DB2Night Show Season 11 #217

Greg Stager
IBM Toronto Lab

Sept. 6, 2019

Light at the End of the Encrypted Tunnel

Agenda

- Learn fundamental principles of TLS encryption and certificates
- Learn all of the locations within Db2 where TLS may be used
- Learn how to configure TLS within Db2, get it working and make sure it's secure
- Learn the operational aspects of working with certificates

What is TLS?

- **Transport Layer Security**
- Provides communication security (encryption, authentication)
  - Data in transit/motion
  - Sits on top of TCP/IP (also UDP and other transports)
- Often the “S” in well know protocols
  - HTTPS
  - LDAPS
  - FTPS
- Originally developed as SSL (Secure Socket Layer) by Netscape in 1995
  - The terms SSL and TLS are often used interchangeably

Security of Various Versions

- TLS 1.0/1.1 - 1999/2006
  - Insecure and known vulnerabilities – don't use it
  - Industry is slowly removing it from products (gone from browsers in 2020)
- TLS 1.2 - 2008
  - Current standard, considered secure
  - Supported in Db2:
    - V9.7 FP9, v10.1 FP4, v10.5 FP4, v11 GA

Must be configured
Not the default!

- TLS 1.3 - 2018
  - Improves security of TLS 1.2
  - Not supported yet in Db2, only beginning to be seen in web browsers

Basics of Public Key (Asymmetric) Cryptography

- Fancy math creates a linked public and private key pair
  - As the name suggests:
    - you keep one private (secret)
    - make the other public
  - Fancy math = prime factorization, elliptic curves etc.

Acronym Soup

RSA

DH

ECC

DSA

What you encrypt with one key of the pair

You can only decrypt with the other

Digital Signatures and Key Distribution

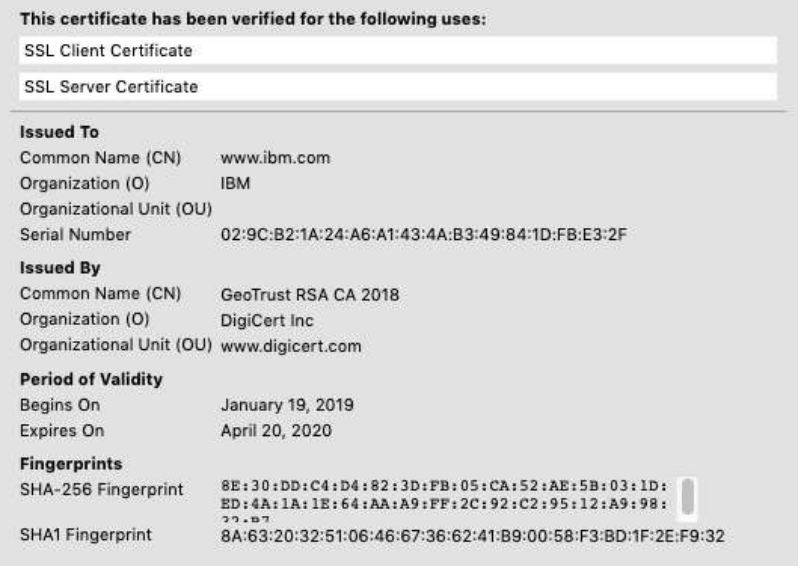
- Signing
  - Encrypting a hash of the message with the private key creates a digital signature
  - You want everyone to be able to decrypt it!
  - Proves possession of the private key that is paired with the public key

- But how do you know you have the correct public key?

- SSH - public key copied to ~/.ssh/authorized_keys – on every server
- PGP – web of trust

What is a certificate?

- A certificate is a public key and associated metadata that has been signed
- Who is it signed by?
  - Anyone!
  - Only useful if it's someone you trust
- Issuer attesting the identity of the certificate holder
  - They have performed some level of metadata validation



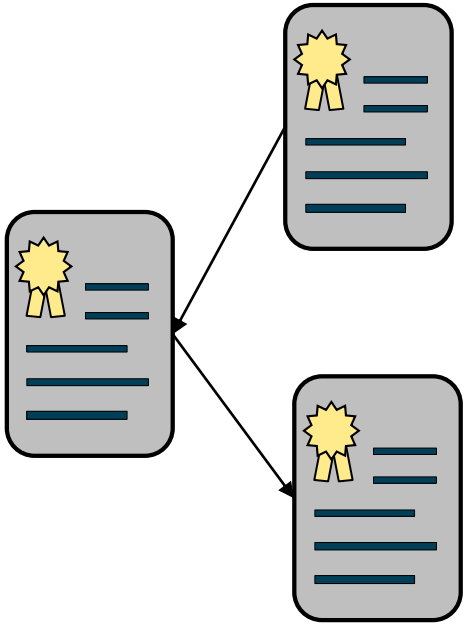
How do certificates help us?

- Reduces the complexity of the key distribution problem
  - You only need to broadly distribute the issuer public key
    - Aka the root CA certificate

- When given a never before seen certificate:
  - Validate it's been signed by the trusted issuer so you can
    - Trust the contents of the certificate

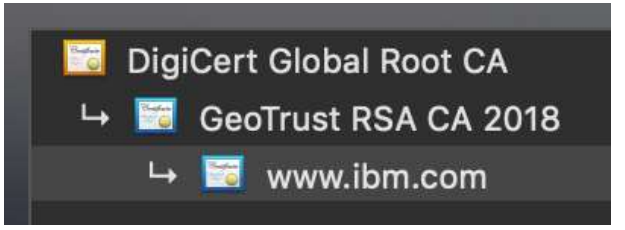
Tree of certificates

- Issuer and certificate holder can be hierarchical in nature
  - A signs B signs C
- The top level (A) is known as the **root certificate**
  - B is known as an **intermediate root certificate**
  - C is an **end entity (endpoint) certificate**
- Public Key Infrastructure (PKI) is the management of certificates and binding with identity



Certificate Authorities

- Certificate Authority (CA)
  - Formal organization that issues certificates



- Public companies
  - Examples: Digicert, Let's Encrypt, Comodo, Geotrust, GoDaddy, and on and on
  - Their focus is the web
  - Fee based (except for Let's Encrypt)

- Internal Enterprise CA
  - Many companies have security teams that act as a certificate authority
  - Various software packages can be used to implement a CA
    - Microsoft AD Certificate Services, OpenSSL, DogTag and many others

