z/TPF Communication and Security Enhancements

Raymond Fan

IBM Z

IBM

Agenda

Recent Deliverables

- Support for OpenSSL AES-GCM Ciphers
- ZPUBK REHASH

What's next?

- z/TPF Network Security Compliance Validation Capability

Support for OpenSSL AES-GCM Ciphers

Background

z/TPF currently only has support for OpenSSL ciphers that use AES in Cipher Block Chaining (CBC) mode.

Support for ciphers that use AES in Galois/Counter Mode (GCM) will improve the overall security and performance of the z/TPF system.

Technical Details

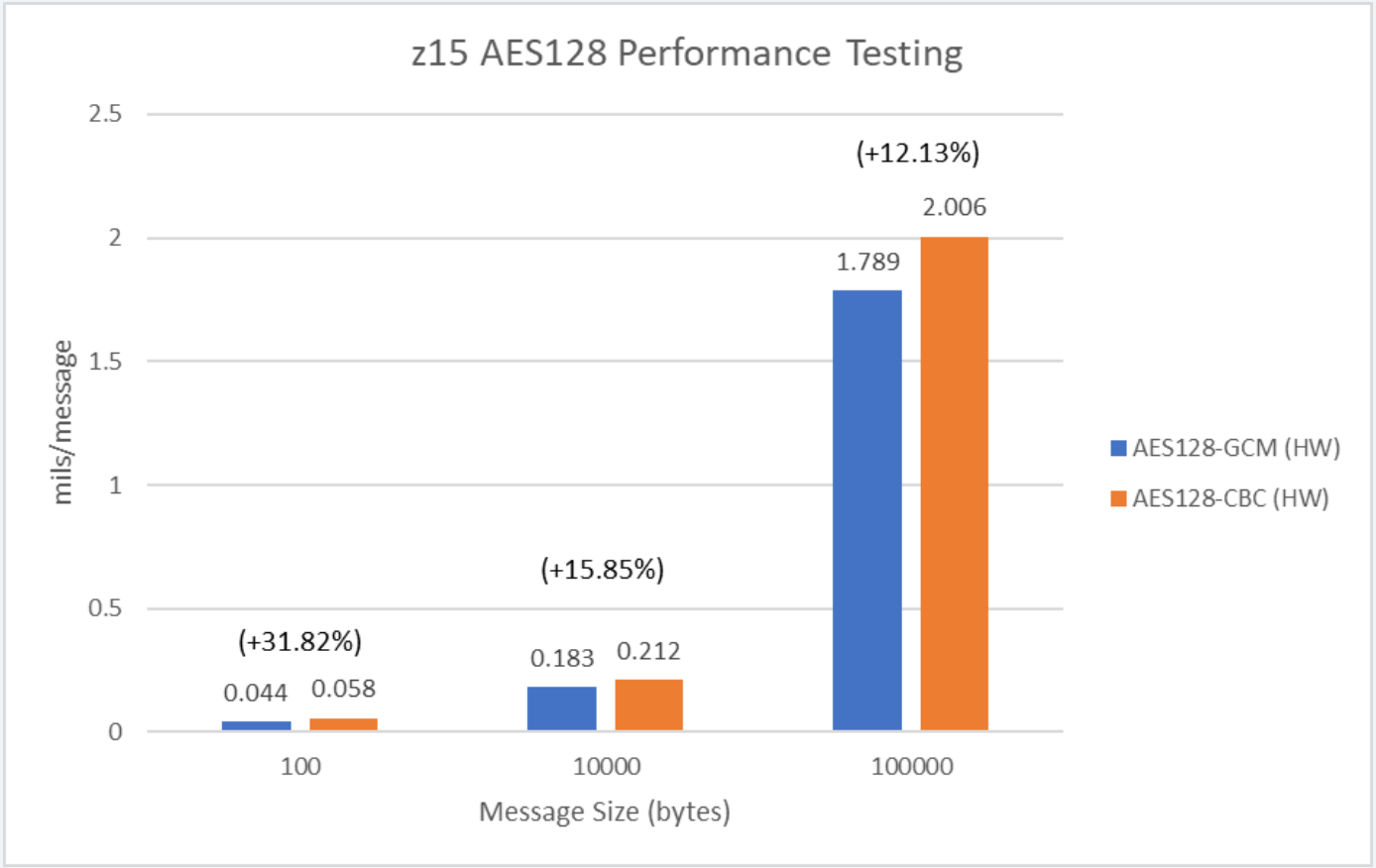
z/TPF now supports:

