



IBM Software Group – Enterprise Networking Software

Certificate Authentication in the z/OS Internet Key Exchange

August 5, 2010

Allen Bailey - eabailey@us.ibm.com

Lin Overby - overbylh@us.ibm.com

Chris Meyer – meyerchr@us.ibm.com

z/OS Communications Server Security

Trademarks, notices, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States or other countries or both:

- ▶ Advanced Peer-to-Peer Networking®
- ▶ AIX®
- ▶ alphaWorks®
- ▶ AnyNet®
- ▶ AS/400®
- ▶ BladeCenter®
- ▶ Candle®
- ▶ CICS®
- ▶ DB2 Connect
- ▶ DB2®
- ▶ DRDA®
- ▶ e-business on demand®
- ▶ e-business (logo)
- ▶ e business (logo)®
- ▶ ESCON®
- ▶ FICON®
- ▶ GDDM®
- ▶ HiperSockets
- ▶ HPR Channel Connectivity
- ▶ HyperSwap
- ▶ i5/OS (logo)
- ▶ i5/OS®
- ▶ IBM (logo)®
- ▶ IBM®
- ▶ IMS
- ▶ IP PrintWay
- ▶ IPDS
- ▶ iSeries
- ▶ LANDP®
- ▶ Language Environment®
- ▶ MQSeries®
- ▶ MVS
- ▶ NetView®
- ▶ OMEGAMON®
- ▶ Open Power
- ▶ OpenPower
- ▶ Operating System/2®
- ▶ Operating System/400®
- ▶ OS/2®
- ▶ OS/390®
- ▶ OS/400®
- ▶ Parallel Sysplex®
- ▶ PR/SM
- ▶ pSeries®
- ▶ RACF®
- ▶ Rational Suite®
- ▶ Rational®
- ▶ Redbooks
- ▶ Redbooks (logo)
- ▶ Sysplex Timer®
- ▶ System i5
- ▶ System p5
- ▶ System x
- ▶ System z
- ▶ System z9
- ▶ Tivoli (logo)®
- ▶ Tivoli®
- ▶ VTAM®
- ▶ WebSphere®
- ▶ xSeries®
- ▶ z9
- ▶ zSeries®
- ▶ z/Architecture
- ▶ z/OS®
- ▶ z/VM®
- ▶ z/VSE

- ▶ Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
- ▶ Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
- ▶ Intel, Intel Inside (logos), MMX and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.
- ▶ UNIX is a registered trademark of The Open Group in the United States and other countries.
- ▶ Linux is a trademark of Linus Torvalds in the United States, other countries, or both.
- ▶ Red Hat is a trademark of Red Hat, Inc.
- ▶ SUSE® LINUX Professional 9.2 from Novell®
- ▶ Other company, product, or service names may be trademarks or service marks of others.
- ▶ This information is for planning purposes only. The information herein is subject to change before the products described become generally available.
- ▶ Disclaimer: All statements regarding IBM future direction or intent, including current product plans, are subject to change or withdrawal without notice and represent goals and objectives only. All information is provided for informational purposes only, on an “as is” basis, without warranty of any kind.

All performance data contained in this publication was obtained in the specific operating environment and under the conditions described and is presented as an illustration. Performance obtained in other operating environments may vary and customers should conduct their own testing.

Refer to www.ibm.com/legal/us for further legal information.

Agenda

- Background
 - Cryptography basics
 - Digital certificates
 - IP Security (IPSec) / Internet Key Exchange (IKE)
- Certificate use in the IKE
 - Authentication
 - Negotiation modes
- Network Security Services Daemon (NSSD)
 - Overview
 - Digital signature operations
 - Advanced certificate services
 - Certificate revocation lists
 - Certificate trust chains
 - IKEv2 certificate encoding types
- Implementing the NSSD solution

z/OS[®] V1R12 Communications Server

Background

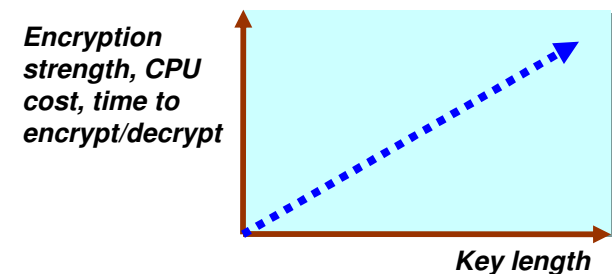


Agenda

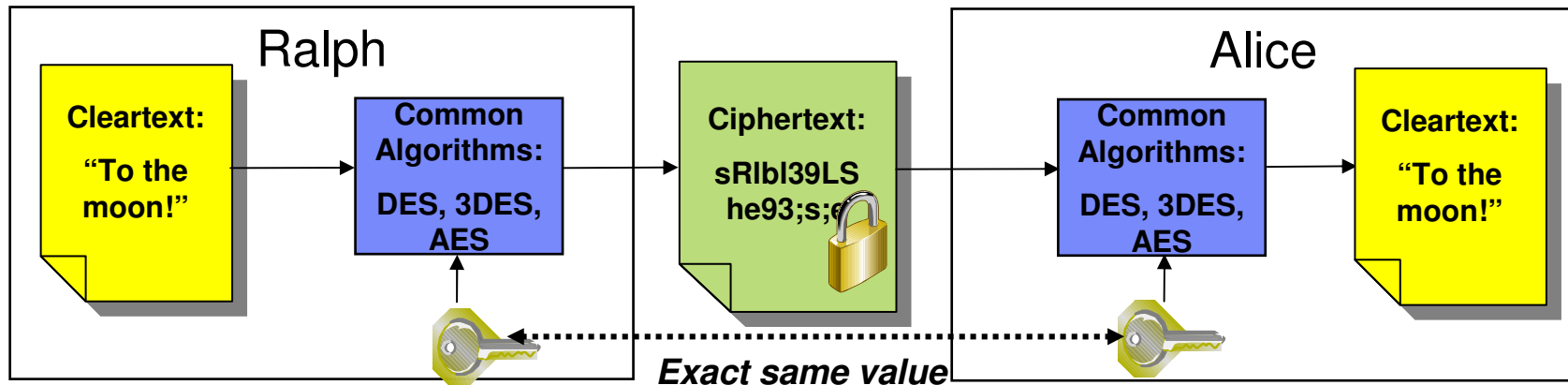
- Background
 - **Cryptography basics**
 - Digital certificates
 - IP Security (IPSec) / Internet Key Exchange (IKE)
- Certificate use in the IKE
 - Authentication
 - Negotiation modes
- Network Security Services Daemon (NSSD)
 - Overview
 - Digital signature operations
 - Advanced certificate services
 - Certificate revocation lists
 - Certificate trust chains
 - IKEv2 certificate encoding types
- Implementing the NSSD solution

