

Setting up security and SSL with Liberty in CICS

Matthew Wilson
IBM UK Ltd

November 2018
Session GM



Notes

Every even-numbered slide in this presentation is provided to reflect the intent of the speaker during the session. These notes pages were not presented during the session itself.

To maintain the even-odd numbering scheme, several notes pages are blank, usually for slides which require very little explanation.

Session abstract

In this session we'll look at various different options for security with Liberty in CICS, and how these interact with Java EE and z/OS security models. We'll talk about how to configure Liberty's security features. We'll also take a deep dive into configuring SSL to achieve optimal performance, including an insight into how Java can exploit the latest z14 hardware in order to provide the highest levels of security for data in flight, with minimal disruption to your online workloads.



Notices and disclaimers

- © 2018 International Business Machines Corporation. No part of this document may be reproduced or transmitted in any form without written permission from IBM.
- **U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.**
- Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed “as is” without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.
- IBM products are manufactured from new parts or new and used parts.
In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply.”
- **Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.**
- Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those
- customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.
- References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.
- Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.
- It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Notices and disclaimers continued

- Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**
- The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.
- IBM, the IBM logo, ibm.com and [names of other referenced IBM products and services used in the presentation] are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.
- .

Agenda

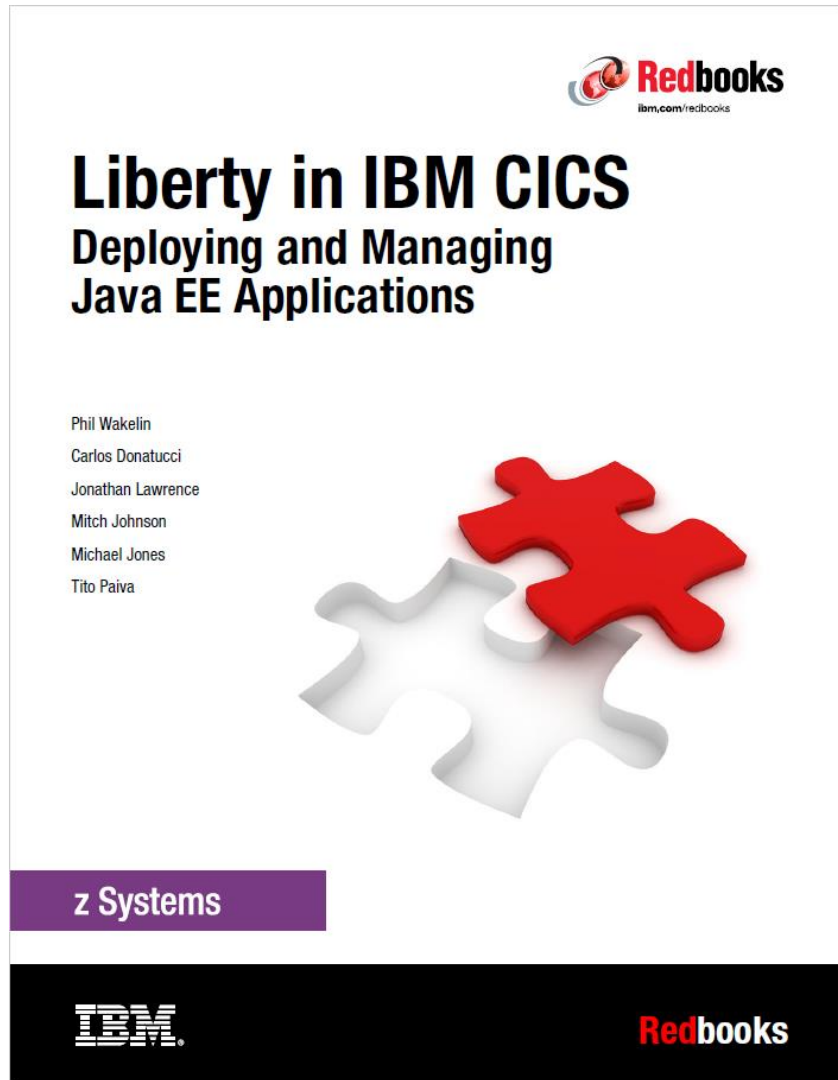
Notes

Topics covered in this session will be:

- Overview of Liberty security options
- Basic security configuration
- SSL: a level-set
- Cipher suites: what's in a name?
- Cipher suites: configuring Liberty
- Performance measurements



IBM Redbooks publication – Liberty in IBM CICS



redbooks.ibm.com/abstracts/sg248418.html?Open

1. Installation and configuration
2. Deploying a web application
3. Link to Liberty
4. Connecting to Db2 by using JDBC
5. Connecting to IBM MQ by using JMS
6. **Configuring Transport Layer Security support**
7. **Securing web applications**
8. Logging and monitoring
9. Port sharing and cloning regions



Notes

The recently-published IBM Redbooks publication “Liberty in IBM CICS: Deploying and Managing Java EE Applications” contains a great deal of information about configuring Liberty in a CICS environment.

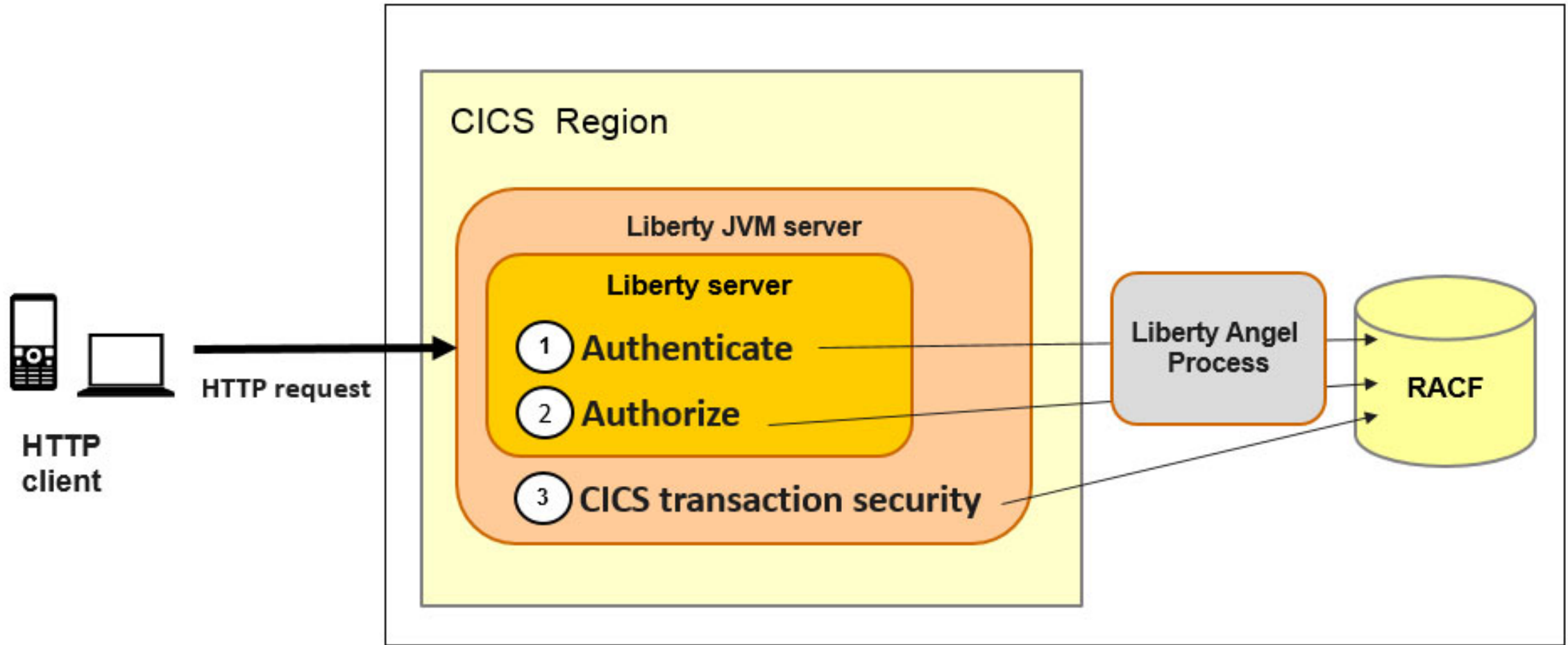
Relevant to this session are chapters 6 and 7 of the publication, which relate to configuring SSL, and securing web applications in Liberty.



Liberty security overview

Notes

CICS and Liberty security flow



Notes

Security choices - authentication

- Basic authentication
- Form
- Client certificate
- Identity token
- Custom



Notes

- We recommend TLS is used with basic authentication whenever possible.
- Client certificate authentication involves mapping of DN in SSL client certificate to RACF user ID
- Identity tokens can be used to provide a user ID that has been authenticated by another party, for example by using OpenID Connect
- Custom authentication methods can be achieved by implementing a Trust Association Interceptor



Security choices - user registries

- One per Liberty instance
- Basic user registry
 - Simple userid / password in XML file
 - Development use only
- System Authorization Facility (RACF, ...)
 - cicsts:security-1.0
- LDAP and distributed identities
 - cicsts:distributedIdentity-1.0

Notes

- To perform security related operations (such as authentication and authorization) in Liberty, you need to configure a user registry.
- Liberty supports federated (i.e. multiple) user registries, but this is not yet supported in CICS.
- If using a basic registry, don't use `cicsts:security-1.0` feature as it will attempt to use the (non-RACF) user ID for task attach
- The LDAP registry can be used with the `cics:distributedIdentity-1.0` feature, which allows the LDAP user ID to be mapped to a SAF user ID for use in CICS.
- Alternatively you can use LDAP by itself, in which case the CICS task runs under the default user ID, or the user ID from the URIMAP



Security choices – privacy & authorization

Decide on privacy

- SSL using keystores in RACF or .jks files

If using CICS transaction/resource security

- Setup URIMAPs to map URIs to TranIDs and allows TCICSTRN/GCICSTRN definitions

If using JEE role based security

- Setup EJBROLE definitions in RACF
- `<safAuthorization id="saf">` in server.xml
- Setup roles in web.xml of Web application

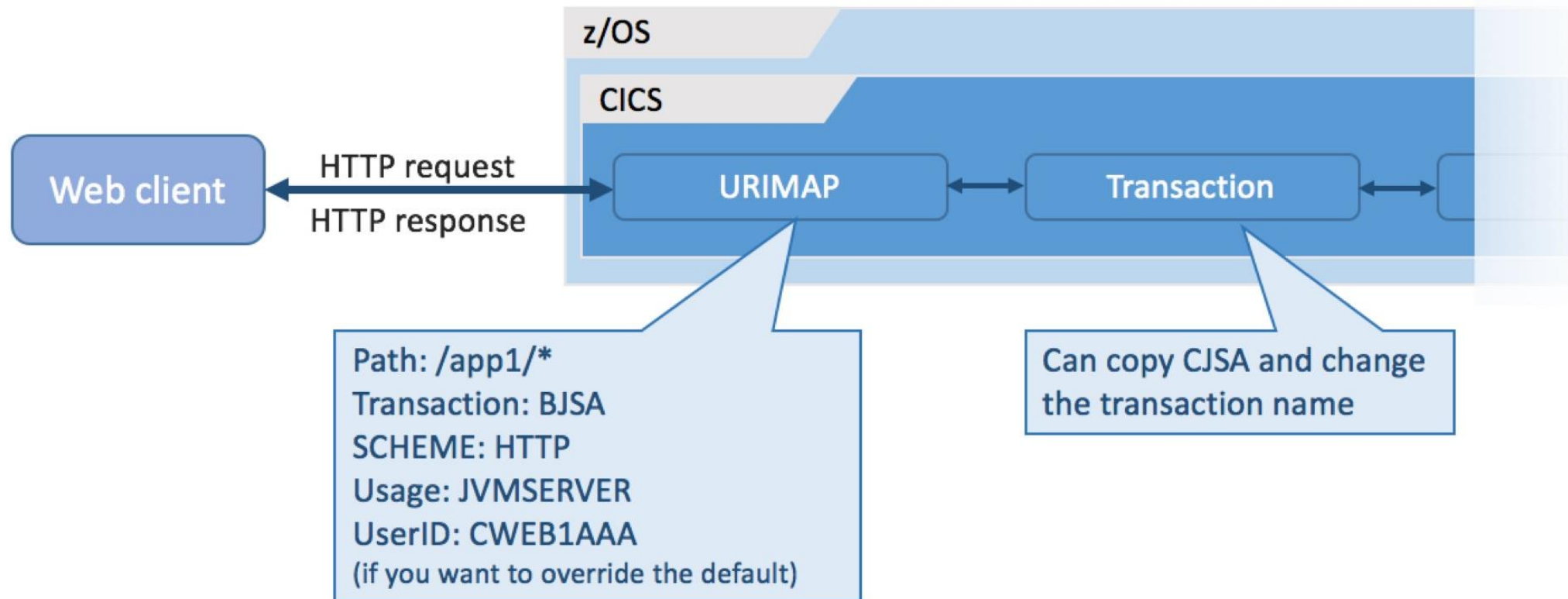


Notes

- EJBROLE RACF profiles can be used to restrict access to individual URLs within the application to individual RACFusers or groups
- The @RolesAllowed annotation can also be used to restrict access to particular Java methods using EJBROLES