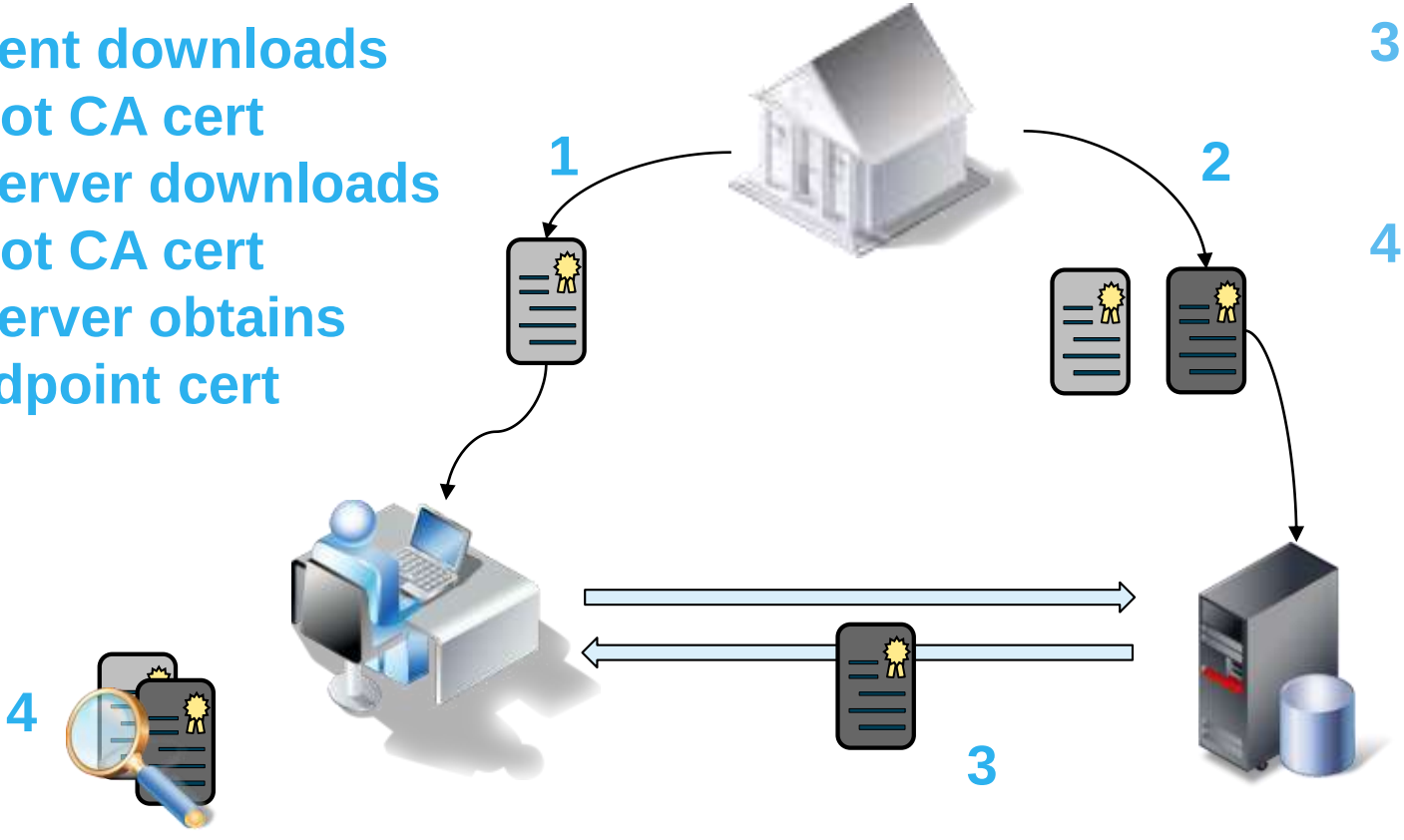
How do I indicate I trust someone (typically a CA)?

- Add their public key (in the form of a certificate) to a **truststore**
  - A keystore holding trusted root certificates which will be used to validate other certificates
- In the case of the web, a combination of your OS and web browser maintain a list of hundreds of trusted root and intermediate certificates
- For Db2 we will be manually adding certificates to a keystore we create

Certificates in use

Setup

1. Client downloads Root CA cert
2. - Server downloads Root CA cert
   - Server obtains endpoint cert



Connection

3. Server sends endpoint cert to client
4. Client validates server cert using Root CA cert

Agenda

- Learn fundamental principles of TLS encryption and certificates
- Learn all of the locations within Db2 where TLS may be used
- Learn how to configure TLS within Db2, get it working and make sure it's secure
- Learn the operational aspects of working with certificates

Where does Db2 use TLS?

- The good news is that TLS is seeing widespread adoption in Db2
  - Client – server (DRDA)
  - HADR – between primary and standby
  - Federation
    - For DRDA and ODBC wrappers
  - KMIP
    - Native Encryption Key Managers
  - LDAP
    - Authentication plugins

Note: TLS is not supported between members in DPF or pureScale

Use TLS instead of DATA_ENCRYPT
- **Uses insecure algorithm (DES)**
- **Deprecated in 11.5**

- Today we will talk about configuration of client-server TLS.
  - Backup material has configuration details for other uses.

How does Db2 use TLS?

- TLS is a really complicated protocol with many optional features

- **Db2 uses TLS for encryption of data over the network**
  - Client supports certificate authentication against Db2 for Z/OS

- Db2 does not support the following optional features
  - Server hostname validation (CN or SAN)
  - Certificate revocation (CRL or OCSP)
  - PSK – Pre Shared Keys

How do I manage the use of TLS?

- The bad news is that each use of TLS is largely managed independently and must be configured separately
  - In practice you can use a single keystore file for the various uses

- One unifying aspect is that they all require a **keystore**
  - File on disk
  - Contains
    - Certificates
    - Private keys
    - Secret keys (for native encryption)

Types of keystores

- .kdb file
  - IBM (GSKit) proprietary keystore file
- .p12/.pfx file
  - PKCS #12 file format for cryptographic objects
  - Industry standard
- .jks
  - Java keystore
- .pem/.csr/.arm/.cer/.crt
  - Base-64 ASCII encoding of a certificate
- Microsoft Certificate Store
- IBMCAC
  - For US DOD Common Access Cards
- All of the files are useable cross-platform

Supported uses of various keystore type

	DRDA– Server	DRDA – CLI Client	DRDA – Java Client	HADR	Federation	KMIP	LDAP plugins
.kdb	✔	✔		✔	✔	✔	✔
.p12	✔	✔		✔	✔	✔	✔
.jks			✔		✔		
.pem/.cer		✔ [1]	✔ [1]		✔ [1]		
MSCS	✔	✔	✔				
IBMCAC			✔ [2]				

Types of Certificate Files - used for exchanging certificates

- .p12 – encrypted file, can contain many certificates and private keys

- Privacy Enhanced Mail format
  - Usually has many file extensions
    - .pem, .arm, .cer, .crt, .cert, .csr
    - Can be encrypted, but currently not supported
    - BASE64 encoded (text based)

```
-----BEGIN CERTIFICATE-----
…
-----END CERTIFICATE-----
```

- PKCS #7
  - .p7b, .p7s
  - Very similar to PEM

```
-----BEGIN PKCS7-----
…
-----END PKCS7-----
```

Methods to create keystores

- From your security team
- IBM GSKit
  - Command line tool (gsk8capicmd) shipped with Db2 for managing keystores
- IBM Keyman
  - Java version of gsk8capicmd, usage is almost identical
  - Not shipped with Db2, but some other IBM products
- Java keytool
  - Oracle tool shipped with Java
- Don’t create one
  - Client “simplified SSL setup” allows you to use a .cer file without a keystore
  - Use the Windows Certificate Store

Keystore from your security team

- Your enterprise security team may just give you a ready made keystore file with appropriate certificates already in it
  - For clients, this often works well as you only need the root CA certificate, which usually has a very long expiry time
  - For servers, you will need to deal with regularly expiring server certificates, which means multiple certificates in the file, so you need to become familiar with creating your own, import new certs etc.

What is GSKit?

- IBM Global Security Kit
  - Encryption/TLS component part of most IBM products
  - FIPS 140-2 certified library
  - gsk8capicmd_64 command line tool for managing keystores
    - Docs Online: ftp://ftp.software.ibm.com/software/webserver/appserv/library/v80/GSK_CapiCmd_UserGuide.pdf
      - Link can also be found in Db2 Docs (Configuring SSL in non-Java clients)

- Included with Db2 Server

- Client includes enough to use TLS
  - Only missing the gsk8capicmd_64 command line tool
  - Can download and install from Passport Advantage

Create keystore using GSKit

- **`gsk8capicmd_64 -keydb -create -db mykeystore.p12 -pw <yourpasswordhere> -strong -stash`**

- -strong: force use of a strong password
- -stash: create stash
  - An obfuscated form of the password in a file

- At the server, somewhere in the instance owner's home directory is a good location for this file

What do I need in my keystore file?