Cryptographic basics

- Cryptography is the use of mathematical algorithms to transform data for the purposes of ensuring:
 - **Partner authentication** – proving the other end point of the secure communication is who it claims to be (certificates and asymmetric encryption)
 - **Data privacy** – hiding the data (encryption/decryption)
 - **Data integrity** – proving the data hasn't been modified since it was sent (message digests and secure message authentication codes)
 - **Data origin authentication** – proving the data's origin (message digests and secure message authentication codes)
- Cryptographic operations are compute intensive, hence the need for hardware assist technologies
- General rule: *For a given algorithm*, the longer keys, the stronger security, the more compute intensive
 - For example, AES-128 vs. AES-256
 - Increases the amount of work an attacker needs to do to crack the code

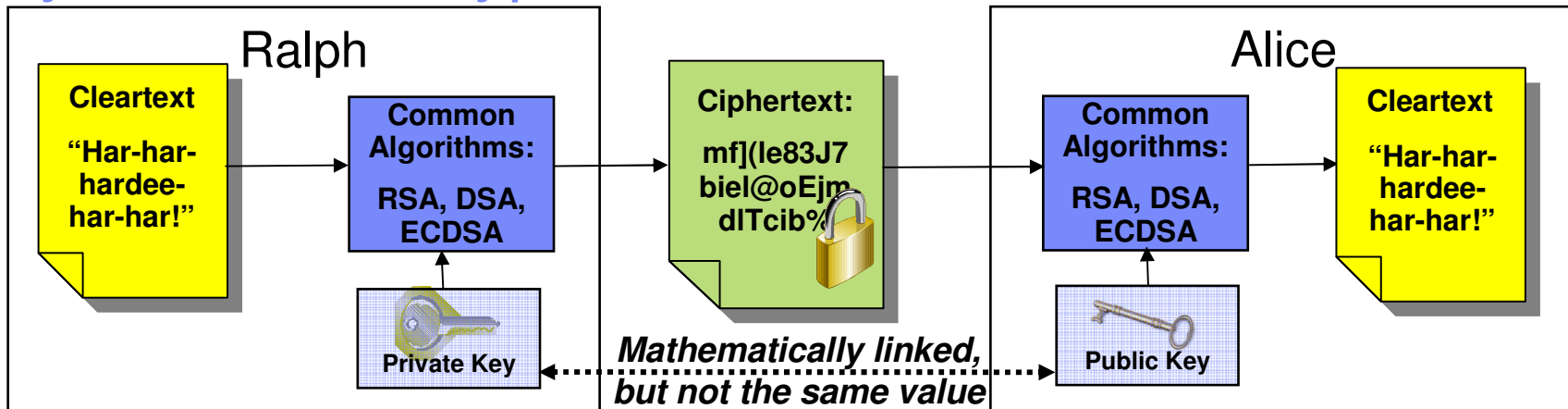


Symmetric encryption



- **Only one key value - “shared secret” between both parties**
 - Used for both encryption and decryption
 - Hence, the symmetry – each side has the same key
- **Much faster than asymmetric cryptography**
 - You typically use symmetric encryption for bulk encryption/decryption
- **Also known as...**
 - “secret key encryption”
 - “private key encryption” (easily confused with asymmetric)
- **Securely sharing and exchanging the key between both parties is a major issue**

Asymmetric encryption

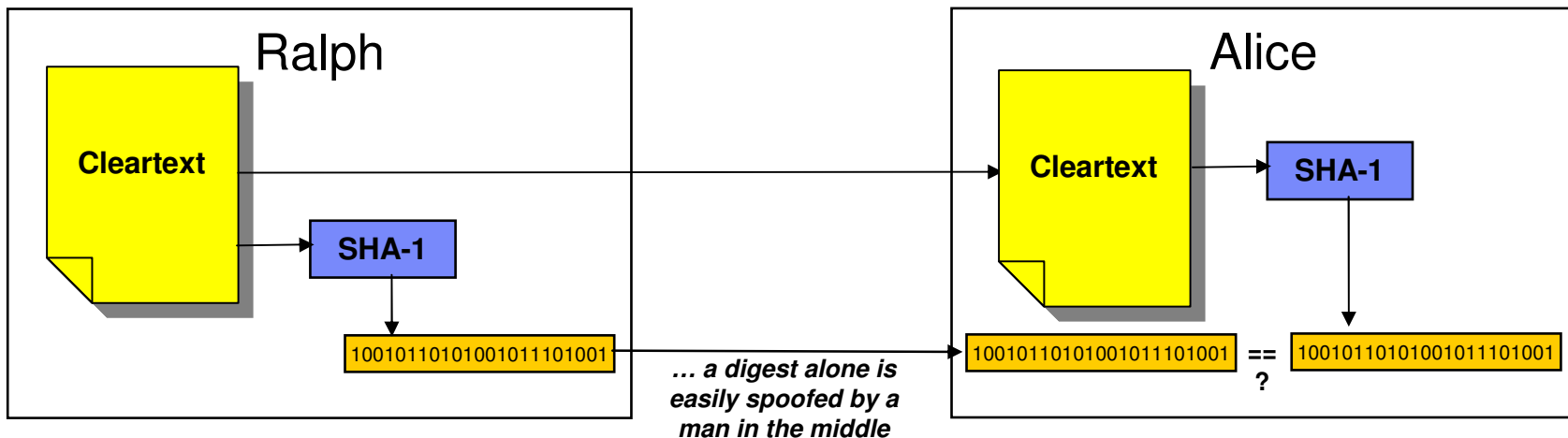
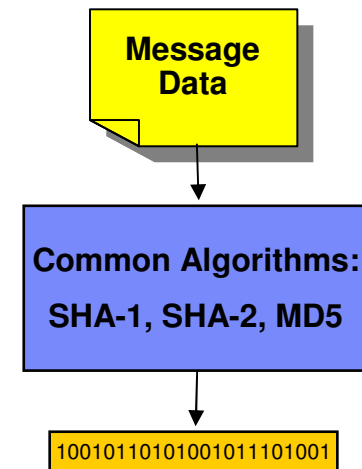


