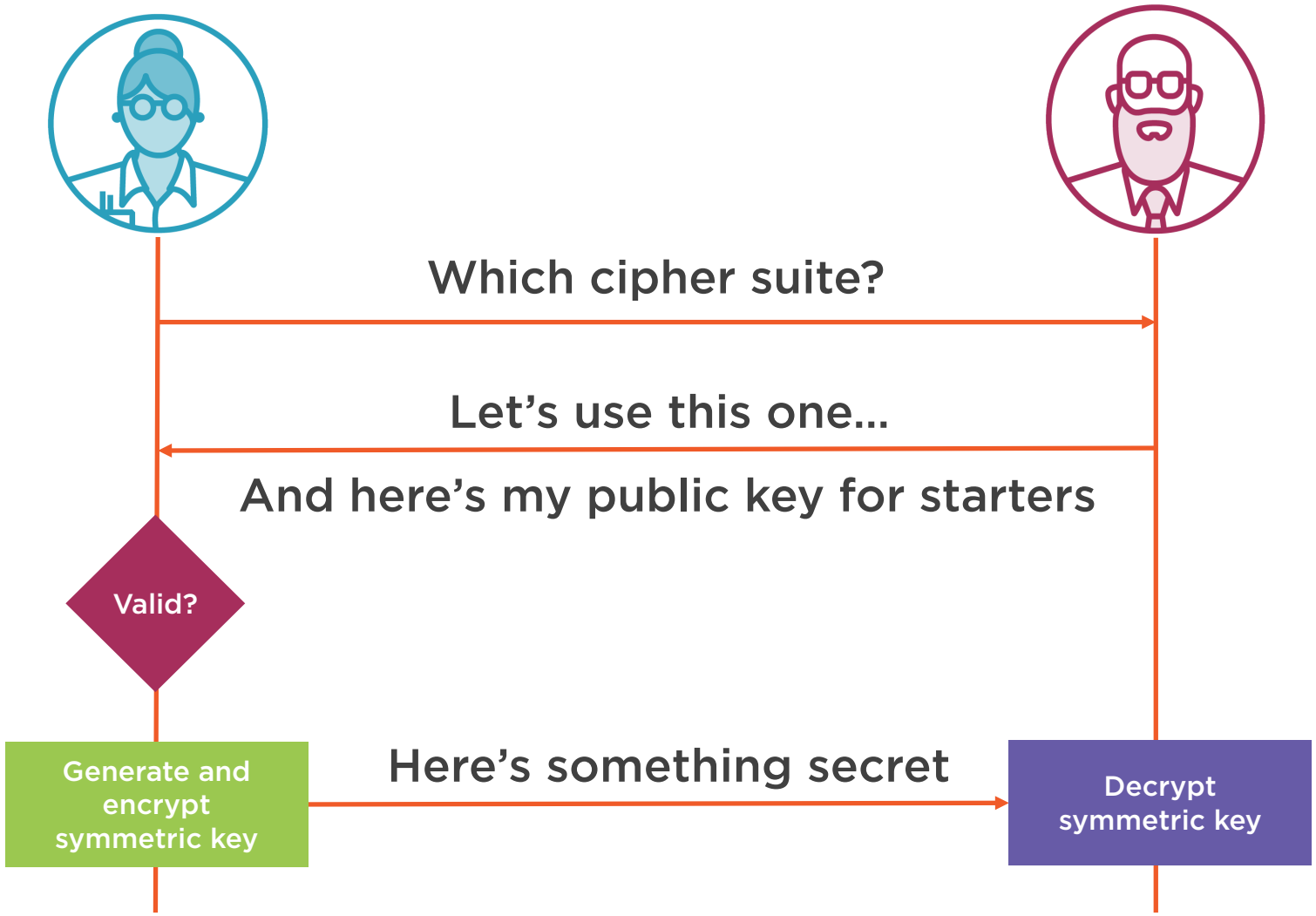


Demo

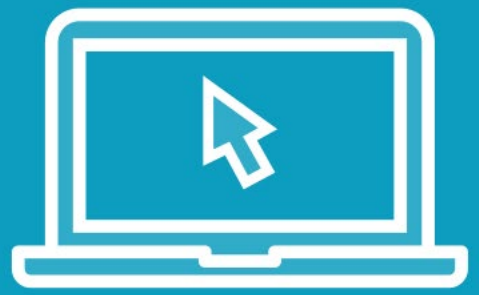


TLS doesn’t encrypt your message with asymmetric keys

RSA TLS Handshake (Abridged)