- AES128-GCM-SHA256
- AES256-GCM-SHA384
  - Key Exchange: Rivest Shamir Adleman (RSA)
  - Authentication: Rivest Shamir Adleman (RSA)
  - Encryption: Advanced Encryption Standard (AES) in Galois/Counter mode (GCM)
  - Message Digest: Secure Hash Algorithm xxx (SHAxxx)*

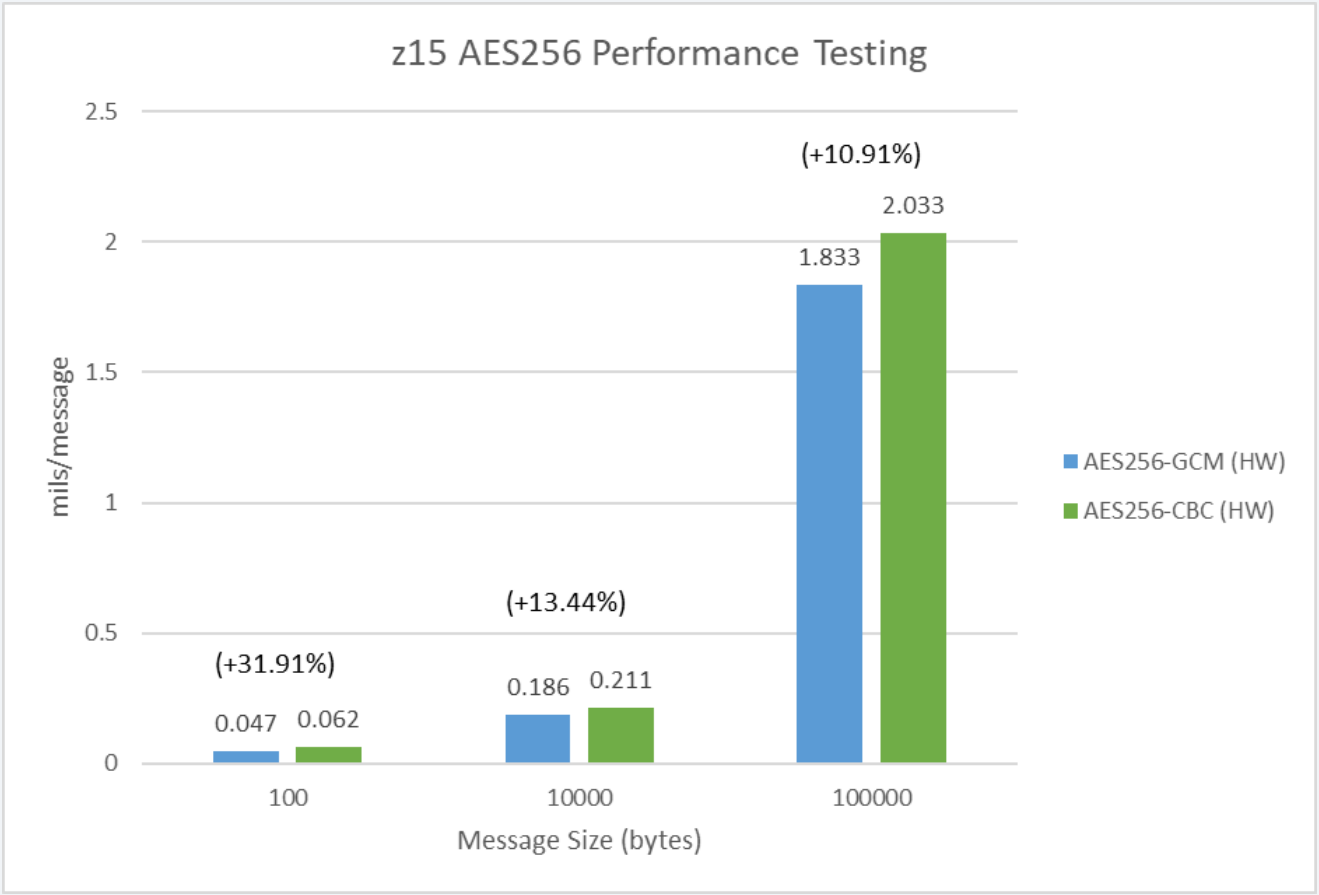
* AES in GCM has authentication built in. SHA is only used for integrity check at session startup.