CICS transaction security

- By default work runs under transaction CJSA with DFLTUSER
- URIMAPs can be used to context switch to a 'user' transaction
 - Can then make use of traditional CICS transaction security



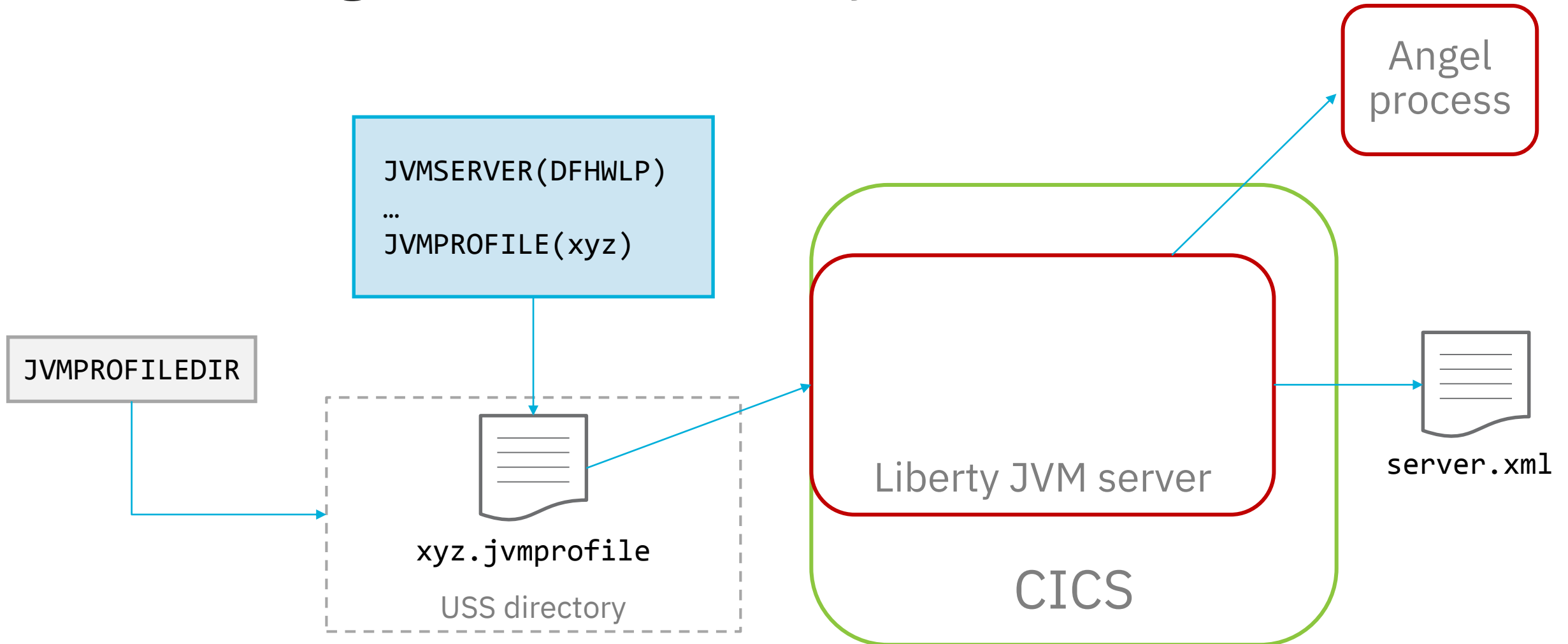
Notes

- By defining a URIMAP, you can specify the default user ID (used if no other authentication information is available) and the transaction ID that the CICS task will run under

Security configuration

Notes

Basic configuration for Liberty in CICS



Notes

When installing a JVMSERVER resource into CICS, one of the few attributes in the resource definition is JVMPROFILE.

The JVMPROFILE attribute references the name of a file that is located on zFS, in the directory specified by the JVMPROFILEDIR SIT parameter.

Inside the zFS file are several configuration options which define the “shape and size” of the installed JVMSERVER resource.

One of those configuration options is a flag to indicate this is a *Liberty JVM server*.

Once established as being a Liberty JVM server, the Liberty runtime reads its own XML configuration file on zFS, known as server.xml.

It is this server.xml file which we will be using to configure the Liberty runtime.



Liberty angel process improvements in CICS TS V5.5

Multiple Liberty servers in the same CICS region

- Each has its own configuration and lifecycle – ideal for developers
- Provides application isolation and redundancy
- WLP_ZOS_PLATFORM profile option is now deprecated

Option to wait for Liberty angel process

- More robust Liberty start-up
- Available on CICS TS V5.4 with PI92676



Notes

- Previously a restriction meant that you could only run one Liberty JVM server connecting to the angel process per CICS region. This restriction has been lifted in CICS TS 5.5
- The WLP_ZOS_PLATFORM, used to run further Liberty JVM servers without connected to the angel process, is no longer needed and is now deprecated.
- Add `-Dcom.ibm.ws.zos.core.angelRequired=true` to your JVM profile. If the angel process is unavailable when a Liberty JVM server is enabled, CICS waits 30 seconds and then retries the connection to the angel. After 5 such attempts CICS issues a WTOR.



Basic security setup

- Setup the angel process to provide access to MVS authorized services
- Create SAF definitions and permissions
- Enable CICS security (SEC=YES)
 - If using automatic configuration, `cicsts:security-1.0` automatically added as feature



Notes

- An instance of the Angel process is required on each LPAR that want to run Liberty with SAF integration



Basic security setup – server.xml

```
<featureManager>  
  <feature>cicsts:security-1.0</feature>  
  ...  
</featureManager>  
  
<safRegistry id="saf" />  
  
<safCredentials profilePrefix="APPLID"  
  unauthenticatedUser="WSGUEST" />
```

Notes

- `profilePrefix` is the prefix Liberty will use when looking up SAF profiles. This defaults to `BBGZDFLT`, but you can use the CICS applid instead, so that SAF profiles for Liberty match up with those used by CICS>

Web application security : web.xml

```
<security-constraint>
  <display-name>examples.web_SecurityConstraint</display-name>
  <web-resource-collection>
    <web-resource-name>examples.web</web-resource-name>
    <description>Protection area for examples.web</description>
    <url-pattern>/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <description>All Authenticated users of examples.web</description>
    <role-name>cicsAllAuthenticated</role-name>
  </auth-constraint>
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```



Notes

- If no `auth-constraint` is specified at all, by default no authentication or authorization will be performed!
- The `cicsAllAuthenticated` role is defined automatically when you deploy an application using a CICS bundle. It maps to a special subject which allows access from any user ID valid in the user registry.
- By specifying `<transport-guarantee> CONFIDENTIAL </transport-guarantee>`, a TLS connection is required for this application – plain HTTP requests are rejected.



Enabling security : <application-bnd> in server.xml

```
<server>
  <!-- Include file for CICS bundle installed applications -->
  <application id="examples.web" name="examples.web" type="war"
    location="{server.config.dir}/installedApps/examples.web.war">
    <application-bnd>
      <security-role name="cicsAllAuthenticated">
        <special-subject type="ALL_AUTHENTICATED_USERS"/>
      </security-role>
    </application-bnd>
  </application>
</server>
```



Notes

- The specified role name must match the one defined in web.xml
- If using user-defined roles without EJBROLE mapping, CICS bundle deployment can't be used as roles must be defined in the application-bnd element



SSL

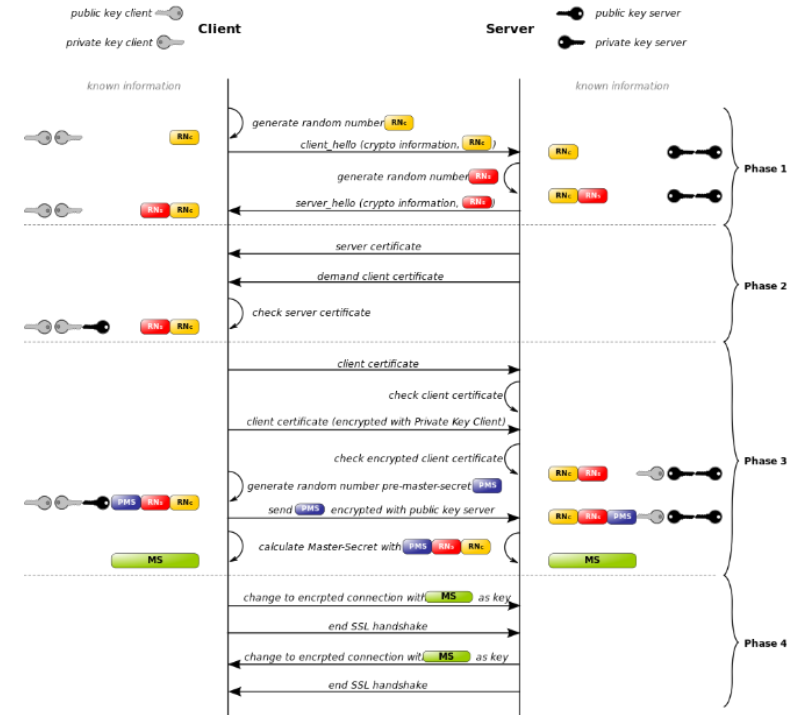
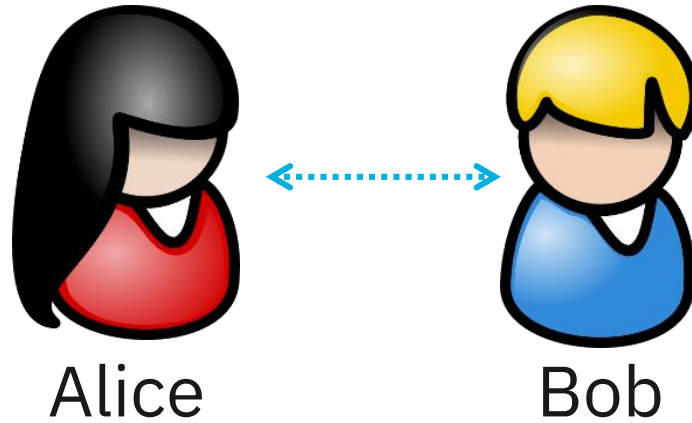
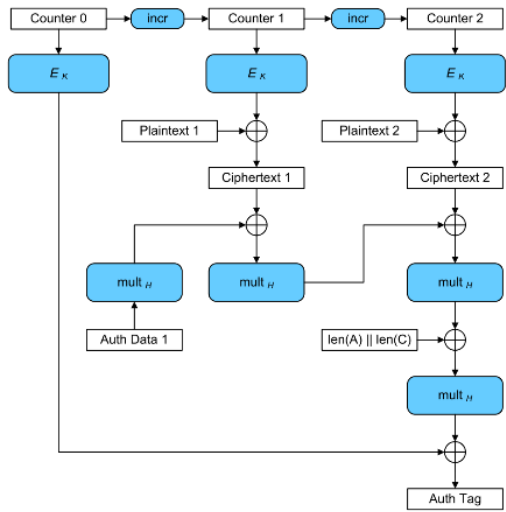
A level-set

Notes

This section will provide a very high-level overview of the SSL process, in order to establish some terminology and a common understanding of the processing involved.



What are we not covering?



https://commons.wikimedia.org/wiki/File:Ssl_handshake_with_two_way_authentication_with_certificates.svg



Notes

We are not looking at the fundamentals of cryptography, although it does seem like every cryptography primer requires the use of “Alice” and “Bob”, so here they are.

We are also not looking at the mathematical theory behind all of the cryptography algorithms.

Neither will we be discussing the SSL protocol in great detail.



Terminology and timeline

- SSL & TLS
- Public-key cryptography
 - + Requires no secure prior communication
 - Computationally expensive
- Private-key cryptography
 - + Computationally cheaper than public-key
 - Requires a pre-agreed secret

Nov 94	SSL v2
Nov 95	SSL v3
Jan 99	TLS v1.0
Apr 06	TLS v1.1
Aug 08	TLS v1.2
Mar 18	TLS v1.3



Notes

The chart shows a brief history of the SSL / TLS protocols. TLS was really just a renaming of SSL and unless specifically noted in this presentation, when referring to “SSL”, we really mean the secure means of communicating defined by any of the SSL or TLS protocols.

There are two main sorts of cryptography used in computer communications: public-key and private-key.

Public key cryptography allows the secure exchange of a message, without the requirement for any communication of a pre-agreed secret. This is also known as asymmetric key encryption, because different keys are used for encryption and decryption.

Private key cryptography does require the use of a pre-agreed secret, but is generally-speaking computationally less expensive than public-key cryptography, for an equivalent level of security. This is also known as symmetric key encryption, because the same key is used for encryption and decryption.



Basic SSL process

- SSL handshake
 - Use public-key cryptography to agree secret key
- SSL session
 - Use private-key cryptography for bulk data transmission
- Full and partial handshakes



Notes

Essentially, we can break down the SSL protocol into two components: the SSL handshake phase and the main SSL data transmission phase.

The handshake phase uses the public-key cryptography process highlighted earlier to agree a session key.

This session key is used to perform the bulk data encryption during the main SSL data transmission phase.

A full handshake uses the full public-key cryptography process, whereas a partial handshake allows each party to resume a previously-terminated SSL session, by having cached the SSL token.



Cipher suites

What's in a name?

Notes

What is a cipher suite and what does the name mean?



A modern cipher suite

TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256



Notes

This is an example of a modern cipher suite, used for data encryption on the WWW.



A modern cipher suite

TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

- TLS
 - Protocol

Transport Layer Security



Notes

In this session, we will only be using TLS v1.2.



A modern cipher suite

TLS_**ECDHE**_ECDSA_WITH_AES_128_GCM_SHA256

- TLS
 - Protocol
- ECDHE
 - Key exchange algorithm

Elliptic Curve Diffie–Hellman Ephemeral



Notes

No notes.



A modern cipher suite

TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

- TLS
 - Protocol
- ECDHE
 - Key exchange algorithm
- ECDSA
 - Authentication algorithm

Elliptic Curve Digital Signature Algorithm



Notes

No notes.



A modern cipher suite

TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

- TLS
 - Protocol
- ECDHE
 - Key exchange algorithm
- ECDSA
 - Authentication algorithm
- AES_128_GCM
 - Bulk encryption algorithm (cipher + size + mode)

Advanced Encryption Standard

128-bit key size

Galois/Counter Mode



Notes

No notes.



A modern cipher suite

TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

- TLS
 - Protocol
- ECDHE
 - Key exchange algorithm
- ECDSA
 - Authentication algorithm
- AES_128_GCM
 - Bulk encryption algorithm (cipher + size + mode)
- SHA256
 - Message authentication code (MAC) algorithm

Secure Hash Algorithm (SHA-2)

256-bit hash output



Notes

The MAC algorithm provides data integrity checks to ensure that the data sent does not change in transit.



Cipher suites

Configuring Liberty

Notes

This section looks at how to configure SSL in Liberty.



Enabling SSL communications