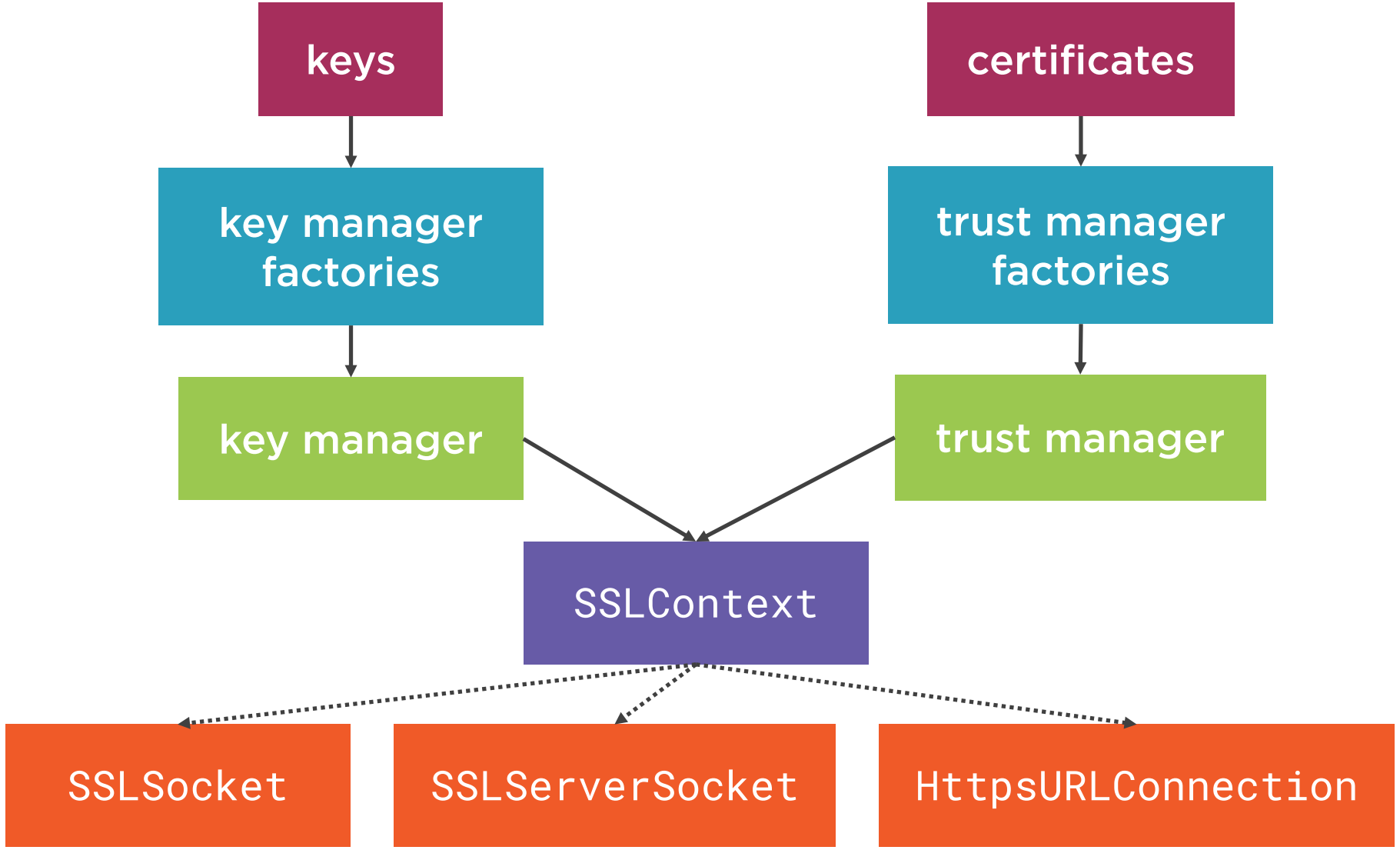
Demo



“Without a correctly configured `SSLContext`, you have nothing”

Will Sargent, https://tersesystems.com

JSSE Overview



```
SSLContext sslContext = SSLContext.getInstance("TLS");
sslContext.init(keyManager, trustManager, random);
```