Performance Testing Parameters

- TPF LPAR: server, measuring performance
- TPF LPAR: client sharing same OSA card
- Measured on z15 with 1 dedicated I-stream
- Tests were run with varying message sizes bounced back and forth
- Run with shared SSL



*Number in parentheses represents the percentage speed increase from using AES128-GCM vs AES128-CBC in hardware



*Number in parentheses represents the percentage speed increase from using AES256-GCM vs AES256-CBC in hardware

Technical Details

- Added support for two ciphers that use Diffie-Hellman Ephemeral (DHE) key exchange
  - DHE-RSA-AES128-GCM-SHA256
  - DHE-RSA-AES256-GCM-SHA384
    - Key Exchange: Ephemeral Diffie-Hellman (DHE)
    - Authentication: Rivest Shamir Adleman (RSA)
    - Encryption: Advanced Encryption Standard (AES) in Galois/Counter mode (GCM)
    - Message Digest: Secure Hash Algorithm xxx (SHAxxx)
- Ephemeral key exchange provides perfect forward secrecy

Technical Details

Perfect forward secrecy (PFS)

- Asymmetric cryptography agrees on symmetric encryption key used for a TLS session (using public/private key pair)
- Private key is single point of failure for secure communication if compromised- possible to go back and decrypt the entire key exchange conversation and obtain the encryption key for a session
- With PFS, public/private key used to exchange the secret symmetric key of an OpenSSL session is uniquely generated for each session
- Limits exposure if private key is compromised to only one session rather than data on all past (and future) sessions

Technical Details

- No hardware support for DHE key exchange – done in software
- DHE SW operations are TE eligible (under OpenSSL)
- AESxxx-GCM and SHAxxx leverage hardware if available to optimize performance for the GCM ciphers on z/TPF

Technical Details

- AES-GCM support available for z14 and up
- ZCPAC QUERY command updated to reflect support for AES-GCM ciphers.

```
User:   ZCPAC QUERY

System: CPAC0012I 07:47:41 CPACF QUERY DISPLAY

           SHA-1:       ENABLED
           DES/TDES:    ENABLED
           AES-128:     ENABLED
           SHA-256:     ENABLED
           AES-256:     ENABLED
           SHA-512:     ENABLED
           DRNG:        ENABLED
           TRNG:        ENABLED
           AES-128-GCM: ENABLED
           AES-256-GCM: ENABLED
           SHA-384:     ENABLED
         END OF DISPLAY
```

Conclusion

- New ciphers available for use with OpenSSL (including shared SSL) sessions
- Ephemeral Diffie-Hellman key exchange provides “Perfect Forward Secrecy (PFS)”
- AES in GCM ciphers performed better than analogous ciphers that use AES in CBC mode
- Improves overall security of the z/TPF system
- OpenSSL is TE eligible
- Delivered with PJ46292 in Dec 2020

ZPUBK REHASH

Background

Specifying file system directory for certificate authority certificates in SSL:

- Certificate Authority path (CAPATH) in application configuration file for SSL
- CApath parameter in SSL_CTX_load_verify_locations function

Background

- When using a directory of certificates (CAPATH) in OpenSSL, you are required to:
  1. Create a hash of the subject name of each certificate in the directory
  2. Add the hashed value as a symbolic link to the real certificate file

- Done so OpenSSL does not have to search through every certificate in the CAPATH directory

Background

- On Linux, there is a c_rehash utility that takes in a directory, creates the subject name hashes automatically, and builds the symbolic file links in that directory
- After this is done, the directory of certificates is ready to be used by OpenSSL

Problem Statement

- On z/TPF, the hash values must be obtained by displaying each certificate and using ZPUBK DISPCERT HASH command on the certificate
- Afterwards, need to issue the necessary ZFILE commands to build the symbolic link with the hash value manually
- On z/TPF, creation of the hashes and symbolic links is a manual process. Not user friendly especially if there are a lot of certificates.