- **Two different key values – no shared secrets!**
 - Private key is known only to owner
 - Public key is freely distributed to others
 - Data encrypted with private key can only be decrypted with public key and vice versa
 - No way to derive one key value from the other
- **Great for authentication and non-repudiation**
 - “digital signatures” – more on this later
- **Very expensive computationally**
 - Not so great for bulk encryption
 - Typically used to encrypt small data objects like message digests or symmetric keys
- **Also known as “public key cryptography”**

Message digests

A message digest is...

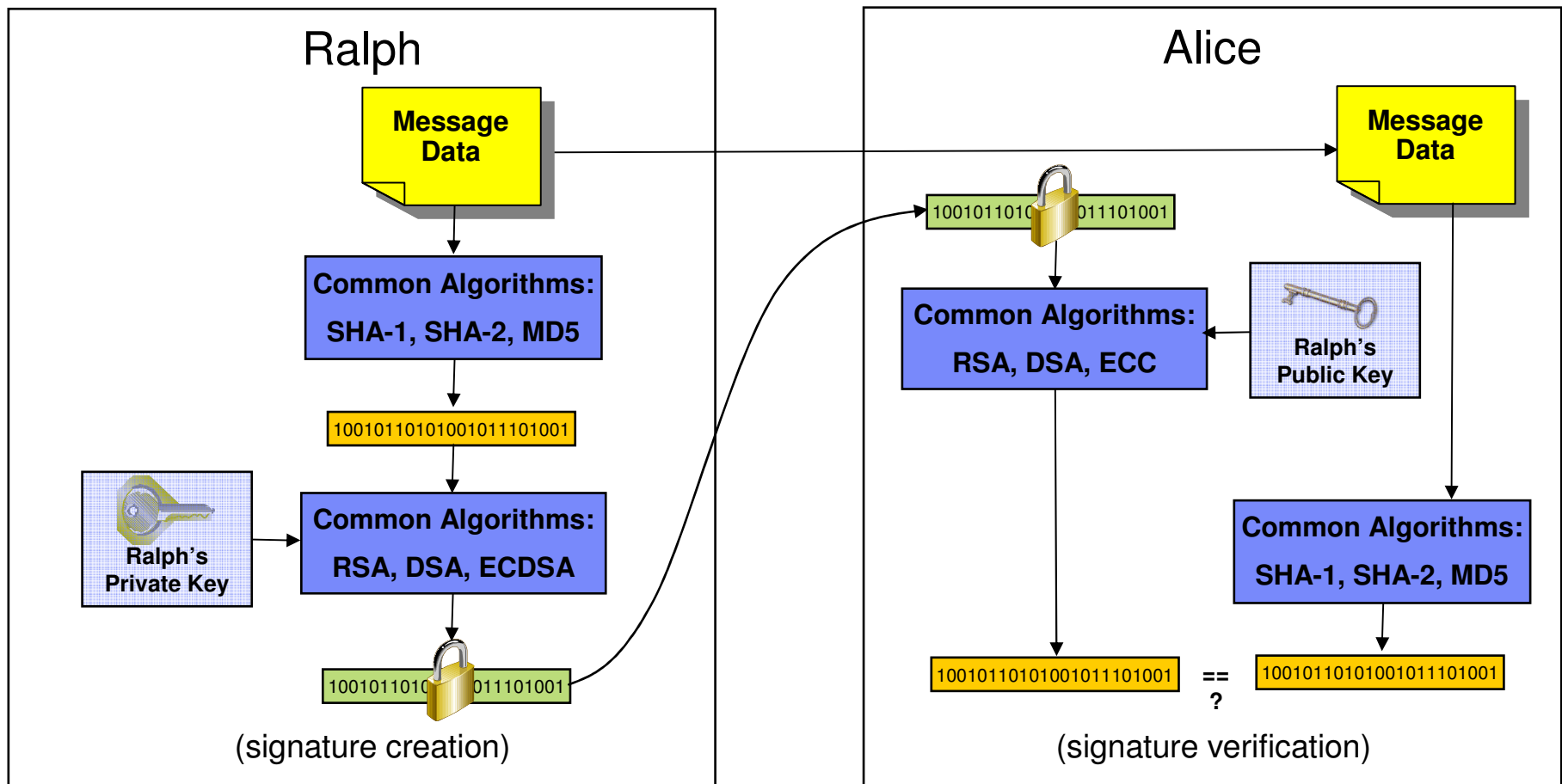
- **NOT based on a secret key**
- **a fixed-length value generated from variable-length data**
- **unique:**
 - the same input data always generates the same digest value
 - small change in data generates a very different hash value
 - extremely difficult (and time consuming) to find two different data values that result in the same hash value
- **one-way: can't reverse a digest value back to the original data**
- **hence, also known as a “one-way hash”**
- **an element of proving data integrity and origin authentication (but not enough on its own...)**



Digital signatures

A digital signature is...