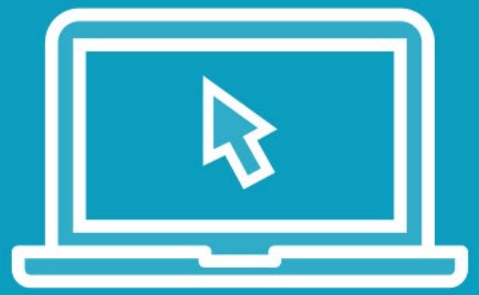
SSLContext

Call getDefault for an immutable instance

Algorithm passed means support versions less-than-or-equal

Just say "TLS" controlling the protocol through other means

Demo



SSLContext Initialization

```
context.init(keys, trusts, random)
```

`KeyManager`

- **Server's private keys and certificates**
- **Takes an array of** `KeyManagers`**...**
- **But it will only use the first element**
- `null` **means no** `KeyManager`

```
KeyManagerFactory factory = KeyManagerFactory
    .getInstance(KeyManagerFactory.getDefaultAlgorithm());
factory.init(keyStore);
```

KeyManager (SunX509)

Uses a single keystore

Expects all private keys to have the same password as thekeystore

Keys can’t be changed out at runtime and it always selects the first key in the store, regardless of validity

```
KeyManagerFactory factory = KeyManagerFactory
    .getInstance("NewSunX509");
factory.init(keyStore);
```

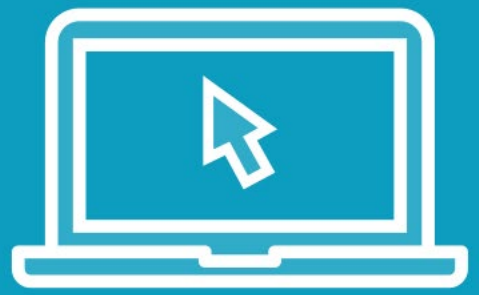
KeyManager (NewSunX509)

Uses a single keystore

Expects all private keys to have the same password as thekeystore

Keys can't be changed out at runtime and it selects the first valid key

Demo



SSLContext Initialization

```
context.init(keys, trusts, random)
```

`TrustManager`

- **Client's trusted authorities**
- **Takes an array of** `TrustManagers`**...**
- **But it will only use the first element**
- `null` **means default** `TrustManager`

Using the Default Truststore

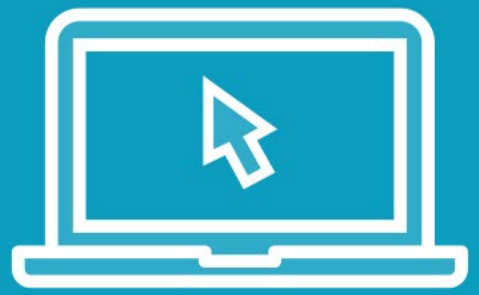
Self-signed Certificates

TLS connections will fail since it's missing from the truststore

Early OpenJDK Versions

OpenJDK versions before February 2018 had an empty default truststore

Demo



Anti-pattern: All-trusting TrustManager

```
new X509TrustManager() {
    public void checkClientTrusted(X509Certificate[] …) {
        // no-op
    }
    public void checkServerTrusted(X509Certificate[] …) {
        // no-op
    }
    public void getAcceptedIssuers() {
        return null;
    }
}
```

Using a Custom Truststore

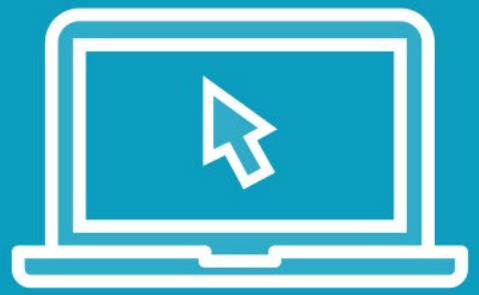
Use the Server's Keystore

If the server is issuing a self-signed certificate, configure the client with the same

Create and Sign with a Local CA

Sign the server's certificate with a locally-generated CA – add the CA to the client's truststore

Demo



A Case for Whitelisting

40+ Cipher Suites

Suites Get Deprecated

Whitelist Cipher Suites

```
SSLSocket sslSocket = (SSLSocket)
    sslSocketFactory.createSocket("localhost", 8443);
```