	Root CA certificate	Intermediate CA certificate	End Point Certificate	End Point Private Key
Db2 Server	✔	✔	✔ (Server cert)	✔ (Server pk)
DRDA Client	✔			
DRDA Client with certificate authentication	✔	✔	✔ (Client cert)	✔ (Client pk)
HADR primary or standby	✔	✔	✔ (own cert)	✔ (own pk)
Federation DRDA Client	✔			
KMIP	✔	✔	✔ (Client cert)	✔ (Client pk)
LDAP Plugin	✔			

How do I get a certificate? – You ask

1. Certificate Signing Request (CSR)

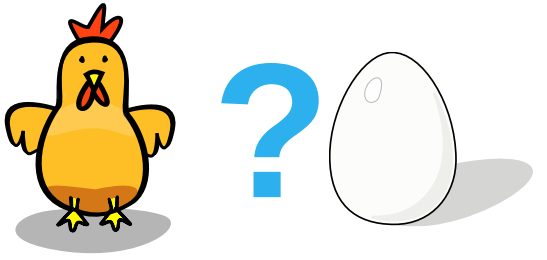
- Generate public/private key pair
- Send public key, along with identity information to CA
- CA validates identity information
- CA signs public key and metadata to create your certificate
- CSR allows you to keep control of private key

2. Or your security team may give you everything

- Private key
- Certificate
- Root certificate chain

Self-Signed Certificates - Perfect for the DIY DBA

- It is possible to sign your own certificate!
  - In fact ALL root certificates are self-signed
  - Issued To (Subject) is the same as Issued By
- But who will trust this certificate?
  - Only the client you explicitly tell to
- Back to the problem of key distribution
  - Typically you will have a different certificate for every server
  - Usually only used for small deployments



How do I import a root CA certificate into my keystore?

- You will almost always be able to download or be given your root CA's certificate as these are meant to be public

- The "**`-cert -add`**" option adds it to the keystore as a trusted cert

- **`gsk8capicmd_64 -cert -add -db mykeystore.p12 -stashed -file RootCA.cer -label MyRootCA`**

- This will add the trusted certificate found in RootCA.cer and give it the label MyRootCA

How do I generated a CSR for an endpoint certificate?

Endpoint being client or server

- ```
  gsk8capicmd_64 -certreq -create -db mykeystore.p12 -stashed
    -label db2Server -size 2048 -sigalg sha256
    -dn "CN=<hostname>,O=<company>,OU=Db2,L=Markham,ST=ON,C=CA”
    -target db2server.csr
  ```

- This will create an RSA public and private key, and place the public key and DN into db2server.csr
- The label of the certificate once received will be “db2Server”
- Send db2server.csr to your CA for them to sign
- See GSKit docs for other options
  ```

My CA signed my CSR, now what?

- Don’t “-add” your cert – that will make it a CA cert
- You need to “-receive” it so that it’s matched with the CSR
- **`gsk8capicmd_64 -cert -receive -db mykeystore.p12 -stashed -file db2server.cer`**
- This command imports the signed certificate in db2server.cer and associates with the private key generated during the CSR process

How do I import a root and endpoint certificate?

- Your security team has given everything to you all at once
  - If root/intermediate are in separate files, follow previous steps first

- If everything is in one file, the order is important
  - The private key MUST appear after it's certificate
  - Private key must not be encrypted
  - Root and intermediate CA certs should appear first if present
  - Have the file name *.pem or add -type pkcs7 to command:

- **`gsk8capicmd_64 -cert -import -file certs.pem -target mykeystore.p12 -target_stashed`**

Importing endpoint certificate continued

- Importing multiple items is broken prior to Db2 10.5 FP10 / 11.1.3.3
  - https://www-01.ibm.com/support/docview.wss?uid=ibm10730657

- multiple certificates in the .pem file? You can't label them on import
  - The label will be the DN
  - Use the –cert –rename command
  - **gsk8capicmd_64 -cert -rename -db mykeystore.p12 -stashed -label "CN=hostname,O=company,OU=Db2,L=Markham,ST=ON,C=CA" -newlabel NewServerCert**

- If you only had a single cert and private key, add to import command:
  - -label NewServerCert

How do I know what is in my keystore?

- **`gsk8capicmd_64 -cert -list -db mykeystore.p12 -stashed`**