```
<featureManager>  
  <feature>ssl-1.0</feature>  
</featureManager>
```

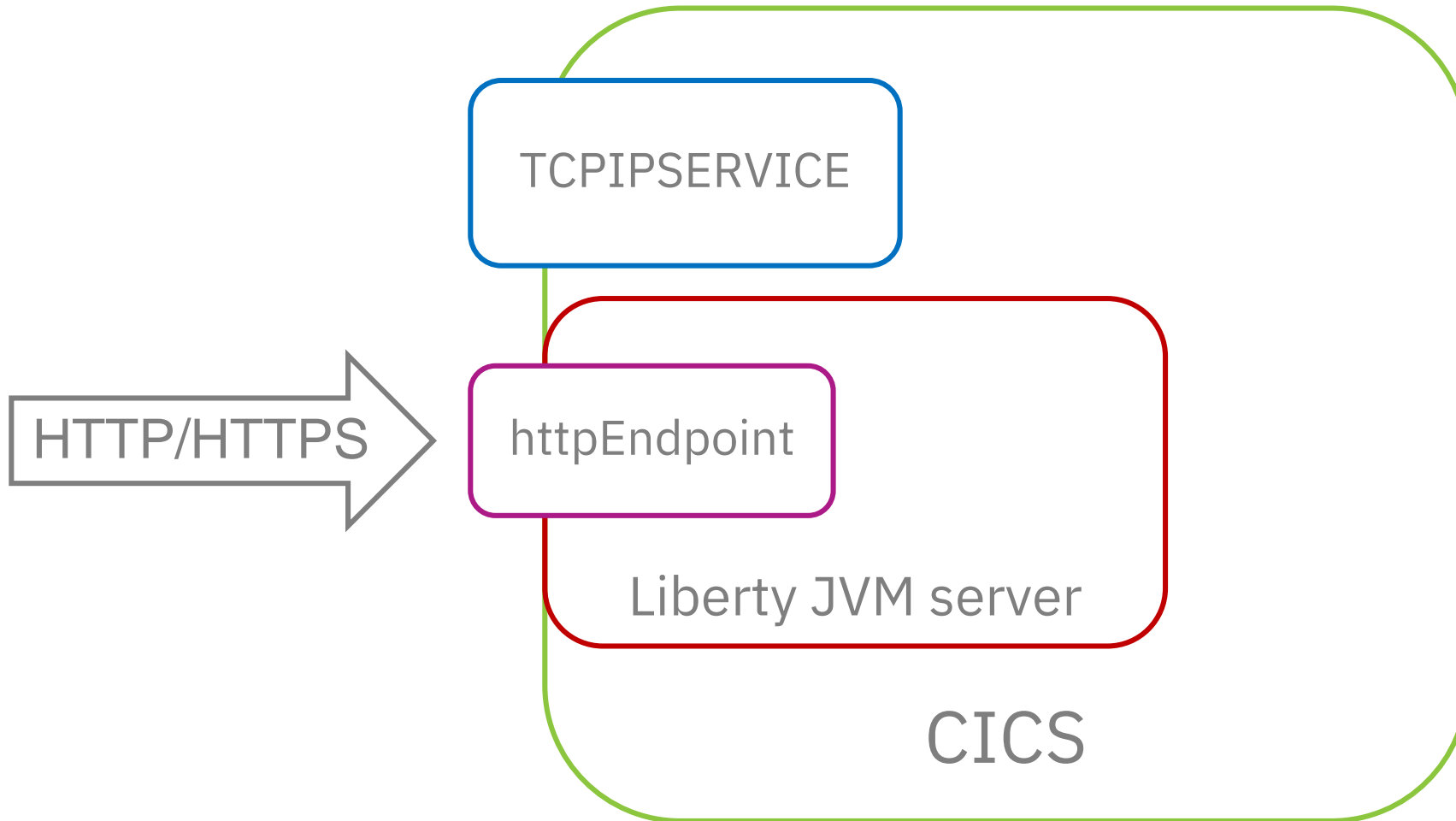


Notes

Add the `ssl-1.0` feature to the `<featureManager>` tag in `server.xml`.



CICS and Liberty HTTP ports



Notes

Note that ports are controlled by Liberty, and are completely independent of CICS control. They are distinct from regular CICS TCPIP SERVICE resources.



Enabling HTTPS port

```
<httpEndpoint id="defaultHttpEndpoint"  
  host="*"   
  httpPort="38010"  
  httpsPort="38011" />
```

(Multiple per Liberty instance)

https://www.ibm.com/support/knowledgecenter/SS7K4U_liberty/com.ibm.websphere.liberty.autogen.zos.doc/ae/rwlp_config_httpEndpoint.html



Notes

To disable the HTTP port, simply set `httpPort` to `-1`.


You can also have multiple HTTP endpoints per Liberty instance.



Additional HTTP and SSL configuration

```
<httpEndpoint id="defaultHttpEndpoint"
              httpOptionsRef="myHttpOptions"
              sslOptionsRef="mySslOptions" ... />
```

defaultHttpOptions



```
<httpOptions id="myHttpOptions" ... />
```

```
<sslOptions id="mySslOptions" ... />
```

https://www.ibm.com/support/knowledgecenter/SS7K4U_liberty/com.ibm.websphere.liberty.autogen.zos.doc/ae/rwlp_config_httpEndpoint.html



Notes

Default for `httpOptionsRef` is `defaultHttpOptions`.

There is no default value for `sslOptionsRef`.

If you want to use a RACF keyring rather than a Java keystore for storing certificates, you'll need to configure it here. See the Redbook for information about setting up a RACF keyring and configuring Liberty to use it.



Persistent sessions and HTTP Keep-Alive

```
<httpOptions id="defaultHttpOptions"  
    keepAliveEnabled="true"  
    maxKeepAliveRequests="100"  
    ... />
```

https://www.ibm.com/support/knowledgecenter/SS7K4U_liberty/com.ibm.websphere.liberty.autogen.zos.doc/ae/rwlp_config_httpEndpoint.html



Notes

keepAliveEnabled

Enables persistent connections (HTTP keepalive). If true, connections are kept alive for reuse by multiple sequential requests and responses. If false, connections are closed after the response is sent.

maxKeepAliveRequests

Maximum number of persistent requests that are allowed on a single HTTP connection if persistent connections are enabled. A value of -1 means unlimited.

A very high value for maxKeepAliveRequests can cause workloads to become “sticky” to a specific region when using High Availability clusters. See the Redbook for details.

Default values for both parameters are shown.



SSL session timeout

```
<sslOptions id="mySslOptions"  
    sslSessionTimeout="10m"  
    ... />
```

https://www.ibm.com/support/knowledgecenter/SS7K4U_liberty/com.ibm.websphere.liberty.autogen.zos.doc/ae/rwlp_config_httpEndpoint.html



Notes

sslSessionTimeout

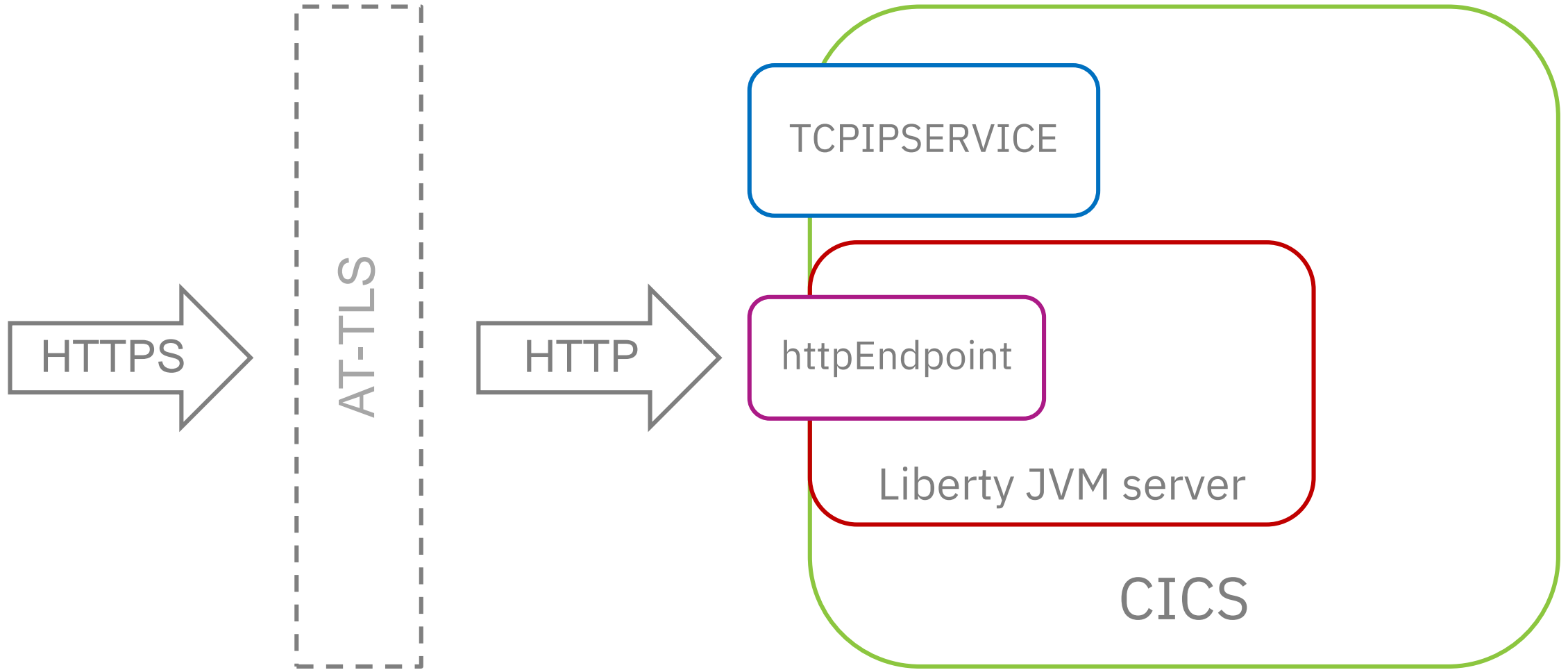
Specifies the time interval in the server during which the server allows SSL session IDs to be resumed.

The default setting is 8640 ms.

Increasing the session timeout can be a performance benefit if HTTP connections are not long-lived, and TLS clients must reconnect frequently. For optimum performance, the `sslSessionTimeout` attribute must be set to longer than the time between client HTTP requests. Setting this attribute allows the server to reuse session IDs.



AT-TLS



Notes

AT-TLS is also fully supported by the Liberty runtime when running in fully-transparent mode.

Note that Liberty cannot operate in AT-TLS aware mode. This is where the AT-TLS stack is providing the encryption, but the application server is aware of the presence of AT-TLS. Due to Liberty not being aware of the AT-TLS stack, client authentication will not function using AT-TLS.



Specifying ciphers in Liberty