SSLSocket

The TLS version of Java sockets

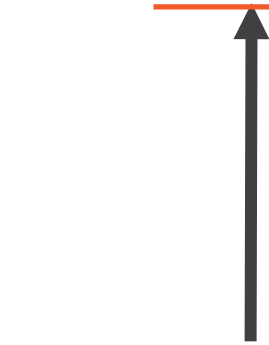
Do full-duplex encrypted communication with a server

```
String[] protocols =
    { “TLSv1.2” };

sslSocket.setEnabledProtocols
    (protocols);

String[] suites =
    { “TLS_DHE_RSA_WITH…” };

sslSocket
    .setEnabledCipherSuites
        (suites);
```

- Whitelist the TLS version
- Whitelist the cipher suite
- Favor suites using emphemeral key exchange, like DHE, and authenticated encryption, like GCM

Cipher Suite Breakdown

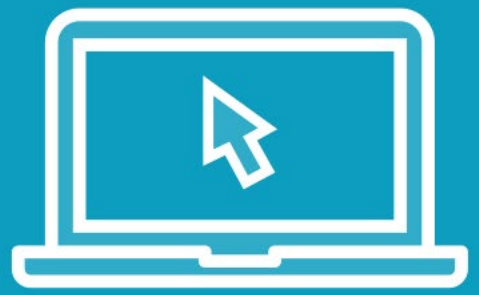
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384