```
Certificates found
* default, - personal, ! trusted, # secret key
!  MyRootCA
-  Db2Server
```

- Legend
  - "!" a certificate that is being trusted to sign other certificates.
    - Should only be root and intermediate CA certs (expect for self-signed certs)
  - "-" an end-point (or personal) certificate
    - Only end-point certificates are valid to specify in SSL_SVR_LABEL
  - "*" default certificates are deprecated by GSKit
  - "#" secret key – used for native encryption, not SSL

How do I find out the details of a certificate in my keystore?

- **gsk8capicmd_64 -cert -details -label db2Server -db mykeystore.p12 -stashed**
- Label : db2Server
  Key Size : 2048
  Version : X509 V3
  Serial : 537cd1057a03f56e
  Issuer :CN=hostname,O=company,OU=Db2,L=Markham,ST=ON,C=CA
  Subject :CN=hostname,O=company,OU=Db2,L=Markham,ST=ON,C=CA
  Not Before : March 30, 2019 3:14:15 PM EDT
  Not After : March 30, 2020 3:14:15 PM EDT

…

Agenda

- Learn fundamental principles of TLS encryption and certificates
- Learn all of the locations within Db2 where TLS may be used
- Learn how to configure TLS within Db2, get it working and make sure it's secure
- Learn the operational aspects of working with certificates

Configuring Client-Server communications – At the server

- Update these DBM CFG values
  - SSL_SVR_KEYDB – full path to your server's keystore
  - SSL_SVR_STASH – full path to the stash file for you keystore
  - SSL_SVR_LABEL – the label in the keystore for the server's certificate
  - SSL_VERSIONS – set to TLSV12 (you should not be using the default TLSV1)
  - SSL_SVCENAME – Port for Db2 to listen to for TLS connections
    - No standard for the port, often see customers using 50001
    - Don't use 443 – that is reserved for HTTPS (web sites)
  - SSL_CIPHERSPECS – Generally leave blank
- db2set DB2COMM=SSL[,TCPIP]
  - You can have both TLS and TCP/IP in use at once
- Restart the instance – more on that later

Configuring Client-Server communications – CLI based clients

- Instance Based Clients
- CATALOG TCPIP NODE … SECURITY SSL

- Update these DBM CFG parameters
  - SSL_CLNT_KEYDB – full path to client keystore file
  - SSL_CLNT_STASH – full path to stash file for keystore

Configuring Client-Server communications – CLI based clients

- IBM Data Server Driver clients
- Set the following parameters in connection string, db2cli.ini etc
  - Security=SSL (db2cli.ini)
    or
  - SecurityTransportMode=SSL (db2dsdriver.cfg)

  - SSLClientKeystoredb=full path to client keystore file
  - SSLClientKeystash=full path to stash file for keystore

Configuring Client-Server communications – Java clients

- See the documentation about configuring runtime environment with JSSE Provider:
- https://www.ibm.com/support/knowledgecenter/en/SSEPGG_11.1.0/com.ibm.db2.luw.apdv.java.doc/src/tpc/imjcc_t0054066.html
- At a minimum, turn on TLS for the connection:
  - properties.put("sslConnection", "true");
- You can switch from the default trust store with the parameter pointing to your .jks keystore file:
  - DB2BaseDataSource.sslTrustStoreLocation
  - db2.jcc.sslTrustStoreLocation
    - Also set sslTrustStorePassword with the keystore password

Configuring Client-Server communications - Simplified Setup

- Instead of creating a keystore, just tell the client where the root CA certificate is loaded in a .cer (.pem, .arm etc) file
  - New in Db2 10.5 FP5
- CLI Based clients:
  - Set SSLServerCertificate keyword to the .cer file
  - Despite the name, this is the Root CA certificate (or server self-signed cert)
- Java
  - Set DB2BaseDataSource.sslCertLocation on a Connection or DataSource to the .cer file

Special Case – connecting to Db2 (Warehouse) on Cloud

- We've taken the simplified SSL setup a step further for cloud
  - The CLI client ships with the Digicert Root CA cert used by Db2 on Cloud
  - It's already there in the default Java truststore

- Just configure the client to use SSL instead of TCP/IP and the appropriate port (50001)
  - CLI: -parameter "SecurityTransportMode=SSL"
  - Java: sslConnection=true

- No need for a keystore or to specify a root certificate file

Agenda

- Learn fundamental principles of TLS encryption and certificates
- Learn all of the locations within Db2 where TLS may be used
- Learn how to configure TLS within Db2, get it working and make sure it's secure
- Learn the operational aspects of working with certificates

Expiring Certificates – or - didn't I just change my password?

- Unfortunately certificates expire
  - Once they do, new connections will fail
    - Existing connections will still work
  - Typically 1-2 year lifespan for endpoint certificates

- "Renewing" a certificate is nothing more than generating a new certificate for the same server with updated expiry dates
  - There is no "renew" command

- Keystores can hold multiple certificates, so place new certificate there with a new label while the old one is in use

Expiring certificates – I have to do what?

- If using CA signed certificates, only change is at the server
  - Client should already have the Root CA certificate it needs
- If using self-signed certificates, you will need to distribute the certificate to the client
- At the Db2 server, **an instance restart is required**
  - HADR – just requires a new connection
    - deactivate/activate db at standby
- We understand the frustration this causes
  - We are looking at a fix, but can’t comment on if/when it might happen

How do I know when my certificates will expire

- There is no administrative SQL interface

- You can use gsk8capicmd_64 to examine the certificate details
  - **gsk8capicmd_64 -cert -details -label db2SelfSigned -db mykeystore.p12 –stashed | grep "Not After"**

    Not After : March 30, 2020 3:14:15 PM EDT

- You can use OpenSSL to remotely query the certificate the server uses
  - **echo | openssl s_client -connect <hostname>:<port> 2>/dev/null | openssl x509 -noout -dates**
  - notBefore=Mar 30 19:14:15 2019 GMT
    notAfter=Mar 30 19:14:15 2020 GMT

Performance

- In general use, TLS has little performance overhead
  - Every setup is different, your mileage may vary
- Largest impact is two extra flows to establish connection
  - Very brief connections will have higher latency
- Hardware acceleration helps
  - Modern Intel CPUs (Intel AES-NI) and Power 8
  - https://www.idug.org/p/bl/et/blogaid=546
- On AIX? – read this malloc technote:
  - https://www-01.ibm.com/support/docview.wss?uid=ibm10741831

Monitoring

- Two ways to check if client connection to server is using TLS

- MON_GET_CONNECTION – the CLIENT_PROTOCOL column
  - select client_protocol, client_ipaddr, system_auth_id
    from table(mon_get_connection(NULL, -2))

- db2pd –applications
  - EncryptionLvl of ‘High’ means TLS is being used

- Not possible to determine whether TLS v1.1 or 1.2 was chosen, nor what particular ciphers were negotiated

Backup of keystore

- The keystore file and stash file are considered external to the instance
- **You are required to back them up**