- a message digest that is encrypted with the sender's private key (hence, based on an asymmetric algorithm)
- very good for proving data integrity and authenticating data origin



Agenda

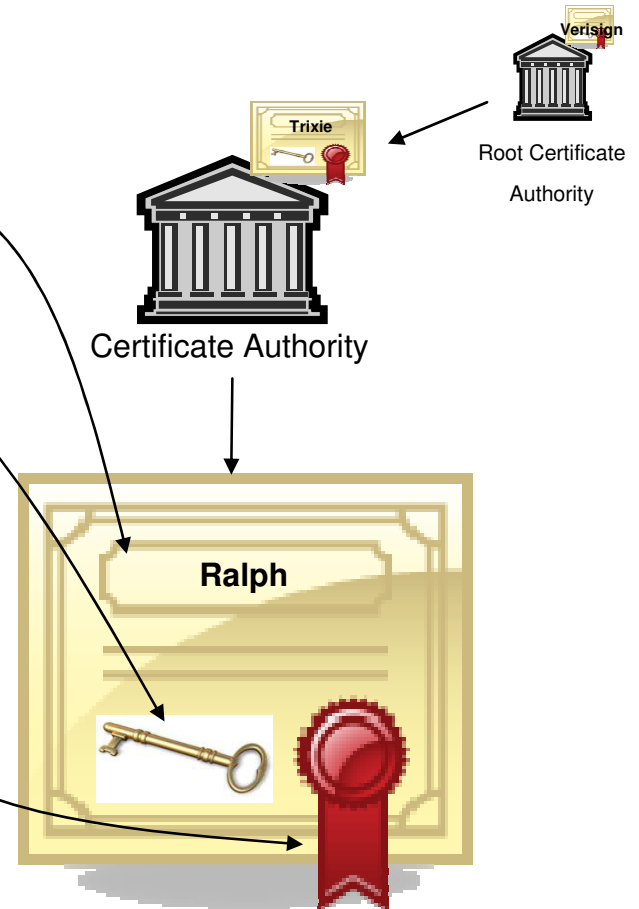
- Background
 - Cryptography basics
 - **Digital certificates**
 - IP Security (IPSec) / Internet Key Exchange (IKE)
- Certificate use in the IKE
 - Authentication
 - Negotiation modes
- Network Security Services Daemon (NSSD)
 - Overview
 - Digital signature operations
 - Advanced certificate services
 - Certificate revocation
 - Certificate trust chains
 - IKEv2 certificate encoding types
- Implementing the NSSD solution

Digital certificates - Purpose

- A digital document...
- ...that binds a *subject* to a public/private key pair
- ...contains the public key to allow for convenient distribution of that key
- ...is *signed* by a trusted third party called a *Certificate Authority (CA)* to prove its authenticity
- ...can be *self-signed*, but such certificates are typically used for testing or very small-scale applications since the trust element is lost

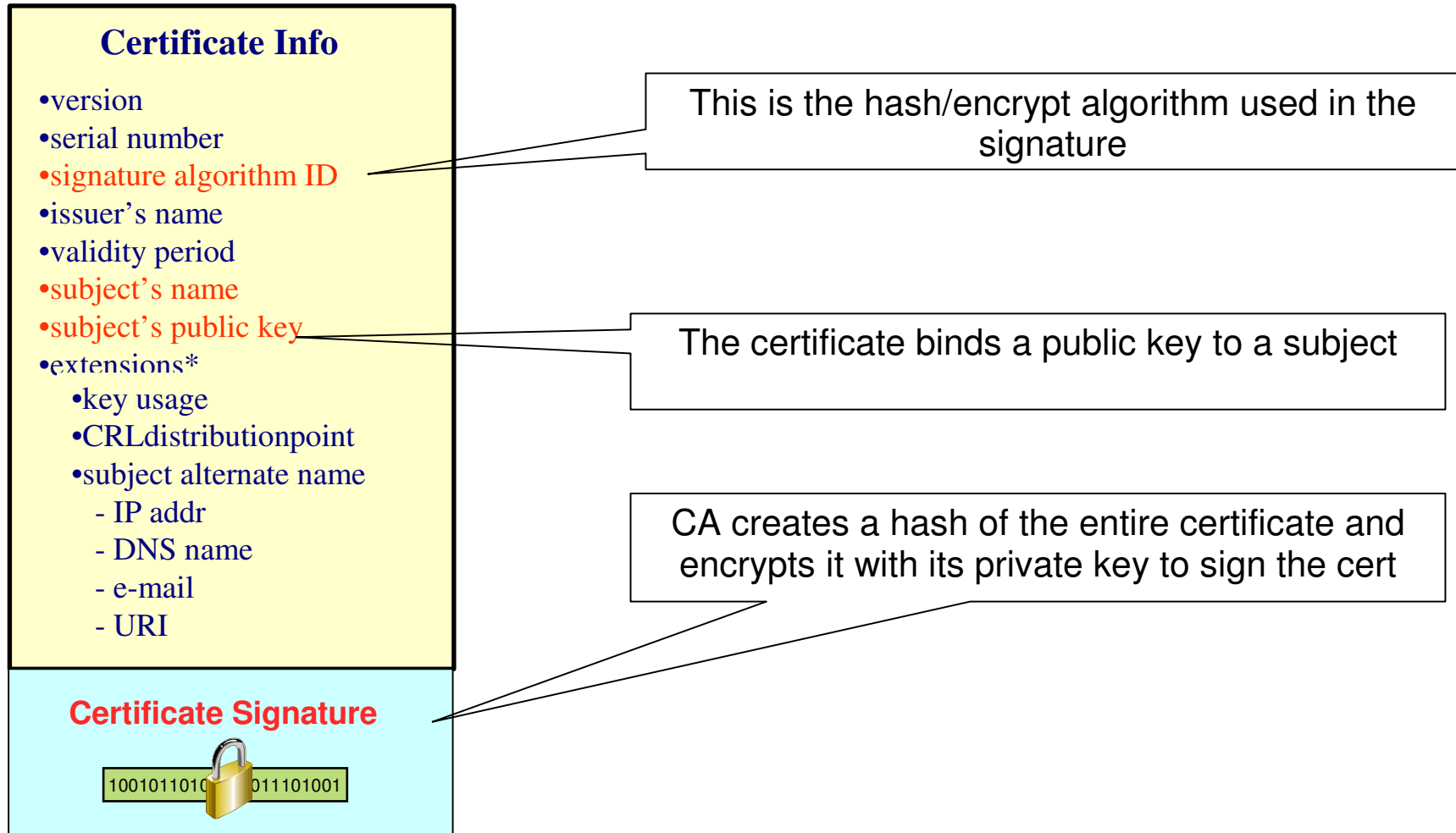
Certificate Authorities...