```
<ssl id="defaultSSLConfig"  
    enabledCiphers="SSL_ECDHE_RSA_WITH_AES_128_CBC_SHA256  
                  SSL_ECDHE_RSA_WITH_AES_128_GCM_SHA256"  
    keyStoreRef="defaultKeyStore"  
    sslProtocol="TLSv1.2" />
```

https://www.ibm.com/support/knowledgecenter/SS7K4U_liberty/com.ibm.websphere.liberty.autogen.zos.doc/ae/rwlp_config_ssl.html



Notes

In this `<ssl>` tag we are specifying two ciphers for use in the server.

See the referenced doc link for more information.



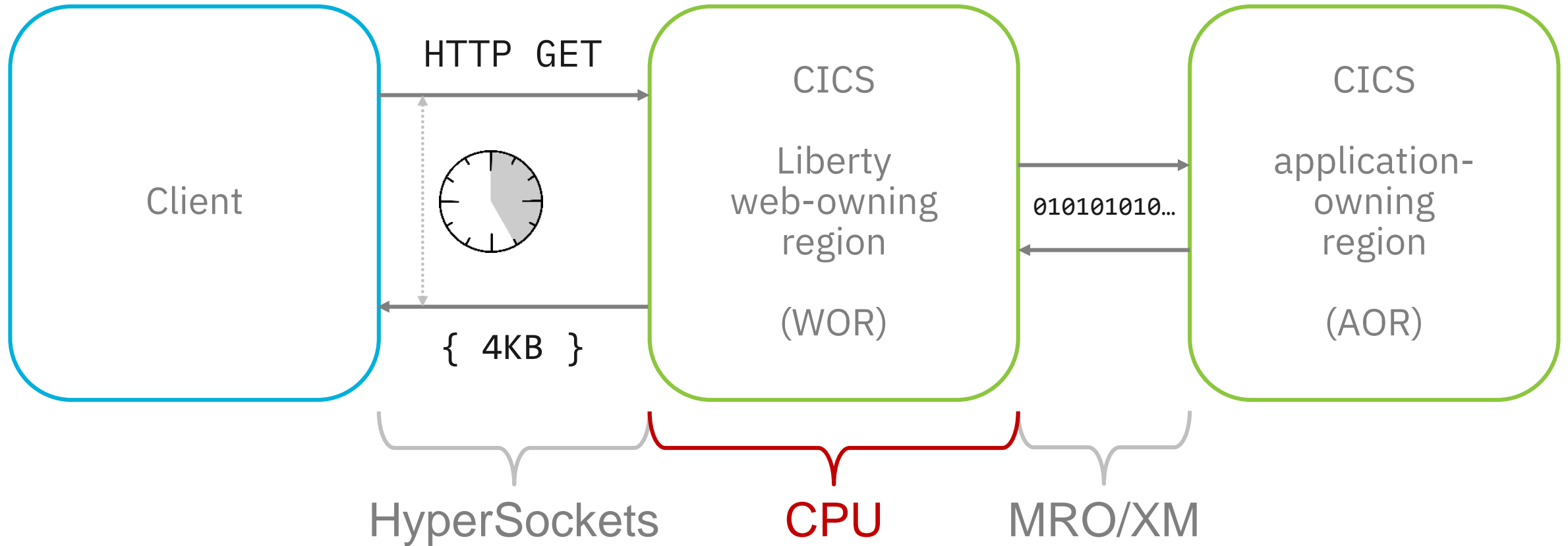
Benchmark configuration

Notes

This section deals with the configuration of the benchmark used to obtain the performance results.



Application configuration



Notes

The client performs an HTTP GET request for a resource in the Liberty web-owning region (WOR).

The WOR is running a simple JAX-RS application which then issues a link using the JCICS command to a program defined as remote in an application-owning region (AOR), connected using MRO/XM.

The application returns binary data to the WOR, which is then converted to 4KB of JSON data, and returned to the client.

The response time is measured at the client, and the CPU consumed is measured only in the WOR.

By using this configuration, we can split out the cost of the business logic from the cost of the data encryption.



System detail

- IBM z14
 - 3906-708 – no zIIP engines (using RMF IIPCP)
- z/OS V2.2
- CICS TS V5.4, Liberty 17.0.0.3, Java 8 SR5
- IBMJCE provider and server certificates stored in RACF
- 250 simulated clients
 - Persistent sessions – approx. 750 requests per second in total
 - Non-persistent sessions – approx. 600 requests per second in total
 - Not using client authentication



Notes

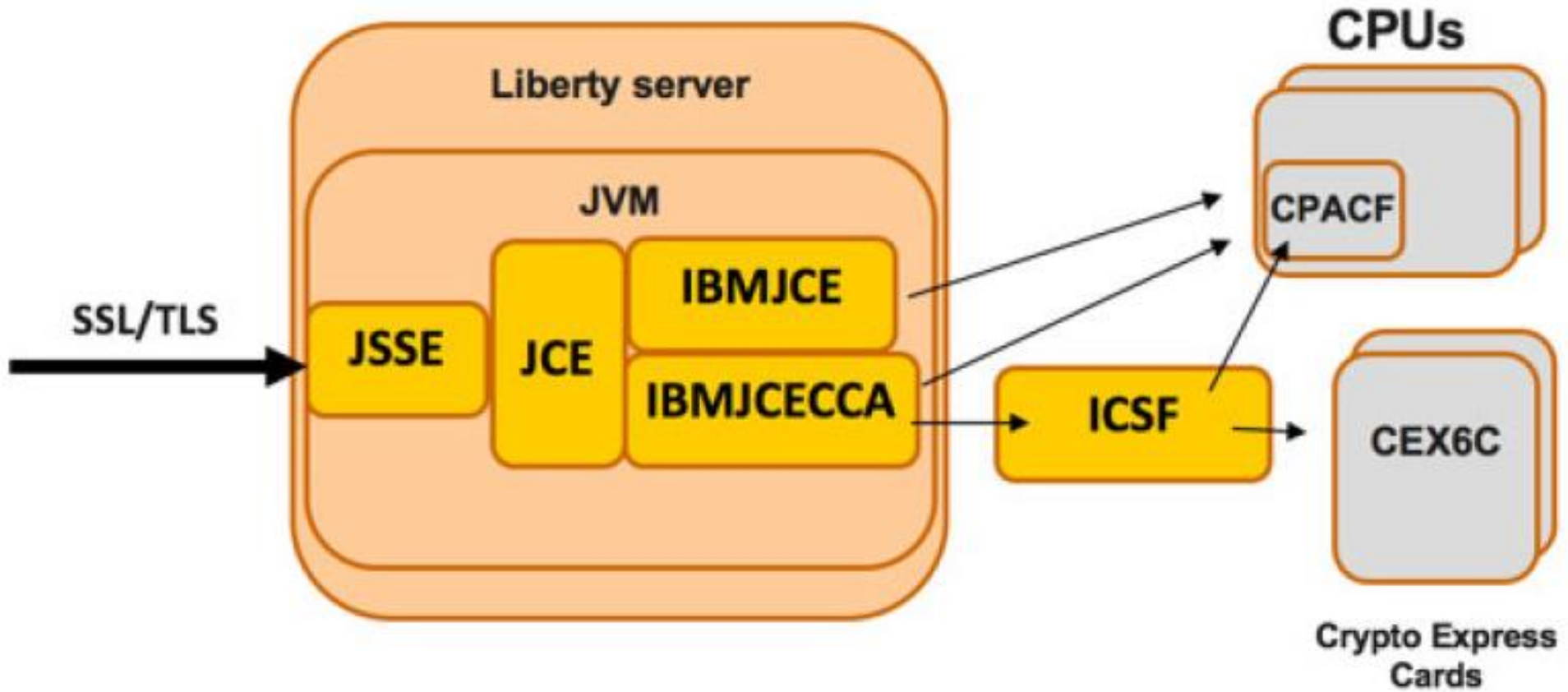
All of these performance figures were obtained using z14 hardware. The z14 architecture provides a significant improvement in cryptographic performance over the z13.

The JVM used the IBMJCE provider for the JCE cryptographic services, and therefore did not use the CryptoExpress cards.

Clients ran on a separate z/OS LPAR, connecting to the server via HyperSockets.



JCE providers



Notes

Java Cryptography Extension (JCE) provides a common Java API for many cryptographic functions. Implementations of this API are called JCE providers. On the Z platform, configuring the JCE provider in the JVM allows you to choose the hardware supported.

The IBMJCE provider only uses CPACF(CP Assist for Cryptographic Functions)

The IBMJCECCA provider uses the ICSF facility, which can be configured to use the CryptoExpress cards, or the CPACF facility.

See the Redbook publication for more details about configuring these providers.



Performance measurements

Key exchange and authentication algorithms

Notes

In this section we are only choosing cipher suites supported by TLS v1.2.

Each SSL handshake scenario used certificates stored in RACF. The server certificate was stored as a personal certificate for the CICS region userid. The server certificate had a signing certificate stored as a certificate authority.