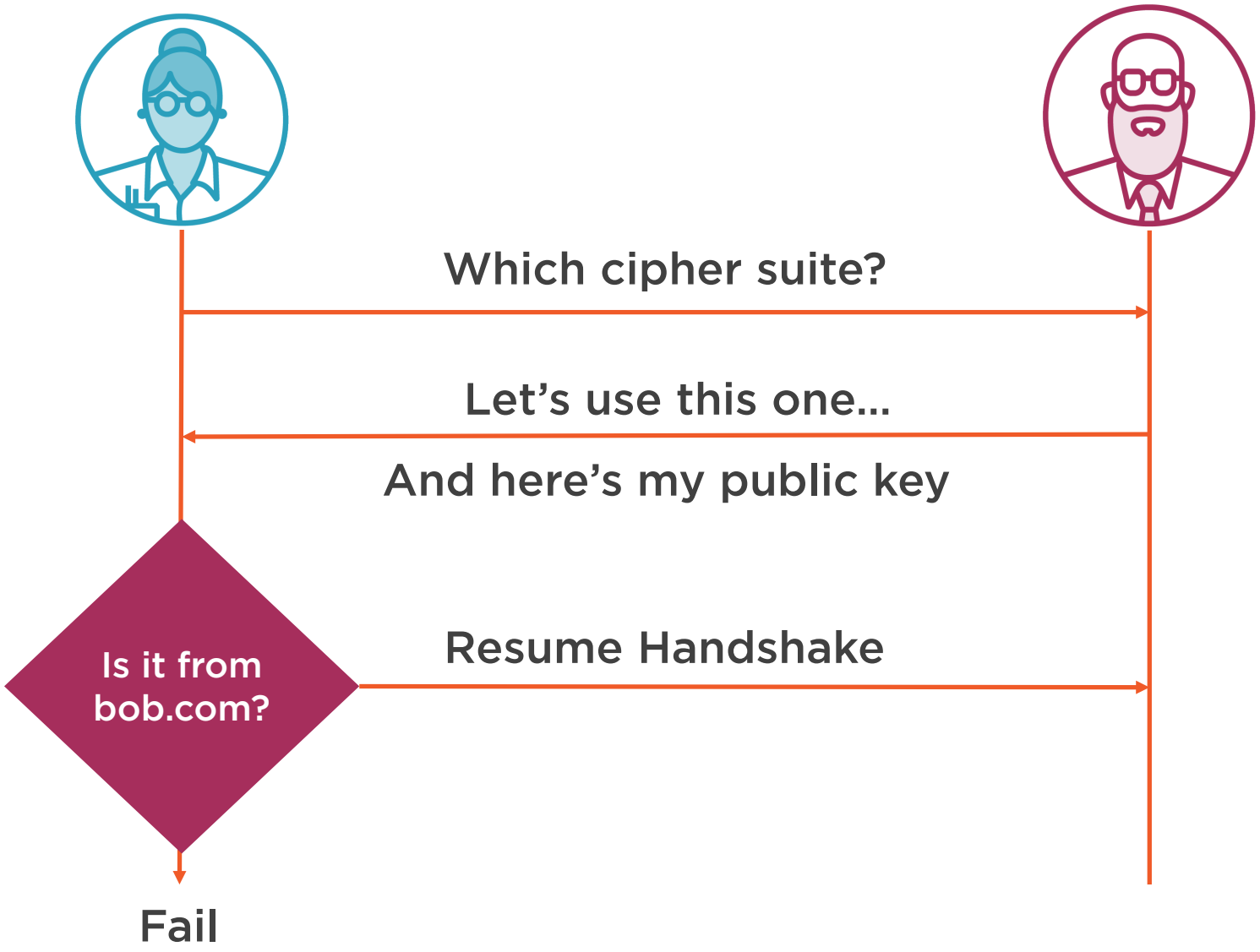
Diffie-Helman
Ephemeral

Authenticated
Encryption

Demo



Hostname Verification



SSLContext has hostname verification turned *off* by default

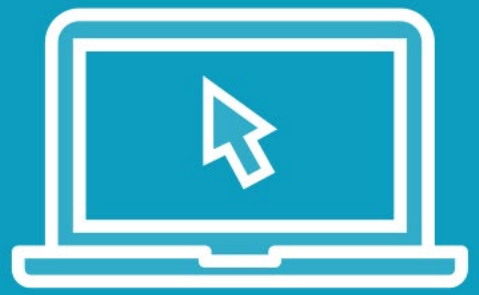
```
SSLSocket socket = sslSocketFactory.createSocket(…);

SSLParamaters parameters = new SSLParameters():
parameters.setEndpointIdentificationAlgorithm(“HTTPS”);
socket.setSSLParameters(parameters);
```