- Good idea to independently keep track of the password for the keystore in case something happens to the stash file
- You can generally re-create the contents of the keystore if it is damaged
  - will take an outage while doing so
  - Regenerate self signed certificates, get new CSRs signed
  - If using same keystore for native encryption keys, absolutely critical that the file is backed up – you risk losing all your data

DPF and pureScale and HADR

- Same label will be used on each member (from DBM CFG)
- You can have a shared keystore or make it independent
  - Think about availability issues if using a single file
- You can use the same certificate, or have a unique one for each server (would require independent keystores in that case)
- The CF does not need a certificate

Problem Determination

- Most problems can be traced back to setup issues
- GSKit Errors in SSL functions (like the 420 and 402 below):
- Client – SQL30081N with
- `"sqlccSSLSocketSetup". Protocol specific error code(s): "420", "*", "*".`
- Server – db2diag.log message like
- `FUNCTION: DB2 UDB, common communication, sqlccMapSSLErrorToDB2Error, probe:30`
  `MESSAGE : DIA3604E The SSL function "gsk_secure_soc_init" failed with the`
  `return code "402" in "sqlccSSLSocketSetup".`
- List of GSKit errors:
- https://www.ibm.com/support/knowledgecenter/en/SSVJJU_6.3.0/com.ibm.IBMDS.doc/progref506.htm?amp;view=kc&origURL=SSVJJU_6.3.0/com.ibm.IBMDS.doc/progref506.htm

GSKit Errors

- 420 at the client
  - The server closed the connection, it didn't like something
  - Need to look at the db2diag.log on the server for the reason

- 407 – GSK_ERROR_BAD_KEYFILE_LABEL
  - The label can't be found in the keystore – check for typos!
  - Don't quote labels in config files (shell it's ok)
  - Look at the contents of your keystore (-cert –list) to ensure your certificate is listed as "-personal" and not "! Trusted"
    - This means you have added it incorrectly to the keystore
    - This is the most common error we see

GSKit Errors

- 402 - GSK_ERROR_NO_CIPHERS
  - Client is not at a fixpack level supporting TLS 1.2
  - Server has misconfigured SSL_CIPHERSPECS for it's certificate (RSA vs ECC)
- 408 – GSK_ERROR_BAD_KEYFILE_PASSWORD
  - The keystore password is bad
  - Check previously mentioned stash file issue
- 410 – GSK_ERROR_BAD_MESSAGE
  - You are not connecting to an SSL port
  - Something like a firewall is returning bad data

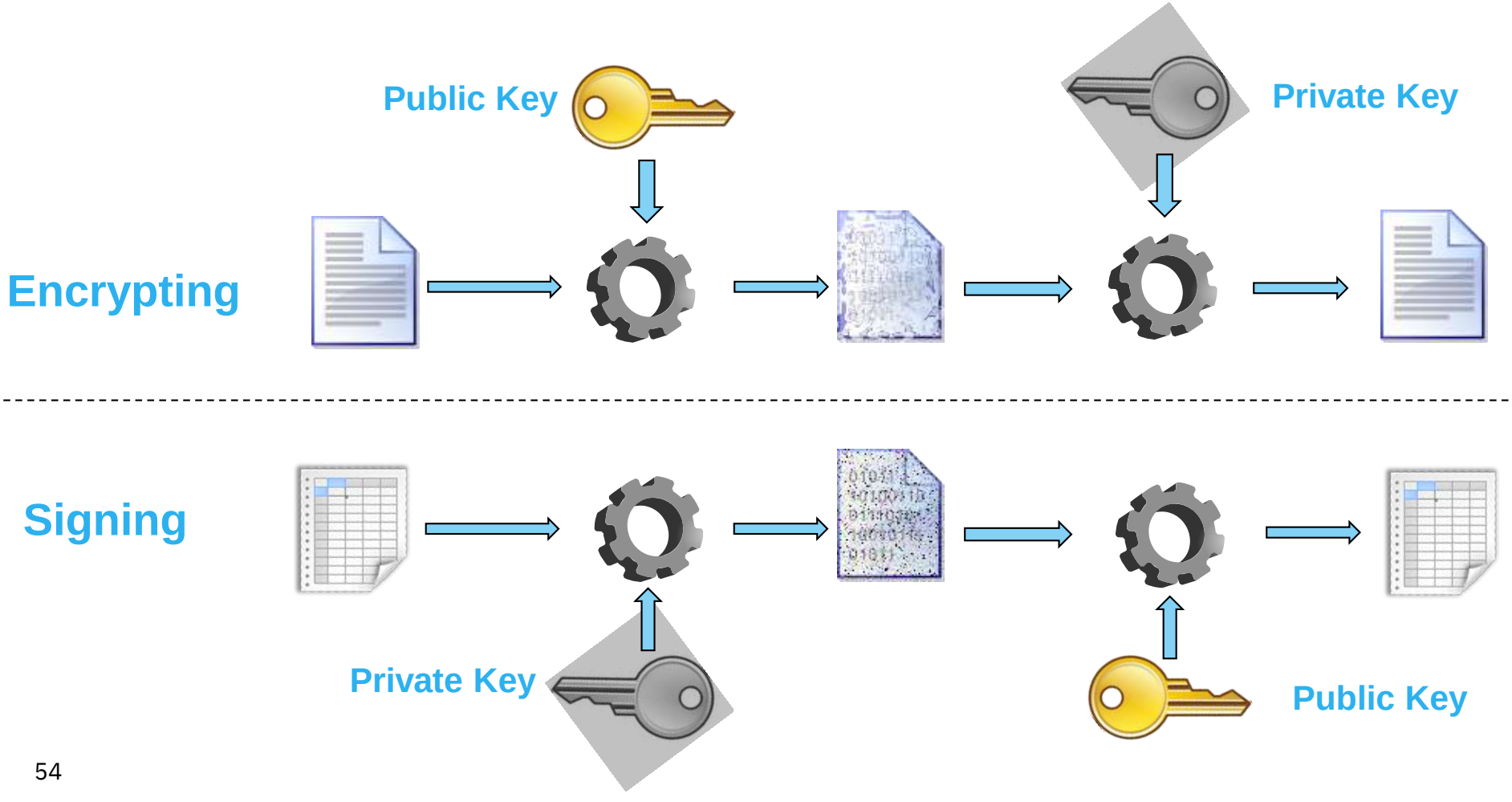
Greg Stager
gstager@ca.ibm.com

Thank You

Additional Material

- There is too much to TLS to cover in 1 hour
- The following is some additional reference and background material which may provide a more complete picture of TLS

Public Key Cryptography



History of TLS

- Originally developed as SSL (Secure Socket Layer) by Netscape in 1995
  - The terms SSL and TLS often used interchangeably
  - SSL v1-v3
- IETF took ownership and it became an internet standard as TLS 1.0

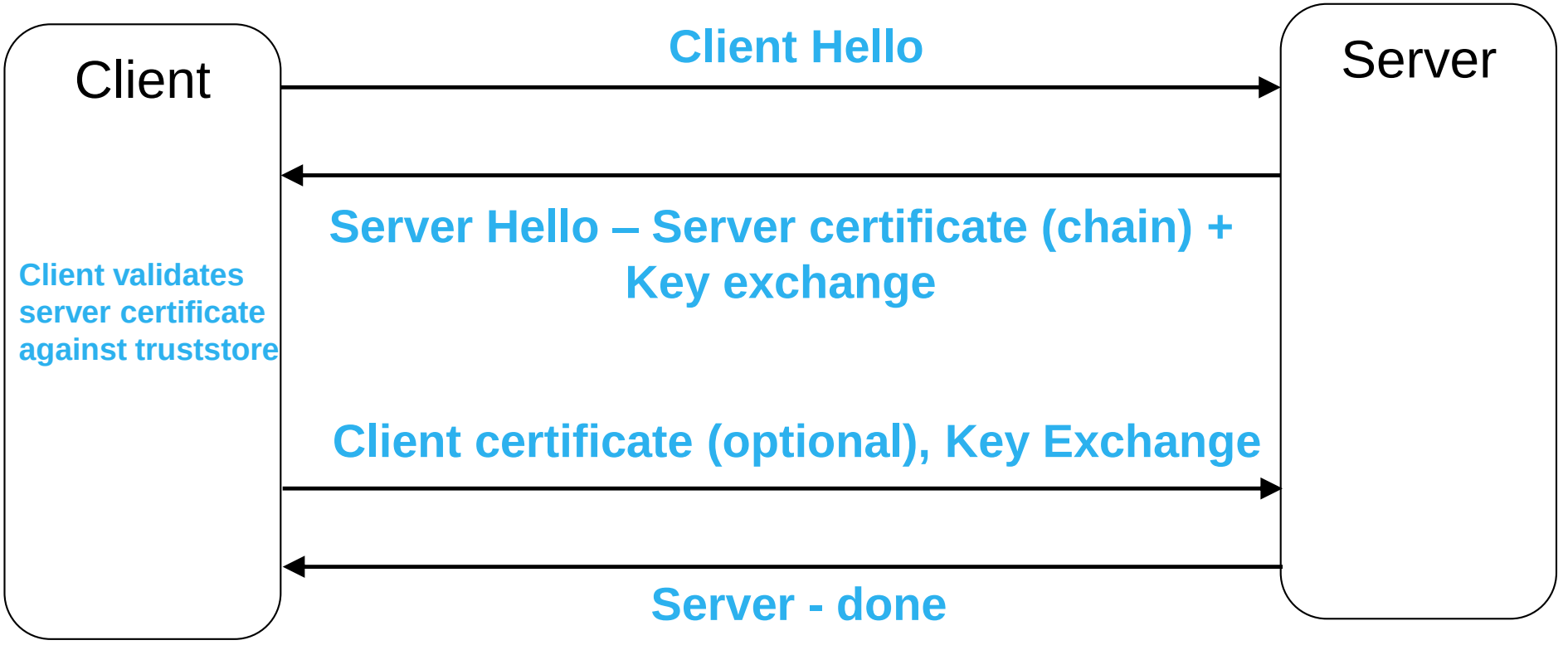
1.0	1.1	1.2	1.3
1999	2006	2008	2018
RFC 2246	RFC 4346	RFC 5246	RFC 8446

How does TLS use certificates – why do I even need one

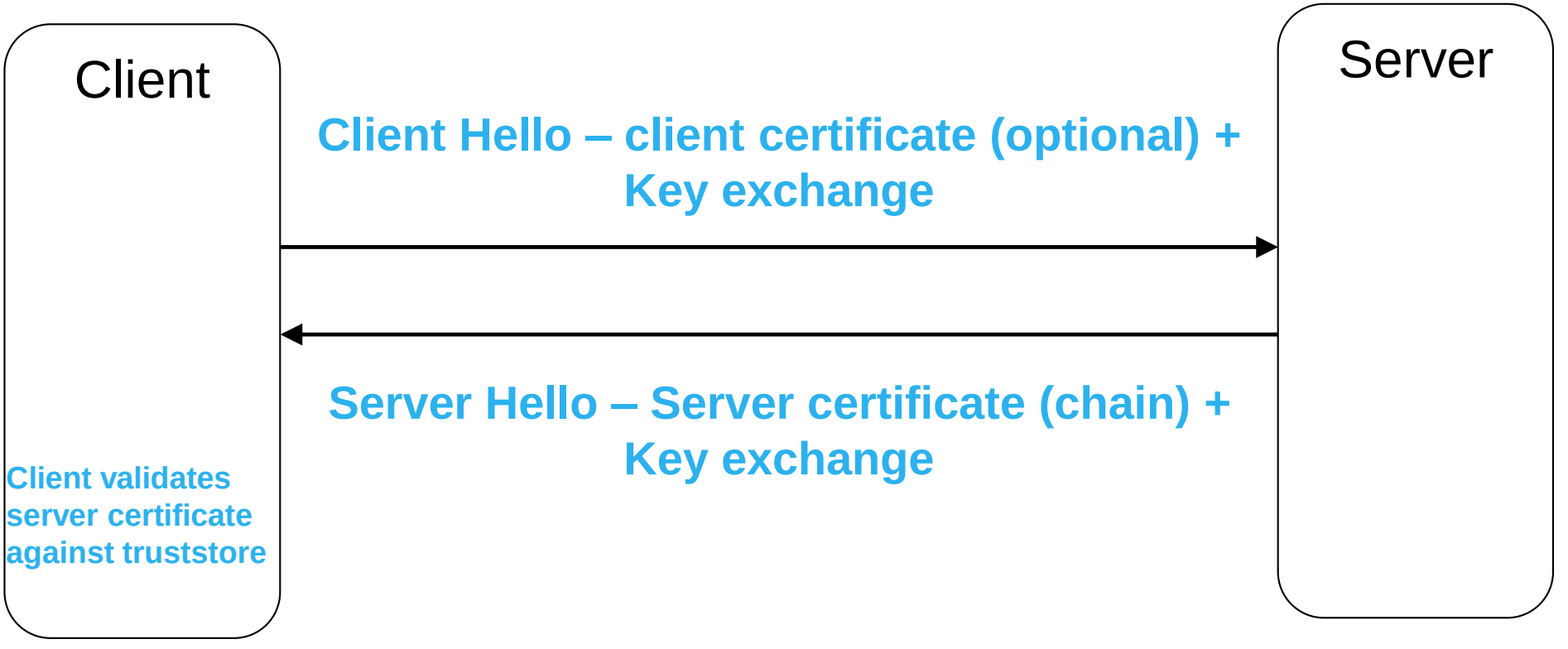
- TLS requires certificates for the following
  - Key Exchange
  - Server Authentication
  - Client Authentication (optional)

- Particularly for the web, allows a small set of root CA certificates to be trusted, but yet encryption to work with a very large number of websites

TLS 1.2 Handshake (simplified) – establishing a connection

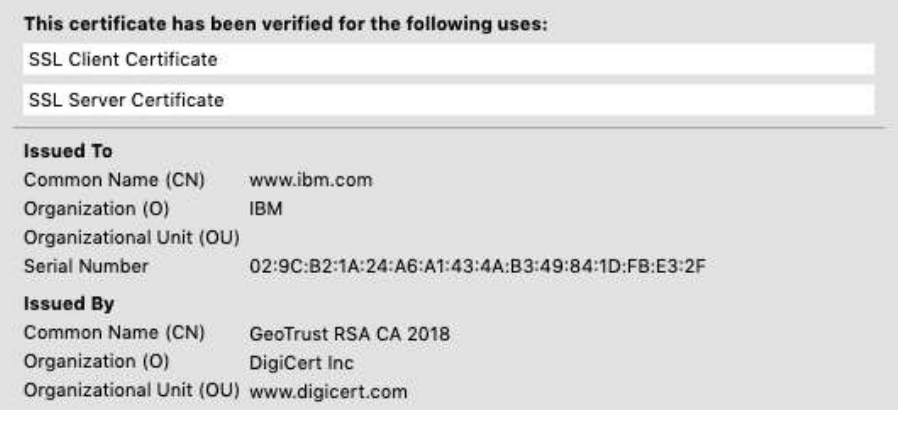


TLS 1.3 Handshake (simplified) – establishing a connection
A glimpse into the future



X.509v3 Certificate Details

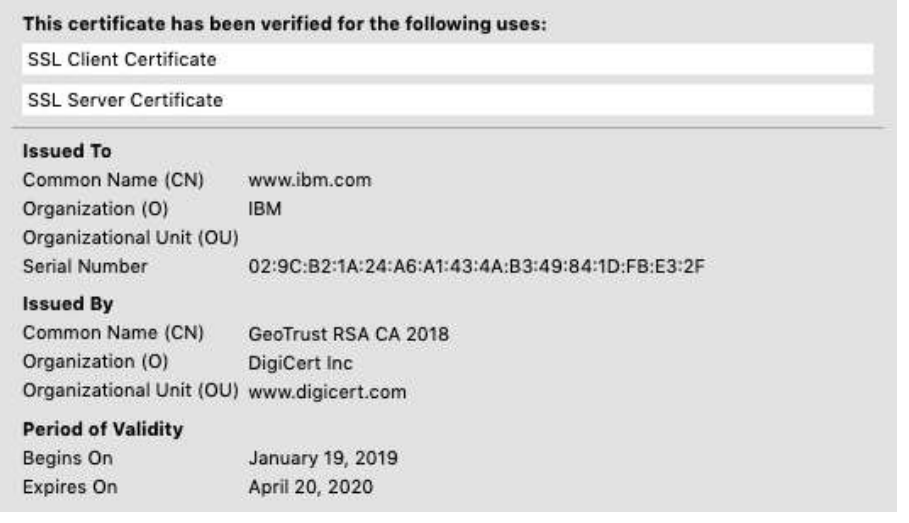
- Distinguished Name (DN)
  - Common name (CN)
    - Usually hostname
  - Organization (O)
    - Usually company name
  - Organizational Unit (OU)
    - Department etc.
  - Country



- Additional hostnames can be in the Subject Alternate Name (SAN)

X.509v3 Certificate Details

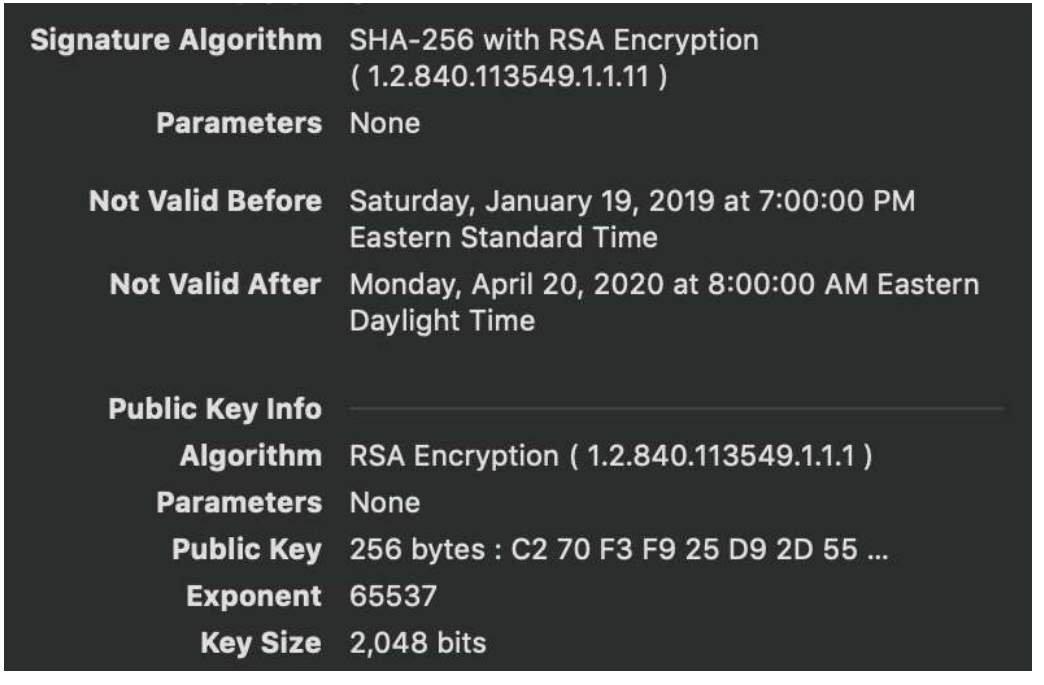
- Date Validity
  - Not Before
  - Not After
    - Certificates Expire!



- Root certificates are usually good for many (20) years
- Server certificates are often only good for 1 or 2 years

X.509v3 Certificate Details – Key sizes and algorithms

- Signature Algorithm
  - Hash + Encryption
  - Should be SHA 2 family +
    - RSA or ECDSA
- Public Key Info
  - Algorithm
    - Should be RSA or ECC
  - Key Size
    - 2048 bits or greater



X.509v3 Certificate Details – Basic Constraint

- CA certificate:
  - `X509v3 Basic Constraints:` **`CRITICAL`**
  - **`CA:TRUE`**
- End Point certificate:
  - `X509v3 Basic Constraints:`
  - `CA:FALSE`

- Some CAs forget to mark CA:TRUE certificates as CRITICAL
  - This causes warnings when importing the certificate into a keystore
  - Can be errors for KMIP authentication – more on this later

NIST SP800-131A compliance

- To be NIST SP800-131A compliant
  - At the server
    - Set SSL_VERSIONS=TLSV12
    - Use certificates with
      - Keysize 2048 or greater
      - SHA 2 family hash
- For LDAP
  - Ensure SECURITY_PROTOCOL=TLSV12

Downloading GSKit and documentation

- See this link about how to download GSKit (only if you need it!)
  - https://www-01.ibm.com/support/docview.wss?uid=swg21432998

- Documentation for this tool:
  - ftp://ftp.software.ibm.com/software/webserver/appserv/library/v80/GSK_CapiCmd_UserGuide.pdf
  - Link can also be found in Db2 Docs (Configuring SSL in non-Java clients)
  - You know you have the latest if the title page says this (or later):
    - Edition 23 March 2018
    - Until recently it was a very old version

Some pre-reqs to using GSKit

- Add it to your $PATH
  - UNIX/Linux
    - $HOME/sqllib/gskit/bin/
  - Windows
    - c:\Program Files\IBM\gsk8\bin
    - c:\Program Files\IBM\gsk8\lib64
- Update $LD_LIBRARY_PATH
  - Add $HOME/sqllib/lib64/gskit
  - LIBPATH on AIX
- On 10.5 FP11 and V11.1.4.5
  - This is done for you as part of $HOME/sqllib/db2profile

- Decide on a location of the keystore
  - Instance owner's home directory is a good location
  - It needs to be protected with appropriate file permissions
    - Read/write for instance owner only

What is a stash file

- A stash file is an obfuscated form of the keystore password
- Allows Db2 to access the keystore file without user intervention, but still prevent the files contents from being casually read
- Watch out, the stash file format changed
  - Db2 10.5 FP9 and V11.1.3.3 and onwards use the new format
    - These version can read the old format
  - Older versions can not read the new format
    - Primarily a problem if you copy keystores between machines
- https://www-01.ibm.com/support/docview.wss?uid=swg22014693

How do I generate a self-signed certificate

- Almost the same as generating a CSR
  - -cert instead of –certreq
  - No -target output option

- **gsk8capicmd_64 -cert -create -db mykeystore.p12 –stashed -label db2SelfSigned -size 2048 -sigalg sha256 –dn "CN=<hostname>,O=<company>,OU=Db2,L=Markham,ST=ON,C=CA”**