- ...have their own digital certificate and public/private key pair. This *CA certificate* is used to verify the certificates that were issued by the given CA
- ...issue certificates for subjects and sign those certificates with their own private key
- ...can be arranged in a hierarchy (the top of the hierarchy is the “root CA”).
- ...are part of a “Public Key Infrastructure” (PKI)



X.509 is the most widely-adopted standard

Digital certificates - Structure

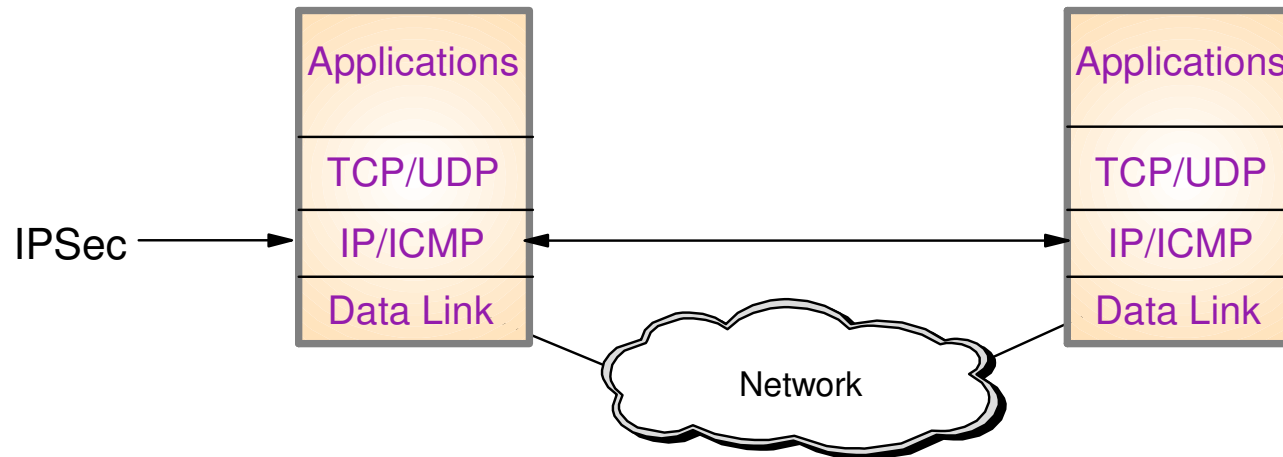


*There are more extensions defined in X.509 standard than those listed here.

Agenda

- Background
 - Cryptography basics
 - Digital certificates
 - **IP Security (IPSec) / Internet Key Exchange (IKE)**
- Certificate use in the IKE
 - Authentication
 - Negotiation modes
- Network Security Services Daemon (NSSD)
 - Overview
 - Digital signature operations
 - Advanced certificate services
 - Certificate revocation lists
 - Certificate trust chains
 - IKEv2 certificate encoding types
- Implementing the NSSD solution

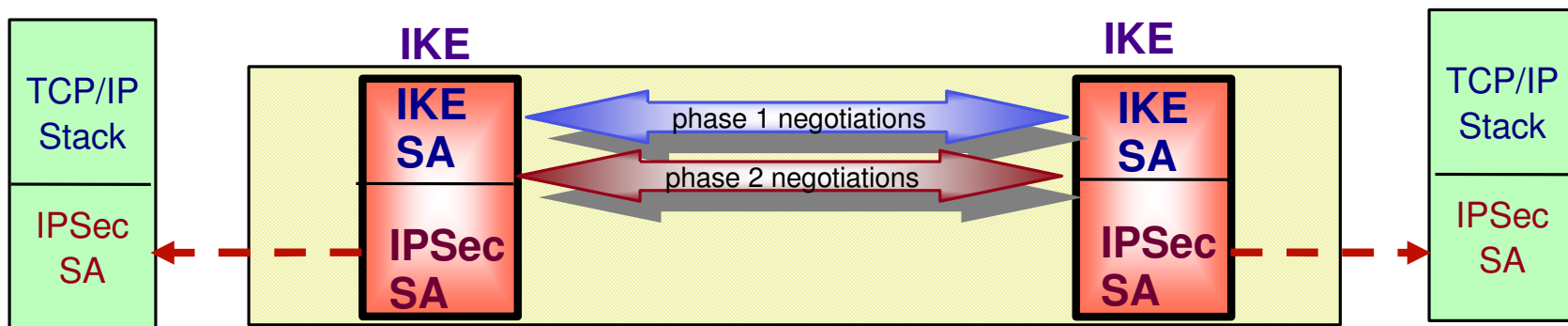
IPSec overview



- **Provides authentication, integrity, and data privacy via IPSec security protocols**
 - Authentication Header (AH) - provides authentication / integrity
 - Encapsulating Security Protocol (ESP) - provides data privacy with optional authentication / integrity
- **Implemented at the IP layer**
 - Requires no application change
 - Secures traffic between any two IP resources - Security Associations (SA)
- **Management of crypto keys and security associations can be:**
 - Manual
 - Automated via key management protocol (IKE)