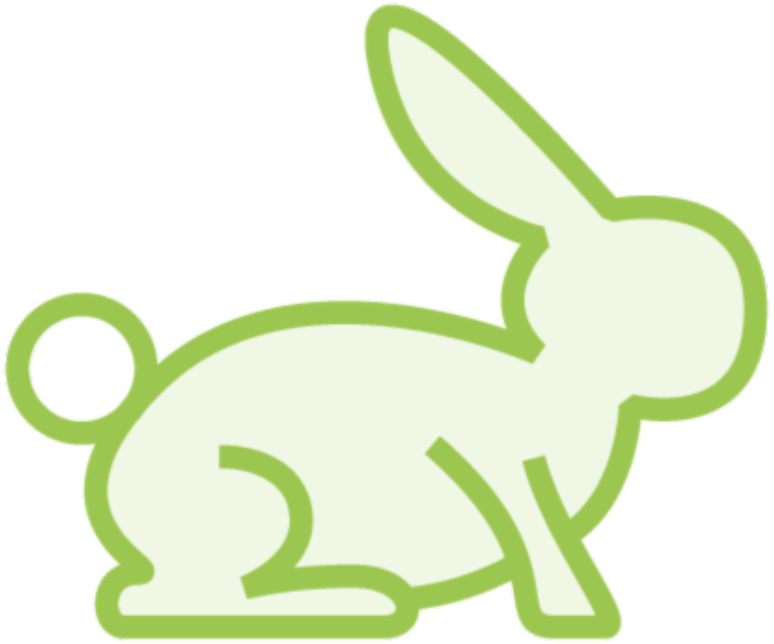
SSLParameters

A simple POJO of parameters, including whether to do hostname verification

Should be configured on the socket instead of the context

Demo





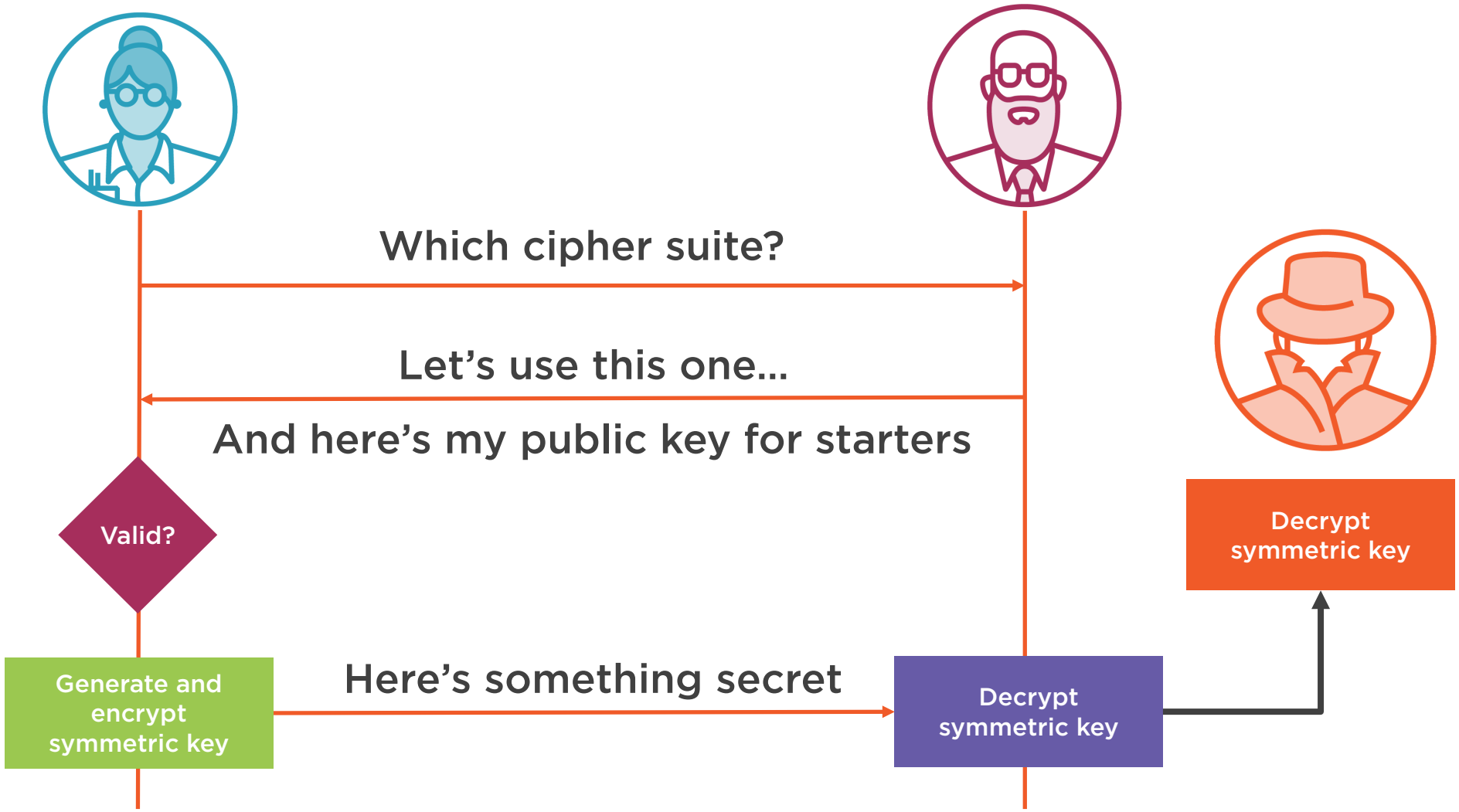
TLS 1.3 just came out

Cleaned up old cipher suites

Simplified the key exchange

It is both faster and more secure than 1.2

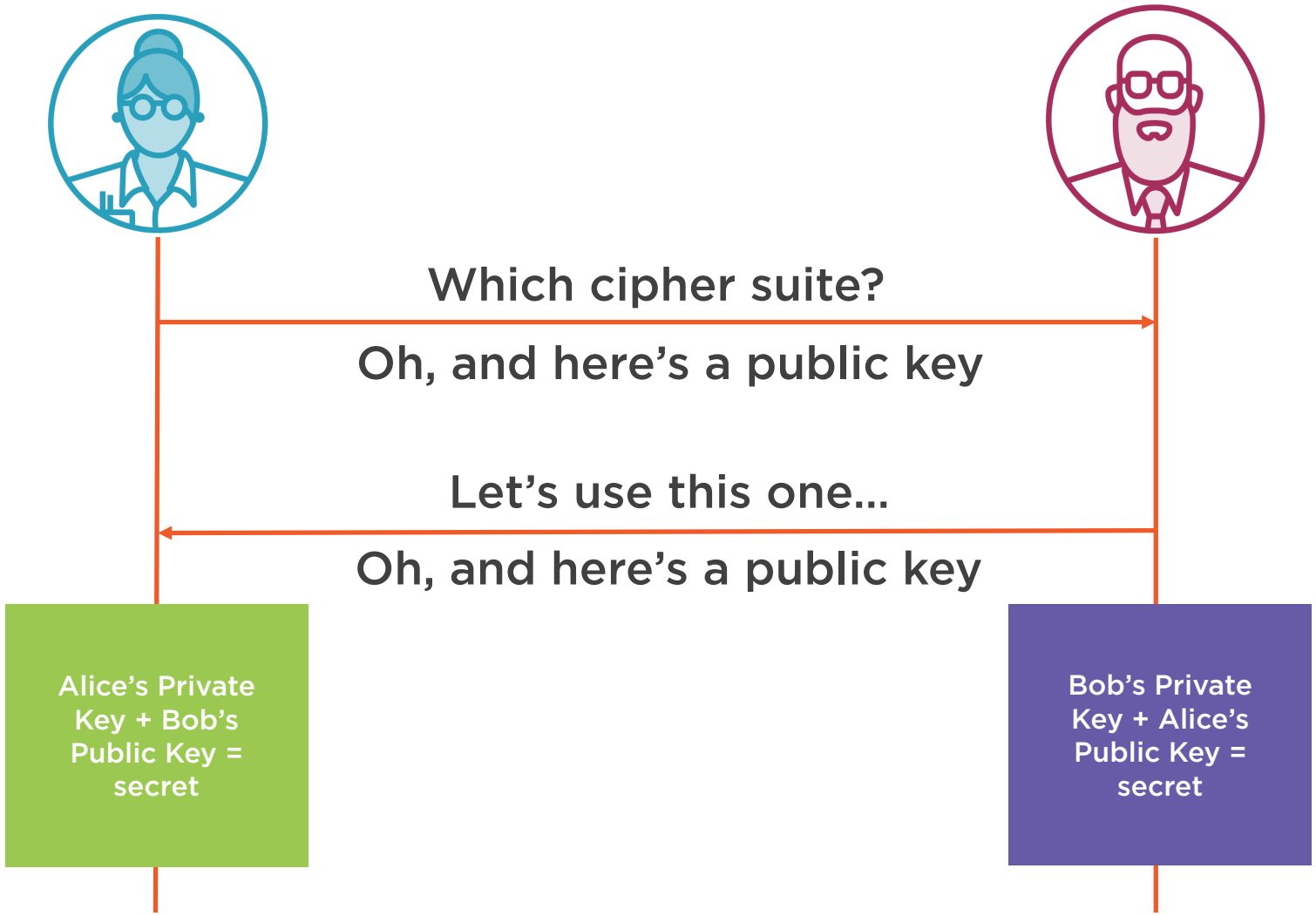
RSA TLS Handshake Drawbacks



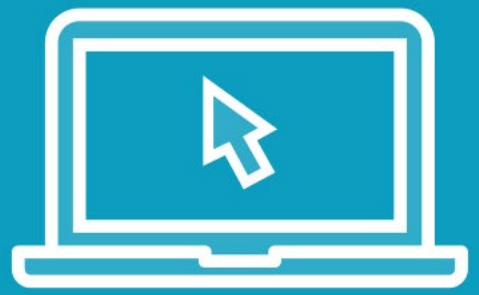
Forward Secrecy

A desirable property of a key exchange algorithm earned from the fact that a different key is used for each key exchange

DHE TLS Handshake (Abridged)



Demo



HttpsURLConnection vs SSLSocket

	HttpsURLConnection	SSLSocket
Keys and Trusts	`conn.setSSLSocketFactory`	`socketFactory.createSocket`
Protocols and Suites	`-Dhttps.protocols` `-Dhttps.cipherSuites`	`socket.setEnabledProtocols` `socket.setEnabledCipherSuites`
Hostname Verification	**included!**	`socket.setSSLParameters`

TLS and JSSE

TLS is a network protocol – 1.3 just released

JSSE supports TLS – JDK 8 defaults to 1.2, 11 supports 1.3

JSSE Pitfalls

KeyManagers prove identity, TrustManagers trust identities

Whitelist protocols and ciphersuites

Switch on hostname verification

Thank you!