Technical Details

- New z/TPF command ZPUBK REHASH
  - Command accepts a path to a directory of certificates as input
  - Hashes the subject names of the certificates of a given directory
  - Creates a symbolic link to each valid certificate
  - Removes any symbolic links that are not valid

Technical Details

Example:

The following example shows a directory of certificates before you run the ZPUBK REHASH command.

```
User:   ZFILE ls -l /certificates
System: CSMP0097I 09.16.19 CPU-B SS-BSS  SSU-HPN  IS-01
        FILE0001I 09.16.19 START OF DISPLAY FROM ls -l /certificates
        total 32
        -rw-r--r--  1 root  bin  1310 Dec 17 09:12 cert1.pem
        -rw-r--r--  1 root  bin  1602 Dec 17 09:14 cert2.pem
        -rw-r--r--  1 root  bin  1602 Dec 17 09:15 cert3.pem _
        -rw-r--r--  1 root  bin  1602 Dec 17 09:15 cert4.pem
        END OF DISPLAY+
```

Technical Details

Example:

The following example creates symbolic links and removes the invalid symbolic links in the /certificates directory.

```
User:   ZPUBK REHASH PATH-/certificates
System: CSMP0097I 09.18.26 CPU-B SS-BSS  SSU-HPN  IS-01
        PUBK0029I 09.18.26 ZPUBK REHASH PROCESSING FOR DIRECTORY /certificates COMPLETED.
```

Example:

The following example shows the /certificates directory after you run the ZPUBK REHASH command.

```
User:   ZFILE ls -l /certificates
System: CSMP0097I 09.17.39 CPU-B SS-BSS  SSU-HPN  IS-01
        FILE0001I 09.17.39 START OF DISPLAY FROM ls -l /certificates
        total 64
        -rw-r--r--  1 root  bin  1310 Dec 17 09:12 cert1.pem
        -rw-r--r--  1 root  bin  1602 Dec 17 09:14 cert2.pem
        -rw-r--r--  1 root  bin  1602 Dec 17 09:15 cert3.pem _
        -rw-r--r--  1 root  bin  1602 Dec 17 09:15 cert4.pem
        lrwxrwxrwx  1 root  bin     9 Dec 17 09:17 739ae239.0 -. cert1.pem
        lrwxrwxrwx  1 root  bin     9 Dec 17 09:17 68672434.0 -. cert2.pem
        lrwxrwxrwx  1 root  bin     9 Dec 17 09:17 65218152.1 -. cert3.pem
        lrwxrwxrwx  1 root  bin     9 Dec 17 09:17 65218152.2 -. cert4.pem
        END OF DISPLAY+
```

Conclusion

- z/TPF command ZPUBK REHASH expedites the process to create new symbolic links for a given directory containing certificate authority certificates
- Easier for customers to use directories of certificates
- Delivered with PJ46281 (January 2021)

Disclaimer

Any reference to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.

Future Deliverables

z/TPF Network Security Compliance Validation Capability

Collect audit information and provide displays to answer the following questions:

- What network ports is my z/TPF system communicating with?
- Are there applications in use that should be added to the Network Services Database (NSD)?
- Which network ports are secure? Which ones are not?
- For secure network ports, what are the network security settings (TLS version, ciphers, etc.)
- Are there any ports that allow a cipher or TLS version to be used that my company no longer considers safe?
- Which remote nodes connected to my z/TPF system are using a given old cipher or TLS version?
- Are any of my z/TPF system's certificates:
  - Going to expire soon?
  - Being used for longer than company policy allows?

Call for Sponsor Users

Will be looking for Sponsor Users to assist in design and implementation, targeting the following personas:

– z/TPF system administrators

– z/TPF operators and coverage

– z/TPF solution architects

– Security compliance officers

If you would like to be involved, contact:

Jamie Farmer (jvfarmer@us.ibm.com) or Danielle Tavella (Danielle.Tavella@ibm.com)

Q&A

Summary of Q&A from the virtual TPFUG event:

Question	Answer
Will DHE eventually be supported by hardware?	This is still being worked out with the Z hardware group. At this time, there are no plans, but hopefully we will have some level of hardware relief in the future. We are also looking at potentially pre-generating and caching DHE keys for use by transactional session startup. These can even be done on fenced Istreams, similar to system recovery boost after an IPL.
Will Certificate Expiry monitor (network security compliance future item) provide the ability to send email notifications via z/TPF JAVA mail?	We are looking at having a summary display of known certificates...but also a user exit invoked daily for each certificate with information for expiration. So you can do things like send an email when a certificate is nearing expiration.
Question regarding encryption and key rotation in z/TPF applications using tpf_cryptc (clear key APIs)	Recommendation from IBM is to use keys in the built in keystore and the tpf_encrypt_data API which has key rotation capabilities built into it.

Thank you