- This will generate a private key and self-signed certificate in the keystore with label db2SelfSigned

How do I export self-signed certificate for use at the client

- You need to distribute the self-signed certificate (not private key) to the clients
- They treat it as if it's a Root CA certificate
- **`gsk8capicmd_64 -cert -extract -db mykeystore.p12 -stashed -label db2SelfSigned -target db2SelfSigned.cer -format ascii`**
- Note – use extract
  - extract – does not have the private key
  - export – contains the private key, not what you want in this scenario

- Send the file db2SelfSigned.cer to clients for them to "-add"

I forget, do I have an outstanding CSR?

- You can list what CSR requests are outstanding
  - also delete or recreate them

- **`gsk8capicmd_64 -certreq -list -db mykeystore.p12 -stashed`**

- **`Certificates requests found`**
  `db2Server`

keytool – GSKit was fun, lets do it again this time in Java

- Consult Oracle docs for full details
  - Equivalent commands for all gsk8capicmd_64 commands shown above
- Java has a default truststore for CA certs called cacerts
  - In $JAVA_HOME/jre/lib/security

- Import a root CA cert into it:
- **keytool -import -keystore cacerts -file RootCA.cer -alias MyRootCA -storepass <password>**

Configuring Client-Server communications – At the server

- The DBM CFG parameter SSL_CIPHERSPECS
  - Allows you to limit the ciphers the server will accept
  - **The default (NULL) is fine** unless future problems with a specific algorithm
  - TLS 1.3 has greatly reduced the list, here's what is equivalent for TLS 1.2
    - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
  - Note the 2nd parameter (RSA or ECDSA) must match the type of public key used in the certificates (RSA or ECC) or else you will get an error
  - Break down the parameter:

Key Exchange	Authentication	Block Cipher	Mode	HMAC
ECDHE	RSA	AES_256	GCM	SHA384

Special Case – Db2 Connect Server

- TLS for connections into the Db2 Connect server is configured independently of connection outbound to Db2 for Z/OS or i.

- You can have only one leg of the trip using TLS, or both
  - You need to configure server settings for inbound connection
  - You need to configure client settings for outbound connection

Special Case – Automatic Client Re-route

- Automatic Client Re-route configuration allows only a single port when issuing the UPDATE ALTERNATE SERVER command
- Fine if you are only using one of TLS or TCP/IP
- But what if you need to use both?
  - For example during a period of migration
- You must use Client Affinities to configure alternate servers at the client
  - You can then control on a client by client basis whether to use TLS or TCP/IP

Configuring HADR – TLS between primary and standby

- TLS must be configured on both primary and standby
- Update these DBM CFG parameters
  - SSL_SRV_KEYDB – full path to server keystore file
  - SSL_SRV_STASH – full path to server stash file
- Update this DB CFG parameter
  - HADR_SSL_LABEL – label of certificate in keystore file
  - Certificates can be unique per server or the same
- Restart the primary and standby instances
- TLSV12 is enforced automatically

New!
Linux – 11.1.1.1
Others – 11.1.3.3
non-pureScale only

Configuring KMIP – Db2 Native Encryption

- KMIP requires clients to have certificates for authentication
- Setup KMIP configuration file with
  - SSL_KEYDB=full path to keystore file
  - SSL_KEYDB_STASH=full path to stash file for keystore
  - SSL_KMIP_CLIENT_CERTIFICATE_LABEL=label of client certificate in keystore

- TLSV12 is enforced automatically

Configuring Federation – DRDA Wrapper

- Add the following to the options of the CREATE SERVER command
  - ssl_keystore ‘<fullpath_to_keystore_file>’,
    ssl_keystash ’<fullpath_to_stash_file>’

- Or for simplified SSL setup, just specify the file containing the Root CA certificate in the options
  - ssl_servercertificate ‘<fullpath_to_ca_cert>’

Configuring LDAP plugins

- In IBMLDAPSecurity.ini (LDAP Security plugin configuration file)
  - ENABLE_SSL=TRUE
    SECURITY_PROTOCOL=TLSV12
    SSL_KEYFILE=full path to keystore file
    SSL_PW=password for keystore file
  - Using MSAD? See: https://www-01.ibm.com/support/docview.wss?uid=swg1IT21848
- Restart the instance
- STARTTLS is not supported
  - LDAP Server must be listening on dedicated TLS port (default 636)

Trusted Context and TLS

- You are able to force the use of TLS to establish a trusted connection
  - Specify WITH ENCRYPTION HIGH on the CREATE TRUSTED CONTEXT statement

Microsoft Certificate Store

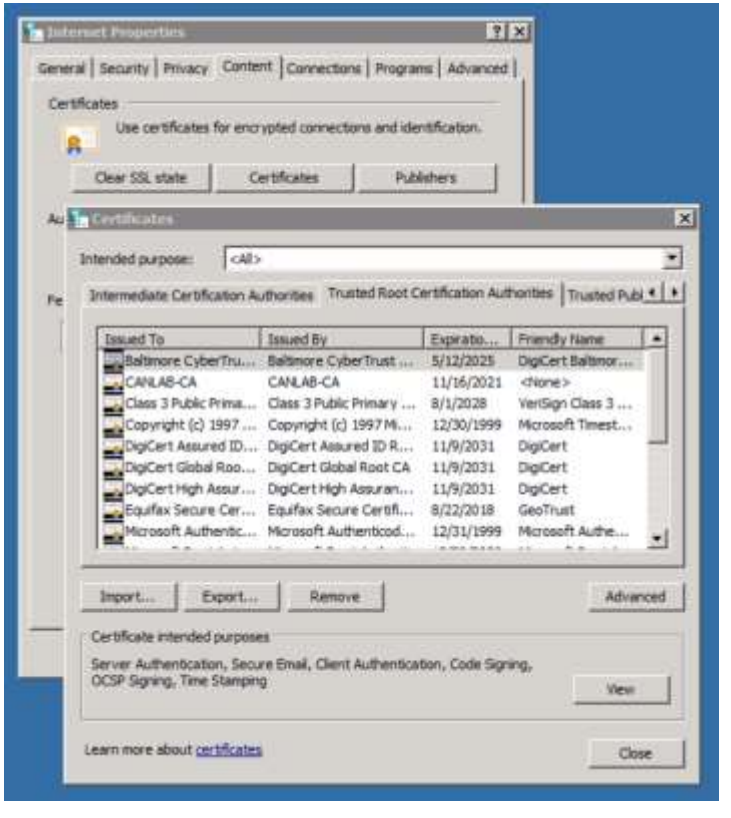
- GSKit and Java can integrate with the Microsoft Certificate Store that is part of Windows
  - For Root Certificate
  - End-point certificates

- If your enterprise is already using the MSCS, this can save you all the work of creating and managing your own keystore files

Microsoft Certificate Store

Find the store in the Control Panel

- Internet Options->Content->Certificate
- Personal tab for end point certificates, “Trusted Root” for Root CA certs
- You can use the GUI to import certificates, or GSKit command line
- The “Friendly Name” is the label
  - Watch out for duplicate friendly names, only the first found is used



Microsoft Certificate Store - GSKit

- Set these DBM CFG:
  - SSL_[CLNT|SVR]_KEYDB GSK_MS_CERTIFICATE_STORE
  - SSL_[CLN|SVR]_STASH NULL
- You can use GSKit to work with the store:
  - **`gsk8capicmd_64 -cert -list –db GSK_MS_CERTIFICATE_STORE`**
- GSKit only works with certificates for the current user account, not the computer account
- Data Server Client missing a file to work with MS Certificate Store
  - For CLI/CLP based applications - fixed in 10.5 FP9 and 11.1.3.3
    - https://www-01.ibm.com/support/docview.wss?uid=swg1IT21914
    - https://www-01.ibm.com/support/docview.wss?uid=swg1IT13953

Microsoft Certificate Store - Java

- Truststore – set one of these to WINDOWS-ROOT
  - db2.jcc.sslTrustStoreType
  - DB2BaseDataSource.sslTrustStoreType
  - javax.net.ssl.trustStoreType
- Keystore – set one of these to WINDOWS-MY
  - For certificate authentication only
  - db2.jcc.sslKeyStoreType
  - DB2BaseDataSource.sslKeyStoreType
  - javax.net.ssl.keyStoreType
  - keyUsage can be used to further refine what types of certificates are used

Special Case – Certificate authentication to Db2 for Z/OS

- Only supported for Db2 for Z/OS
- Certificates used instead of a password
  - In addition to importing root certificate, you will need to follow steps to obtain a certificate for your client
- AKA 2 way TLS or mutual authentication

- In db2cli.ini or db2dsdriver.cfg (not DBM CFG) specify:
  - SSLClientLabel=label of client's certificate in keystore
  - Authentication=CERTIFICATE

Special Case – Certificate authentication continued

- Java differentiates between truststore for root CA certificates and keystores containing endpoint certificates
- You will need to use keytool to create a .jks keystore
- Then set these values (in addition to setting truststore as before)