Both the server and CA certificates for the RSA-based algorithms were 2048 bits.

Both the server and CA certificates for the EC-based algorithms were 256 bits.



Key exchange and authentication algorithms

- [003C] TLS_RSA_...
- [C023] TLS_ECDHE_ECDSA_...
- [C027] TLS_ECDHE_RSA_...
- [C029] TLS_ECDH_RSA_...

..._WITH_AES_128_CBC_SHA256



Notes

All HTTP connections were non-persistent to force an SSL handshake every time.

RSA = Rivest, Shamir and Adleman

ECDH = Elliptic Curve Diffie–Hellman

ECDHE = Elliptic Curve Diffie–Hellman Ephemeral

TLS_RSA_.... means that RSA is used for both key exchange and authentication



Excluded from comparisons

- [003F] TLS_DH_RSA_... (Diffie-Hellman)
 - Not supported by IBM Java

..._WITH_AES_128_CBC_SHA256

https://www.ibm.com/support/knowledgecenter/en/SSYKE2_8.0.0/com.ibm.java.security.component.80.doc/security-component/jsse2Docs/ciphersuites.html

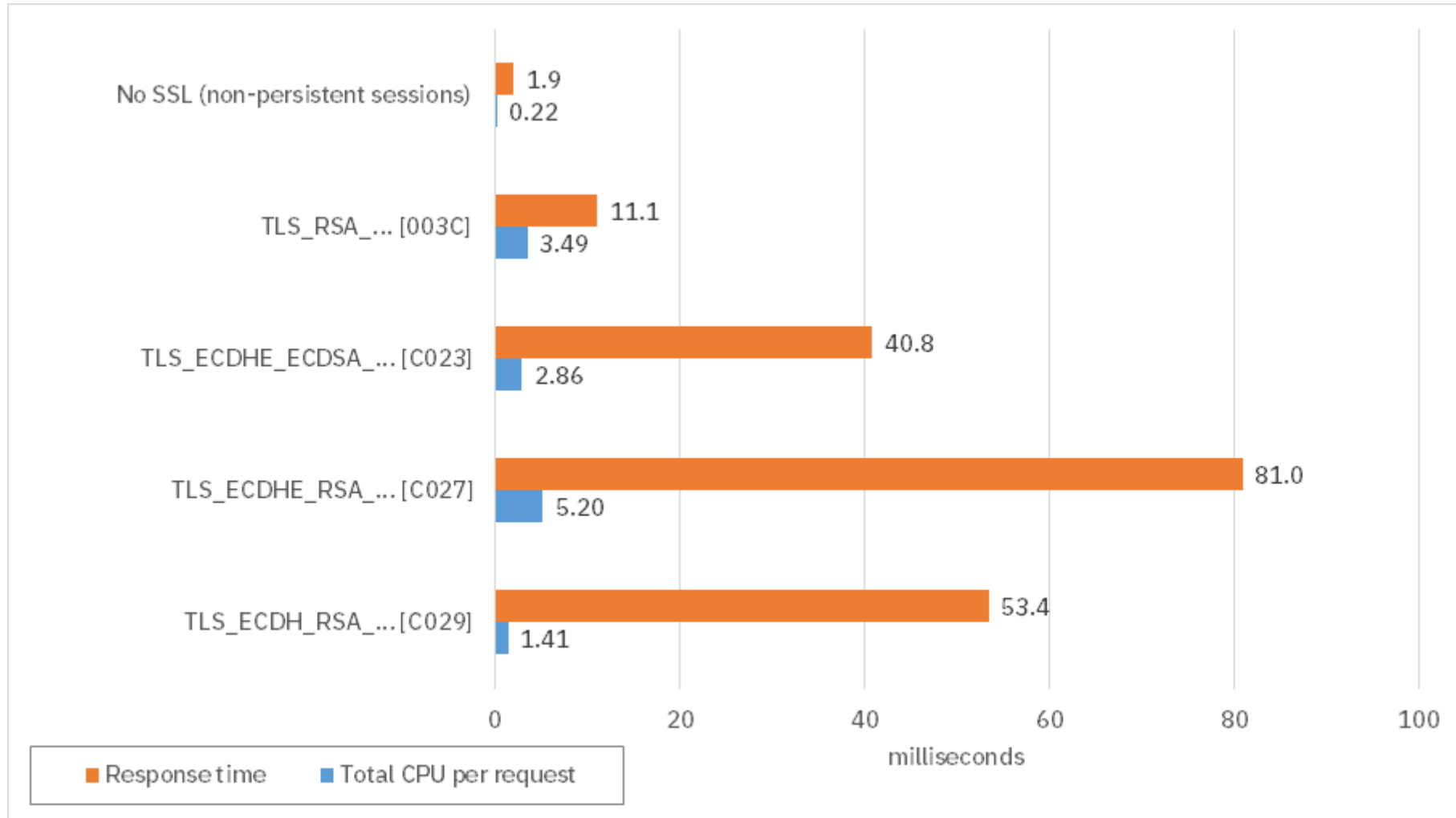


Notes

The 003F cipher is not supported by IBM Java SDK for z/OS, therefore not included in these comparisons.



CPU and response times (full handshake)

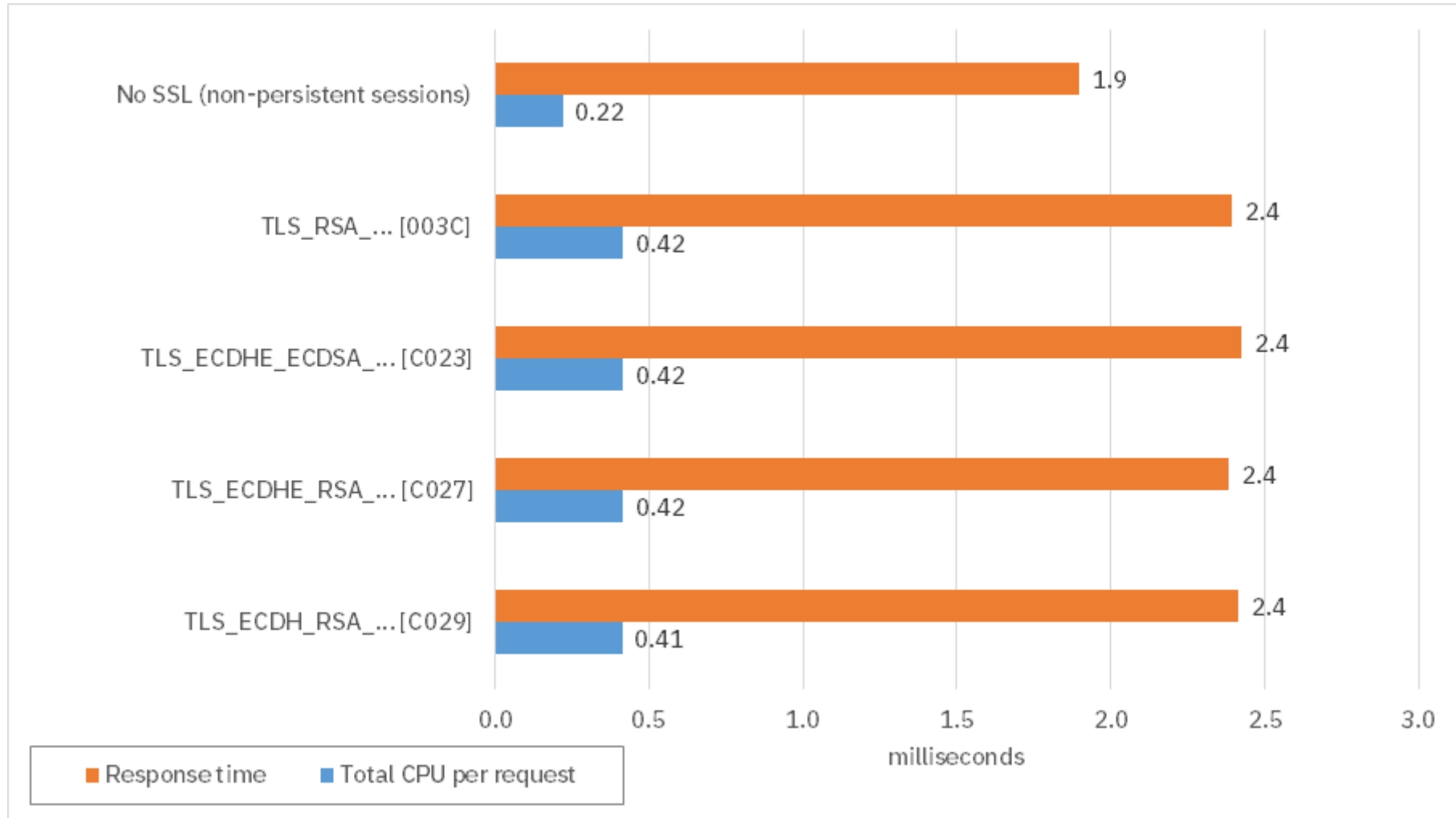


Notes

Comparison S01.



CPU and response times (partial handshake)



Notes

Comparison S02.

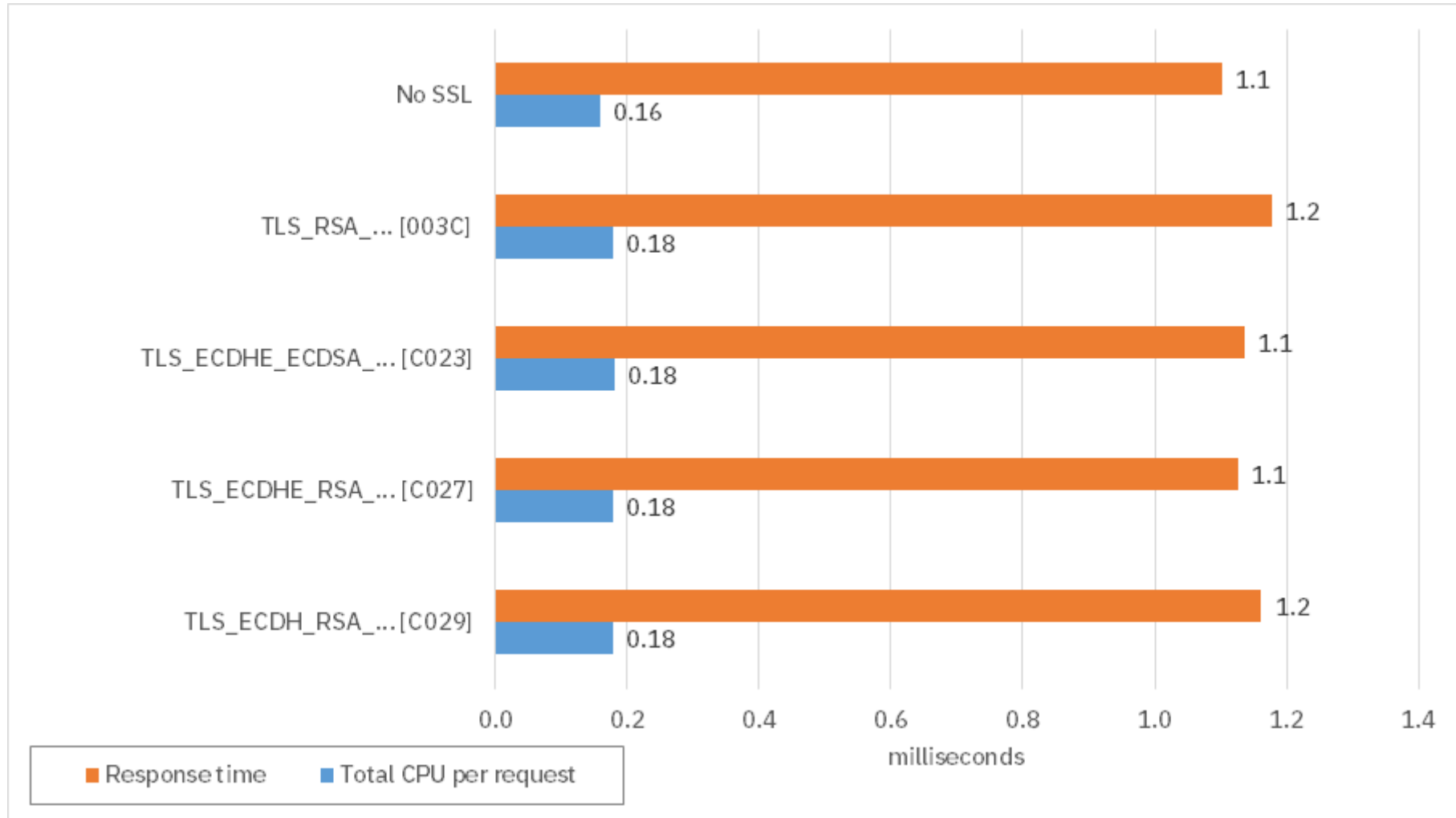
Note the significantly different scale for the x-axis when viewed alongside comparison S01. See comparison S04 later in this presentation for a side-by-side view.

Regardless of the key exchange algorithms used, if you have a session key cached, then cost is the same to re-establish an SSL session, regardless of the original key exchange mechanism.

Refer back to earlier in the presentation for how the SSL handshake process works.



Persistent connections results



Notes

Comparison S03.

With no SSL handshakes taking place during the performance measurement, and all algorithms are using the same bulk encryption and MAC algorithms, we would expect the results to be identical for each of the cipher suites.



Performance measurements

Persistent connections

Notes

The previous section showed the high cost of establishing a new session with a full handshake, and how this can be improved using partial handshakes.

When using persistent sessions, no SSL handshakes take place after the initial connection request.

This section provides a comparison of all of the combinations of SSL, non-SSL, persistent sessions, partial and full handshakes.



Persistent and non-persistent connections

- No SSL
 - Persistent connections
 - Non-persistent connections
- [C023] TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
 - Persistent connections
 - Non-persistent connection with partial handshake
 - Non-persistent connection with full handshake

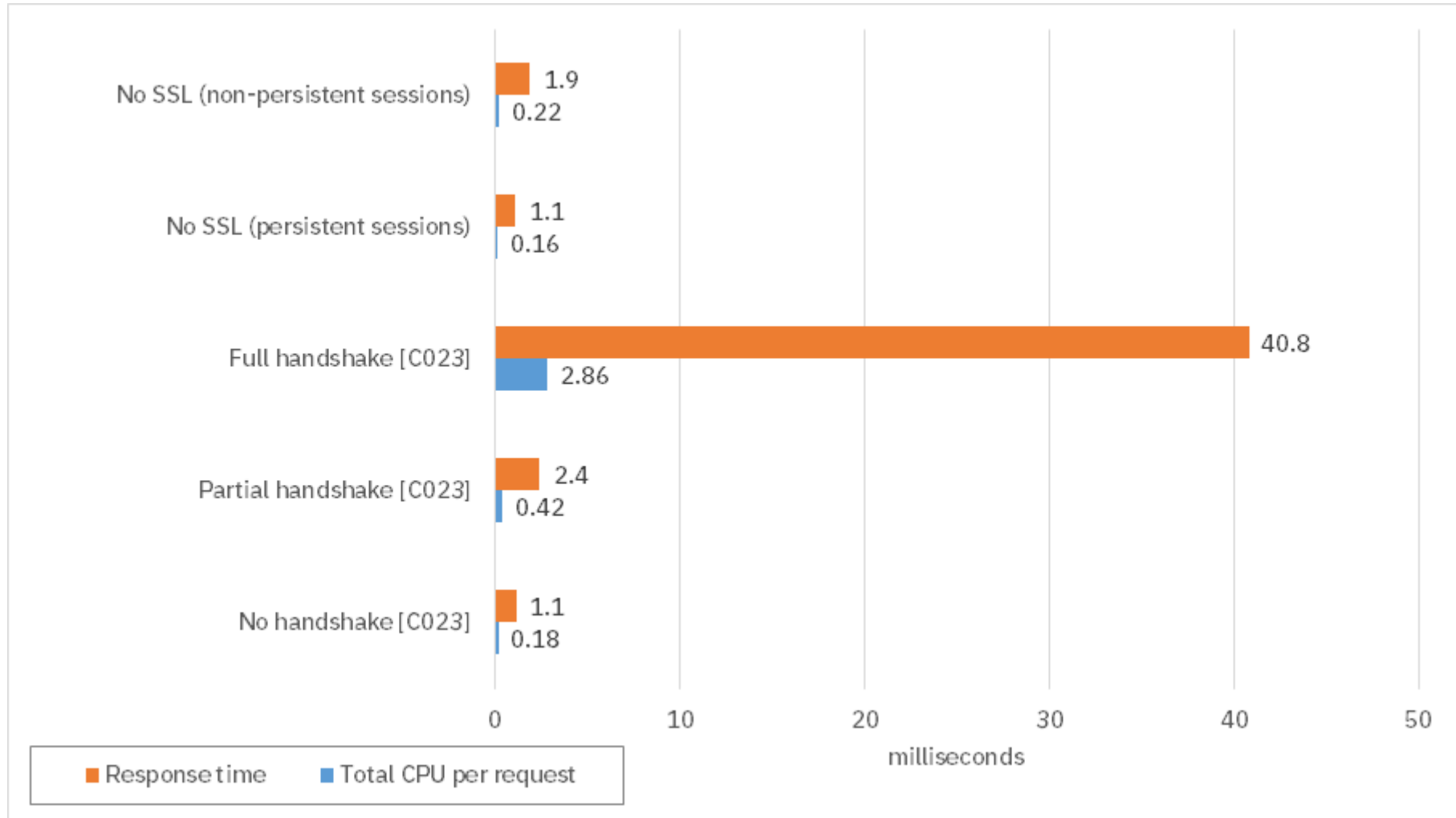


Notes

No notes.



Persistent and non-persistent connections results



Notes