IPSec/IKE – Two phases of IKE

- Phase 1
 - Negotiates an IKE SA
 - Generates cryptographic keys that will be used to protect
 - Phase 2 negotiations
 - Informational exchanges
 - **Authenticates the identity of the parties involved**
- Phase 2
 - Negotiates an IPSec SA (a.k.a. 'CHILD SA') with a remote security endpoint
 - Generates cryptographic keys that are used to protect data
 - Performed under the protection of an IKE SA



z/OS® V1R12 Communications Server

Certificate use in the IKE



Agenda

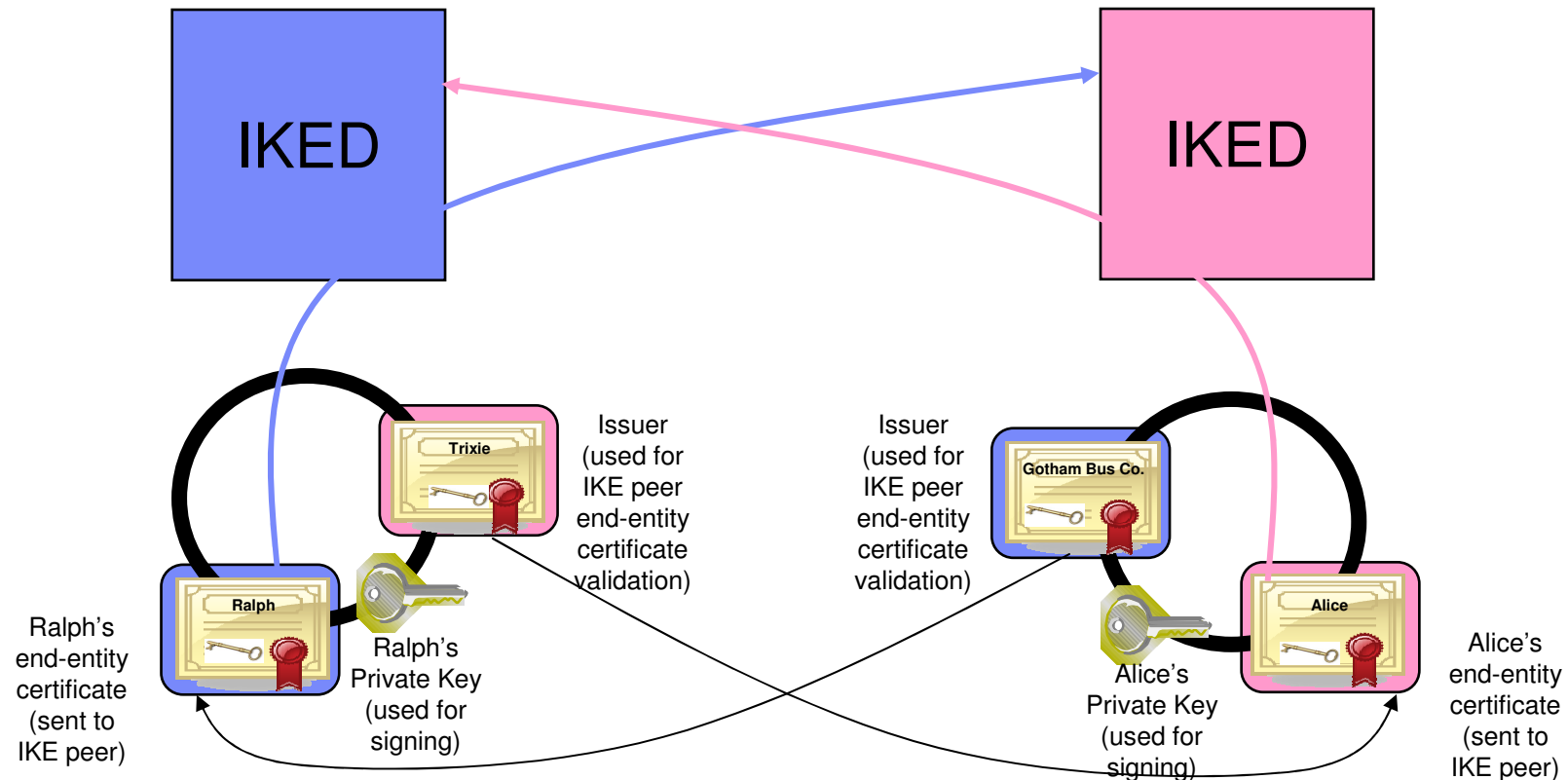
- Background
 - Cryptography basics
 - Digital certificates
 - IP Security (IPSec) / Internet Key Exchange (IKE)
- Certificate use in the IKE
 - **Authentication**
 - Negotiation modes
- Network Security Services Daemon (NSSD)
 - Overview
 - Digital signature operations
 - Advanced certificate services
 - Certificate revocation lists
 - Certificate trust chains
 - IKEv2 certificate encoding types
- Implementing the NSSD solution

IKE Authentication

- **Authentication is a crucial function of the IKE**
- **Authentication happens during Phase 1 IKE SA exchanges**
- **Some messages are used to carry authentication information**
- **Authentication methods**
 - Pre-shared key (not very scalable)
 - Digital signature using x.509 certificates (very scalable)
 - RSA
 - ECDSA (z/OS V1R12) - requires NSSD
- **IKE processing involves creation and verification of digital signatures**
- **Digital signature mode requires access to certificates and keys stored in a SAF repository (keyring)**