```
System.setProperty("javax.net.ssl.keyStore",”pathtokeystorefile");
System.setProperty("javax.net.ssl.keyStorePassword",”<pw>");
dataSource.setSecurityMechanism
(com.ibm.db2.jcc.DB2BaseDataSource.TLS_CLIENT_CERTIFICATE_SECURITY);
```

  - Could also set
    - db2.jcc.sslKeyStore
    - DB2BaseDataSource.sslKeyStore

Client certificates to Db2 for Z/OS – mutual authentication (not certificate authentication)

- As an alternative to certificate authentication discuss above, Db2 for Z/OS can request the client send it's certificate in addition to username and password
  - Client is configured with SERVER_ENCRYPT authentication, not CERTIFICATE
  - The use of SSLCLientLabel requires CERTIFICATE authentication
- Within a GSKit keystore, you can designate a certificate as the "default" one to send in such a scenario
  - The default option has been deprecated
  - Use at your own risk, it likely will not work in the future
  - Only in .kdb file, not .p12

Diagnostics with OpenSSL

- The openssl s_client command can be used to establish a TLS connection to a Db2 server
  - Can test for
    - What certificate is in use
    - What ciphers are available for use
- Exampe:
  - openssl s_client -connect <hostname>:<port>

How do I find out the details of a certificate file - openssl

- **openssl x509 -noout -text -in server.cer**

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            53:7c:d1:05:7a:03:f5:6e
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=CA, ST=ON, L=Markham, O=IBM, OU=DB2, CN=myserver
        Validity
            Not Before: Mar 30 19:14:15 2019 GMT
            Not After : Mar 30 19:14:15 2020 GMT
…
```