Comparison S04.

As expected, an HTTP connection with persistent sessions and no encryption is the cheapest in terms of CPU per request. Re-establishing the HTTP connection for each non-SSL request adds a very small overhead in terms of CPU and response time.

When adding encryption, the full SSL handshake protocol adds a significant amount of CPU to the overall cost, and the number of network flows used to establish an SSL session results in a relatively large response time.

By allowing both the client and server to cache recent SSL session IDs, the cost of re-establishing a previously-closed session is significantly smaller than performing a full SSL handshake each time.

In this scenario, adding SSL encryption to a connection using persistent sessions is a very low-cost operation.

Response time is the same as for the non-SSL case with persistent sessions. There is a very small CPU overhead due to the presence of encryption.

In your environment, make sure you are using persistent connections wherever possible. If that is not possible, then try to re-use SSL session IDs using the session cache.



Performance measurements

Bulk encryption and MAC algorithms

Notes

Only choosing cipher suites supported by TLS v1.2.



Bulk encryption and MAC algorithms

- [C006] ..._WITH_NULL_SHA
- [003B] ..._WITH_NULL_SHA256
- [C009] ..._WITH_AES_128_CBC_SHA
- [C023] ..._WITH_AES_128_CBC_SHA256
- [C02B] ..._WITH_AES_128_GCM_SHA256 (AEAD)
- [C00A] ..._WITH_AES_256_CBC_SHA
- [003D] ..._WITH_AES_256_CBC_SHA256
- [C024] ..._WITH_AES_256_CBC_SHA384
- [C02C] ..._WITH_AES_256_GCM_SHA384 (AEAD)



Notes

All tests used persistent connections to remove any handshake overheads.

AEAD = Authenticated Encryption with Associated Data

The following cipher suites are disabled due to the Bar Mitzvah security vulnerability CVE-2015-2808:

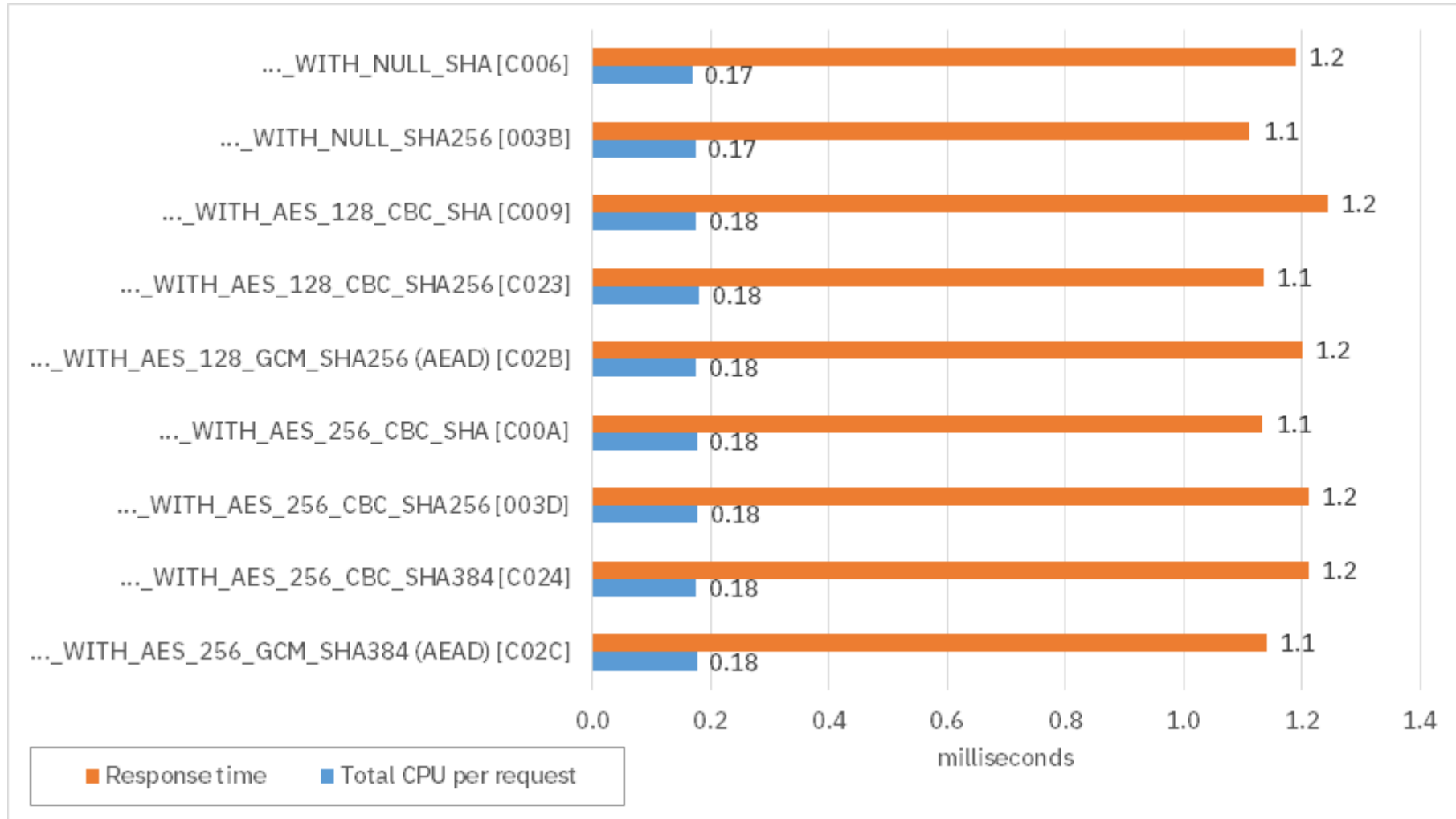
- TLS_ECDHE_ECDSA_WITH_RC4_128_SHA [C007]
- TLS_ECDHE_ECDSA_WITH_RC4_128_SHA [C008]

https://www.ibm.com/support/knowledgecenter/SSYKE2_8.0.0/com.ibm.java.security.component.80.doc/security-component/jsse2Docs/disablerc4.html

https://www.ibm.com/support/knowledgecenter/en/SSYKE2_8.0.0/com.ibm.java.security.component.80.doc/security-component/jsse2Docs/ciphersuites.html



Bulk encryption and MAC algorithms



Notes

Comparison S05.



Message authentication code algorithm

- [0035] ..._WITH_AES_256_CBC_SHA
- [003D] ..._WITH_AES_256_CBC_SHA256
- [C024] ..._WITH_AES_256_CBC_SHA384



Notes

Handshake algorithm not included for clarity – not relevant due to persistent connections.

MD5 is also available as a hash algorithm, but generally associated with weaker algorithms, and therefore not included here.



Excluded from comparisons

- MD5 only used by the following cipher suites in TLS v1.2:
 - [0001] TLS_NULL_WITH_NULL_MD5
 - [0004] TLS_RSA_WITH_RC4_128_MD5

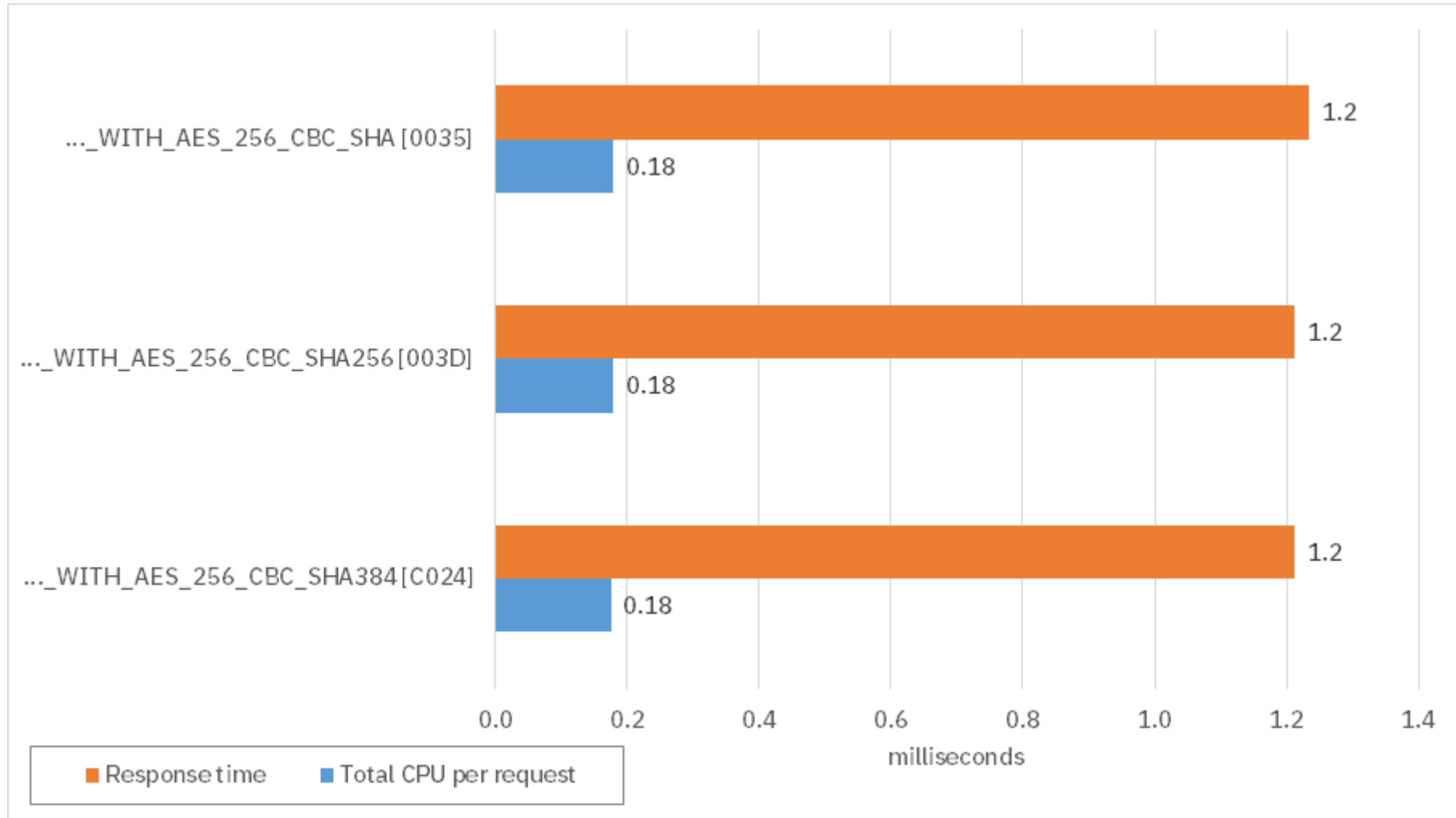


Notes

No notes.



Message authentication code algorithm



Notes