IKE peer-to-peer certificate relationships

- **Each host needs only its own end-entity certificate and the certificate of the trusted Certificate Authority that signed the IKE peer's end-entity certificate**
 - *(Requirements for the CA of the peer certificate can differ with V1R12 Certificate Trust Chain support)*



Agenda

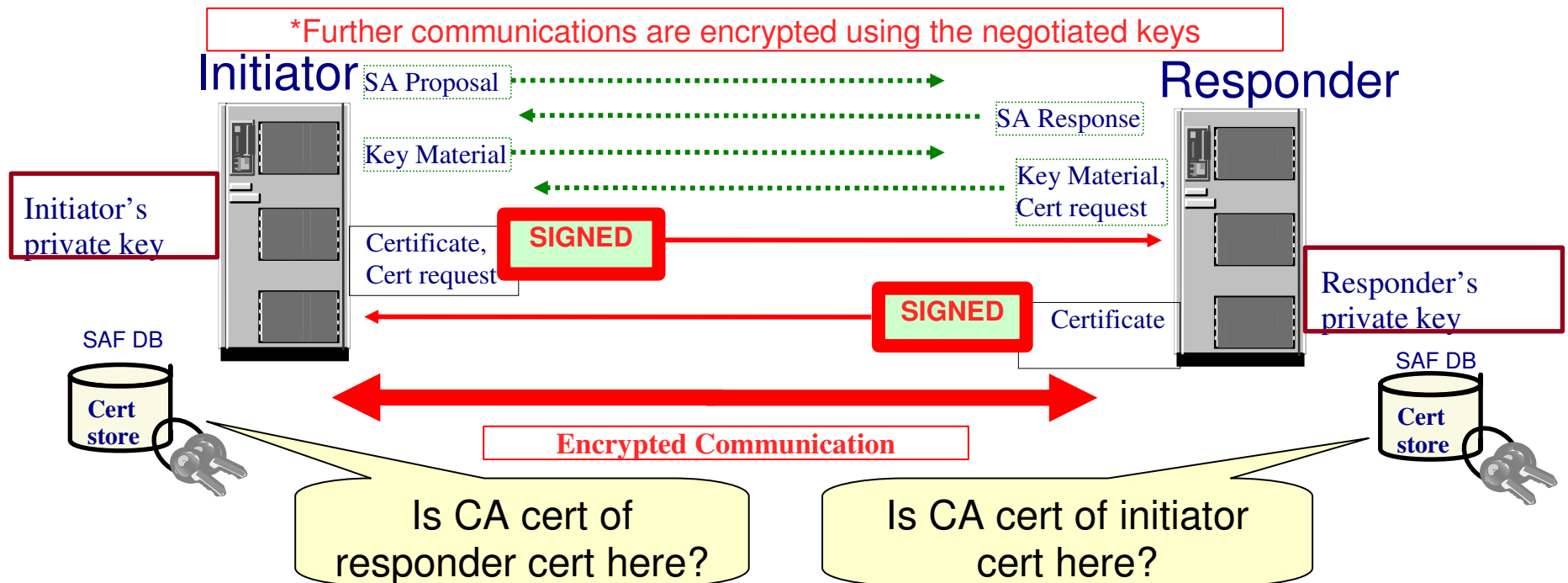
- Background
 - Cryptography basics
 - Digital certificates
 - IP Security (IPSec) / Internet Key Exchange (IKE)
- Certificate use in the IKE
 - Authentication
 - **Negotiation modes**
- Network Security Services Daemon (NSSD)
 - Overview
 - Digital signature operations
 - Advanced certificate services
 - Certificate revocation lists
 - Certificate trust chains
 - IKEv2 certificate encoding types
- Implementing the NSSD solution

Negotiation Modes of IKE Phase 1

- IKEv1 Main
 - uses six messages
 - a.k.a. identity-protected mode (peer identities are encrypted)
- IKEv1 Aggressive
 - uses three messages
 - peer identities are **not** encrypted
 - faster than main mode but potentially less secure
- IKEv2 (V1R12)
 - uses four messages
 - peer identities **are** encrypted
 - better performance than main mode
 - better protection than aggressive mode