Migrating from TLS 1.0/1.1 to 1.2 or 1.3

- Moving to TLS 1.2
  - Keys MUST be 2048 bits or greater
  - Must use SHA-2 family hash algorithms (SHA 256, 384, 512)

- Preparing for the future TLS 1.3
  - Don’t use DSA signed certificates (Digital Signature Algorithm)
  - Instead use RSA or ECDSA (Elliptic Curve DSA)

Outstanding Issues

- Connection concentrator is not supported with TLS
- REGISTER DB2 SERVER IN LDAP & CATALOG LDAP NODE
  - Do not support Db2 server using TLS
  - Do not support communicating with LDAP over TLS
  - You will need to catalog the TCPIP node directly on each client
- STARTTLS is not supported
  - Ability to use a single port for both TCP/IP and TLS traffic
- DB2NODE support requires an APAR to fix
  - Inter-member communication is hardcoded to TCP/IP
  - Expected to fix in Db2 10.5 FP11, Db2 11.1.4.5 and Db2 11.5.1.1

Some of the well known attacks against TLS

- Heartbleed
  - Db2 not affected as GSKit does not implement heartbeat
- BEAST
  - Only affects TLS 1.0 – You should be using TLS 1.2
- CRIME
  - HTTPS specific – Db2 does not use compression with TLS
- POODLE
  - Fixed: http://www-01.ibm.com/support/docview.wss?uid=swg21692618

More Attacks

- LogJam
  - Fixed: http://www-01.ibm.com/support/docview.wss?uid=swg21967893
- Sweet32
  - Db2 removed use of 3DES from ciphers:
  - Fixed: http://www-01.ibm.com/support/docview.wss?uid=swg21994375
- RC4
  - Fixed: http://www-01.ibm.com/support/docview.wss?uid=swg21717865
- FREAK
  - Fixed: http://www-01.ibm.com/support/docview.wss?uid=swg21968869

Post Quantum Cryptography

- Public Key Cryptography will be the most impacted by Quantum Computers
  - The math it is based on will be easier to solve with quantum computers
  - Symmetric cryptography thought to be more resistant
- An area of active research
  - https://www.zurich.ibm.com/securityprivacy/quantumsafecryptography.html
- New algorithms required that are hard for quantum computers to break
- Something to keep an eye on in the future