Comparison S06.



Performance measurements

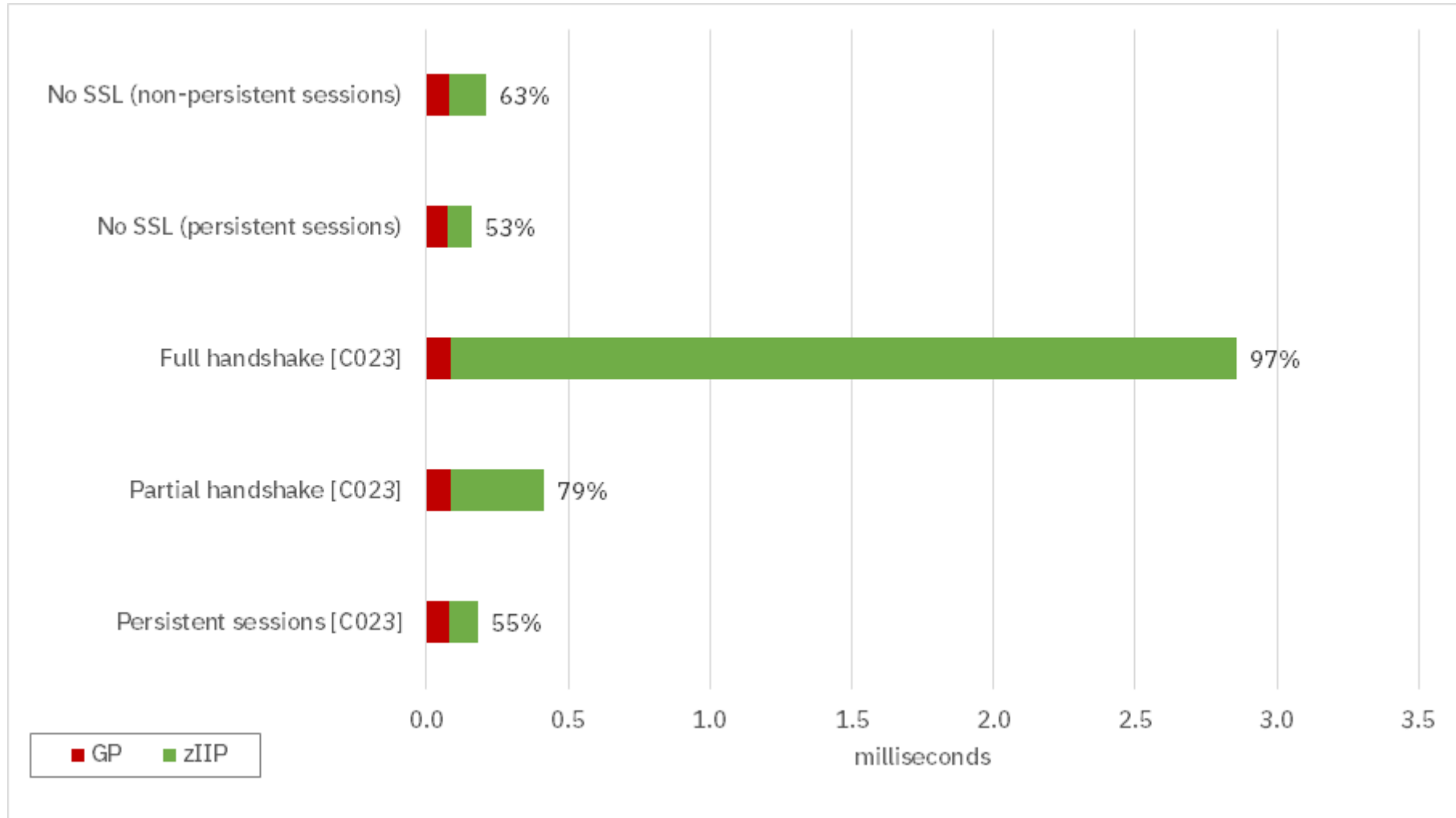
zIIP eligibility

Notes

A quick look at the zIIP eligibility of SSL in Liberty.



zIIP eligibility



Notes

Comparison S07.

In all cases, the GP cost per request remains the same at around 0.08ms of CP GPU per request.

As expected, all of the SSL processing is zIIP-eligible because it happens inside the JVM.



Java implementation

Notes

This section looks at the version of the IBM Java SDK, and how it affects the performance of Java applications.



IBM SDK for Java 7.0 and 7.1 End of Service

[Withdrawal Announcement 916-121 – August 2, 2016](#)

To be withdrawn from service on September 30, 2019:

- IBM 64-bit SDK for z/OS, Java Technology Edition V7.0.0 and V7.1.0

Blog post “[Java 8 recommended for CICS TS V5](#)”



Notes

Note that IBM has announced an End Of Service date for the following two products:

- IBM 64-bit SDK for z/OS, Java Technology Edition V7.0.0
- IBM 64-bit SDK for z/OS, Java Technology Edition V7.1.0

This section outlines the performance benefits of upgrading as soon as possible.



IBM SDK for Java 7.0 vs 8.0

- IBM SDK for Java 7.0 SR10
 - Exploitation of zEC12 and earlier
- IBM SDK for Java 8 SR5
 - Exploitation of z14 and earlier

z14 opcodes

KMA R_1, R_3, R_2

z13 opcodes

zEC12 opcodes

Notes

At runtime, the JVM compiles Java bytecode to native machine instructions. Later versions of the Java SDK for z/OS have knowledge of the instructions available in the latest hardware, can therefore make optimal use of the platform on which it is executing.

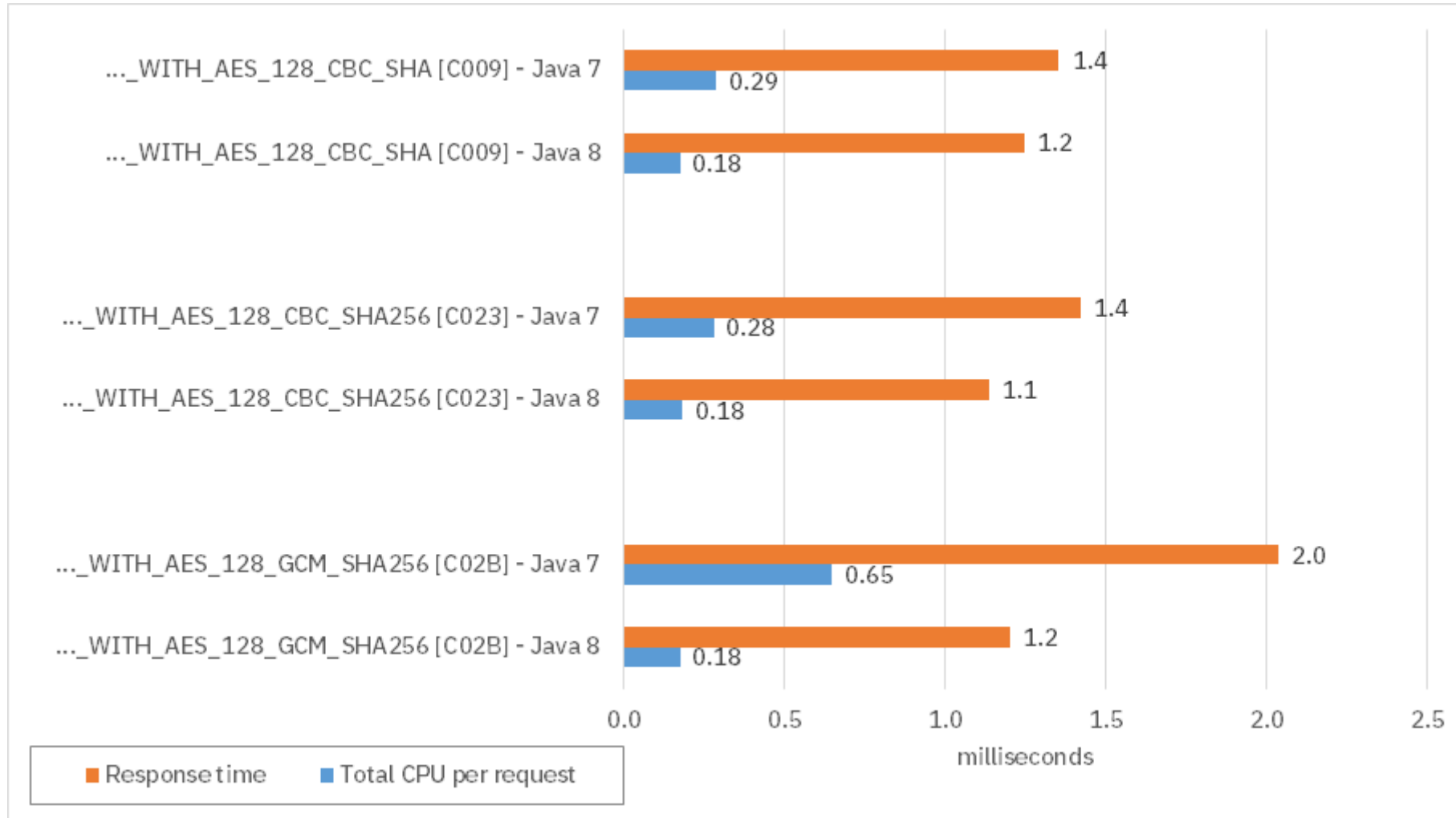
For example, the z14 adds extra function codes to the **COMPUTE INTERMEDIATE MESSAGE DIGEST (KIMD)** and **COMPUTE LAST MESSAGE DIGEST (KLMD)** hardware instructions, which are useful in modern cryptography algorithms. It also provides the new **CIPHER MESSAGE WITH AUTHENTICATION (KMA)** instruction for GCM-based encryption.

The most recent version of Z Systems hardware that IBM SDK for Java 7.0 SR10 can exploit is the IBM zEC12 and earlier.

IBM SDK for Java 8 SR5 exploits z14 hardware and earlier.



Java 7 vs. Java 8 Persistent sessions



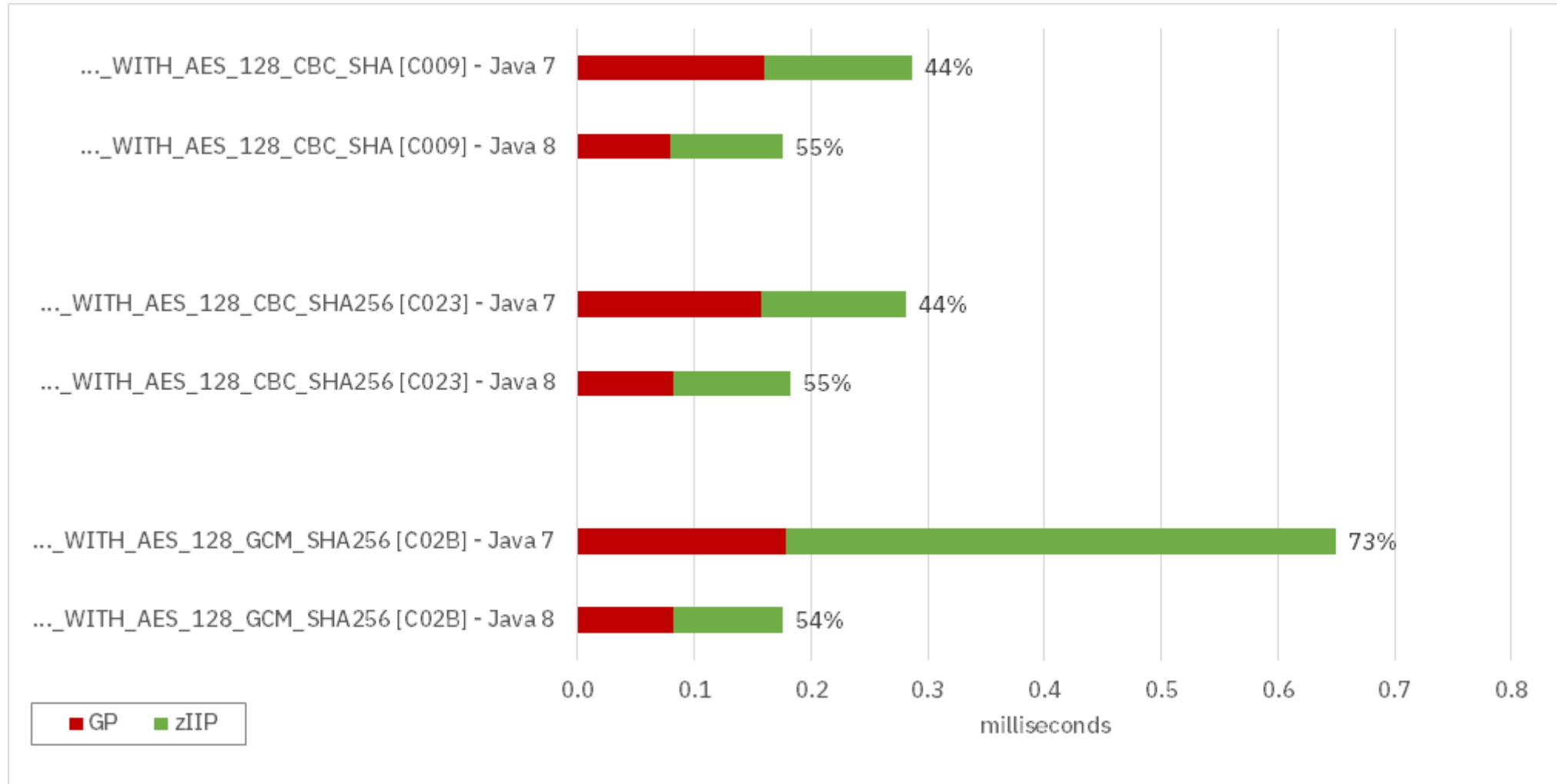
Notes

Comparison S08.

Note the pronounced improvement in the GCM-based encryption when using Java 8 SR5. This is because the JVM can exploit the new instructions in the z14.



Java 7 vs. Java 8 Persistent Sessions (zIIP)



Notes

Comparison S09.

For the `_CBC_SHA` and `_CBC_SHA256` cipher suite, not only is the overall CPU reduced when using Java 8 SR5, but the fraction which is zIIP eligible is also increased.

For the `_GCM_SHA256` cipher suite, the Java 7 zIIP eligibility is higher, simply because the overall amount of processing in Java is much larger than the total of the CPU used by the Java 8 SR5 implementation.



TLS v1.3

- Key exchange algorithms (all with forward secrecy)
 - DHE-RSA
 - ECDHE-RSA
 - ECDHE-ECDSA
- Block-mode cipher algorithms
 - AES GCM
 - AES CCM
 - Camellia GCM
 - ARIA GCM
- Data integrity
 - AEAD



Notes

TLS v1.3 (which was approved by the IETF in March 2018) significantly reduces the number of valid cipher suites permitted by the protocol, removing outdated and insecure encryption methodologies.

This page lists the key exchange algorithms, the block-mode cipher algorithms, and data integrity algorithms that are currently included in the TLS v1.3 draft specification at the time of writing.

TLS 1.3 represents the largest overhaul to secure communications for a long time. When TLS v1.3 becomes mandated in your environment, you will need to ensure that you are using an “approved” cipher suite.



Summary

- Decide on the following aspects of security configuration
 - Authentication method
 - User registry
 - Privacy
 - Authorization
- Use persistent connections wherever possible
 - If not possible, try to allow partial handshakes
- Use the most recent version and FP of Java SDK
 - Provides greatest gains on your hardware



Notes

No notes.



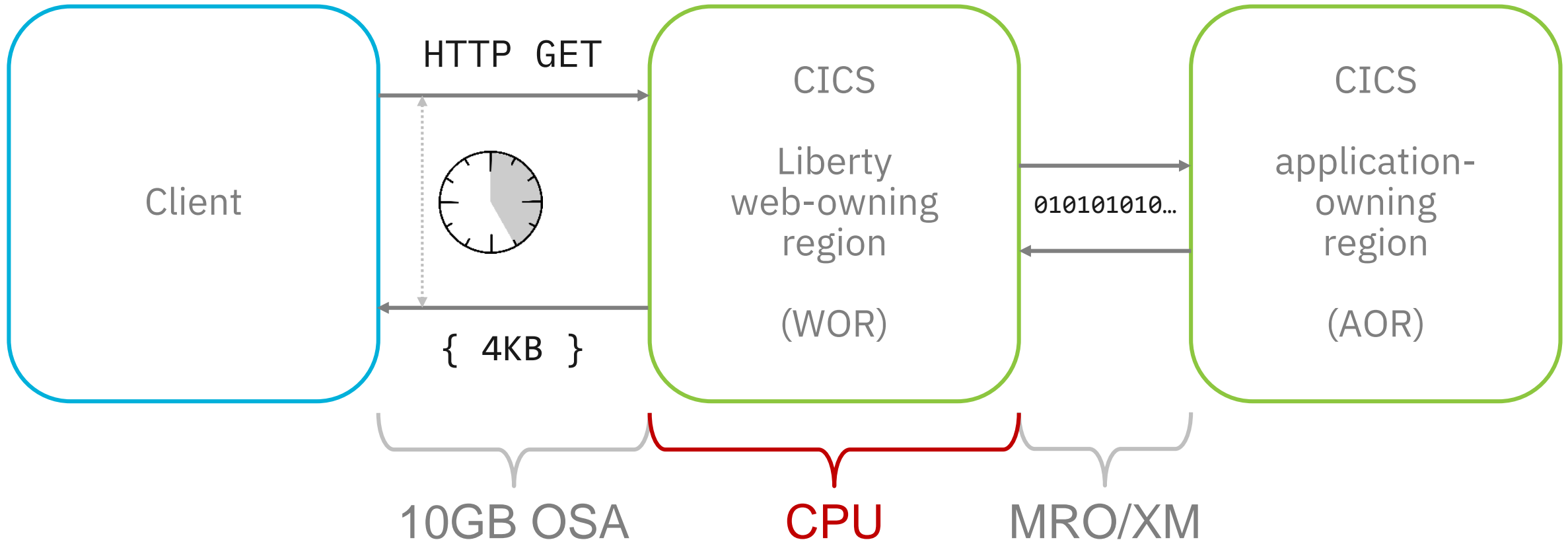
We want your feedback!

- Please submit your feedback online at
 - <http://conferences.gse.org.uk/2018/feedback/GM>
- Paper feedback forms are also available from the Chair person
- This session is **GM**

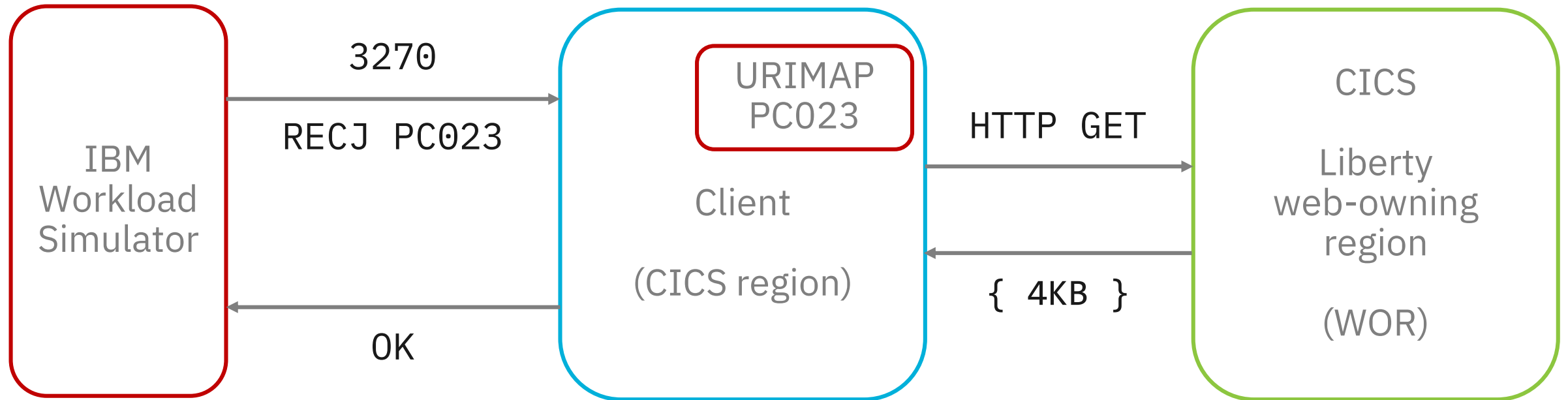


Implementation Details

Application Overview



Implementation detail



Resources

Resources

- IBM Redbooks publication:
“IBM CICS and Liberty: What You Need to Know”
 - redbooks.ibm.com/abstracts/sg248335.html?Open
- IBM Redbooks publication:
“Liberty in IBM CICS: Deploying and Managing Java EE Applications”
 - redbooks.ibm.com/abstracts/sg248418.html?Open
- Securing communications with Liberty
 - https://www.ibm.com/support/knowledgecenter/SS7K4U_liberty/com.ibm.websphere.wlp.zseries.doc/ae/twlp_sec_comm.html
- Tuning Liberty for secure applications
 - https://www.ibm.com/support/knowledgecenter/en/SSEQTP_liberty/com.ibm.websphere.wlp.doc/ae/twlp_tun_sec.html



Resources continued

- TLS V1.2 – RFC 5246
 - <https://datatracker.ietf.org/doc/rfc5246/>
- TLS V1.3 (draft)
 - <https://datatracker.ietf.org/doc/draft-ietf-tls-tls13/>



IBM Z Trial Program

Experience the value of the latest IBM Z capabilities today at no charge, and with no install required.



ibm.biz/ibmztrial



No charge, on-demand environment

With no lead times, and access to a no charge remote environment. Trying out IBM Z capabilities is now easier than ever.



No setup, no install

We provision an environment for you. With all the tooling and connections pre-configured, start trying out the latest IBM Z has to offer in minutes, not hours.



Hands-on tutorials

Experience the latest products and features on the mainframe, with short, step-by-step walkthroughs built in to your trial environment.

CICS TS Java trial now available

- Build a Java web application that provides a REST API
- Deploy into CICS Liberty



IBM®