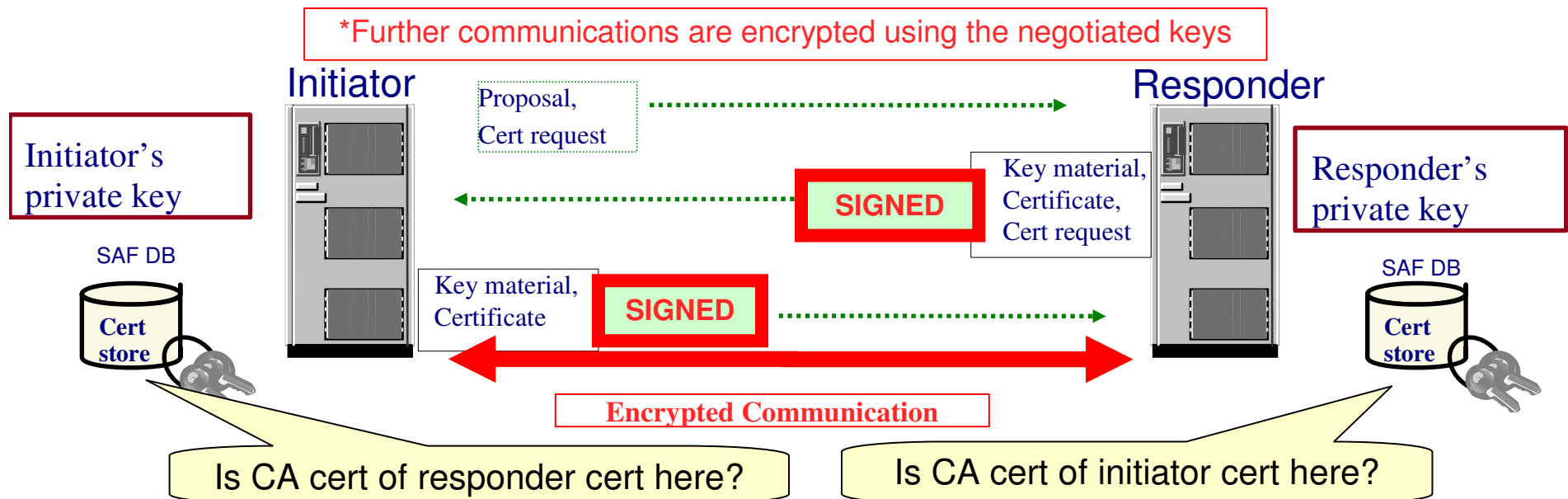
Certificate use in IKEv1 Main Mode

1. Initiator sends a proposal to peer
2. Responder replies with selected proposal to peer
3. Initiator sends keying material to peer
4. Responder replies with keying material and certificate request to peer
5. **CreateSignature** - Initiator creates message and uses private key to sign
6. Initiator sends signed message, certificate and certificate request to peer
7. **VerifySignature** - Responder verifies the integrity of the message and the authenticity of peer
8. **CreateSignature** - Responder creates message and uses private key to sign
9. Responder sends signed message, certificate and certificate request to peer
10. **VerifySignature** - Initiator verifies the integrity of the message and the authenticity of peer



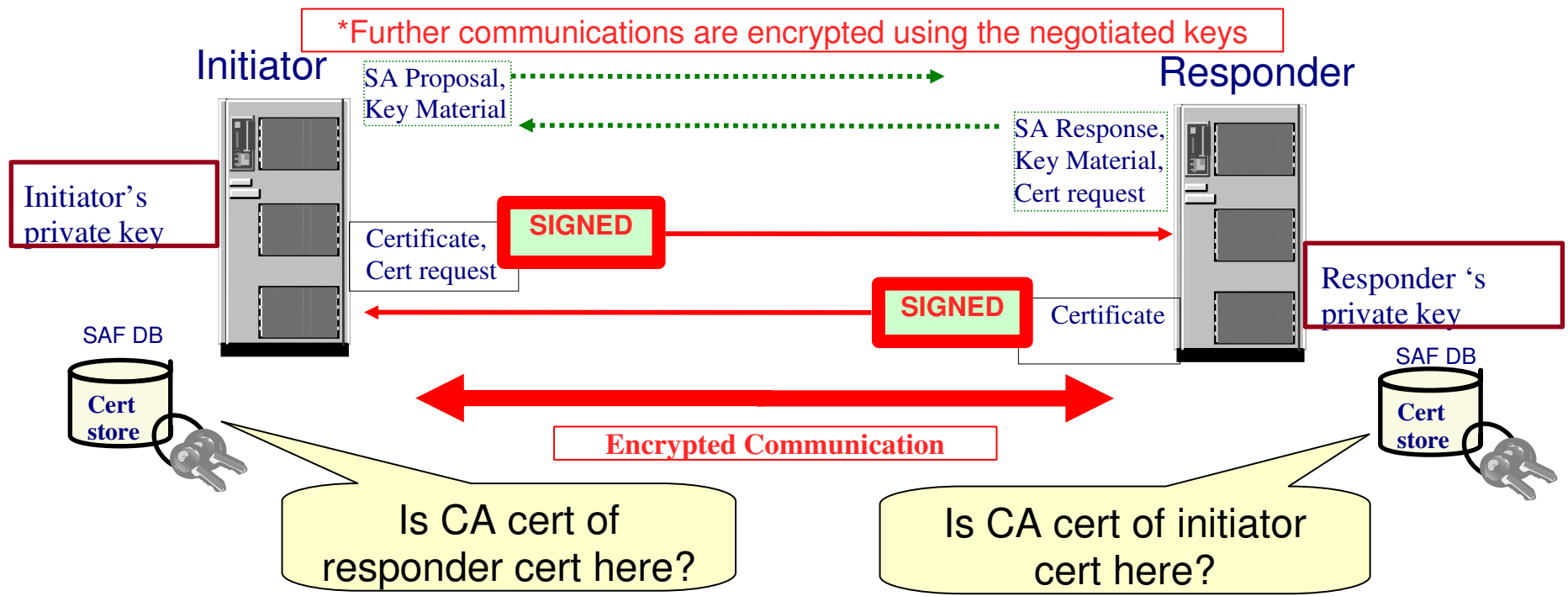
Certificate use in IKEv1 Aggressive Mode

1. Initiator sends a proposal along with a certificate request to peer
2. **CreateSignature** - Responder creates message and uses private key to sign
3. Responder sends signed message including certificate and certificate request to peer
4. **VerifySignature** - Initiator verifies the integrity of the message and the authenticity of peer
5. **CreateSignature** - Initiator creates message and uses private key to sign
6. Initiator sends signed message including certificate to peer
7. **VerifySignature** - Responder verifies the integrity of the message and the authenticity of peer



Certificate use in IKEv2 Mode

1. Initiator sends a proposal and keying material to peer
2. Responder replies with selected proposal, keying material and certificate request to peer
3. **CreateSignature** - Initiator creates message and uses private key to sign
4. Initiator sends signed message, certificate and certificate request to peer
5. **VerifySignature** - Responder verifies the integrity of the message and the authenticity of peer
6. **CreateSignature** - Responder creates message and uses private key to sign
7. Responder sends signed message, certificate and certificate request to peer
8. **VerifySignature** - Initiator verifies the integrity of the message and the authenticity of peer



z/OS® V1R12 Communications Server

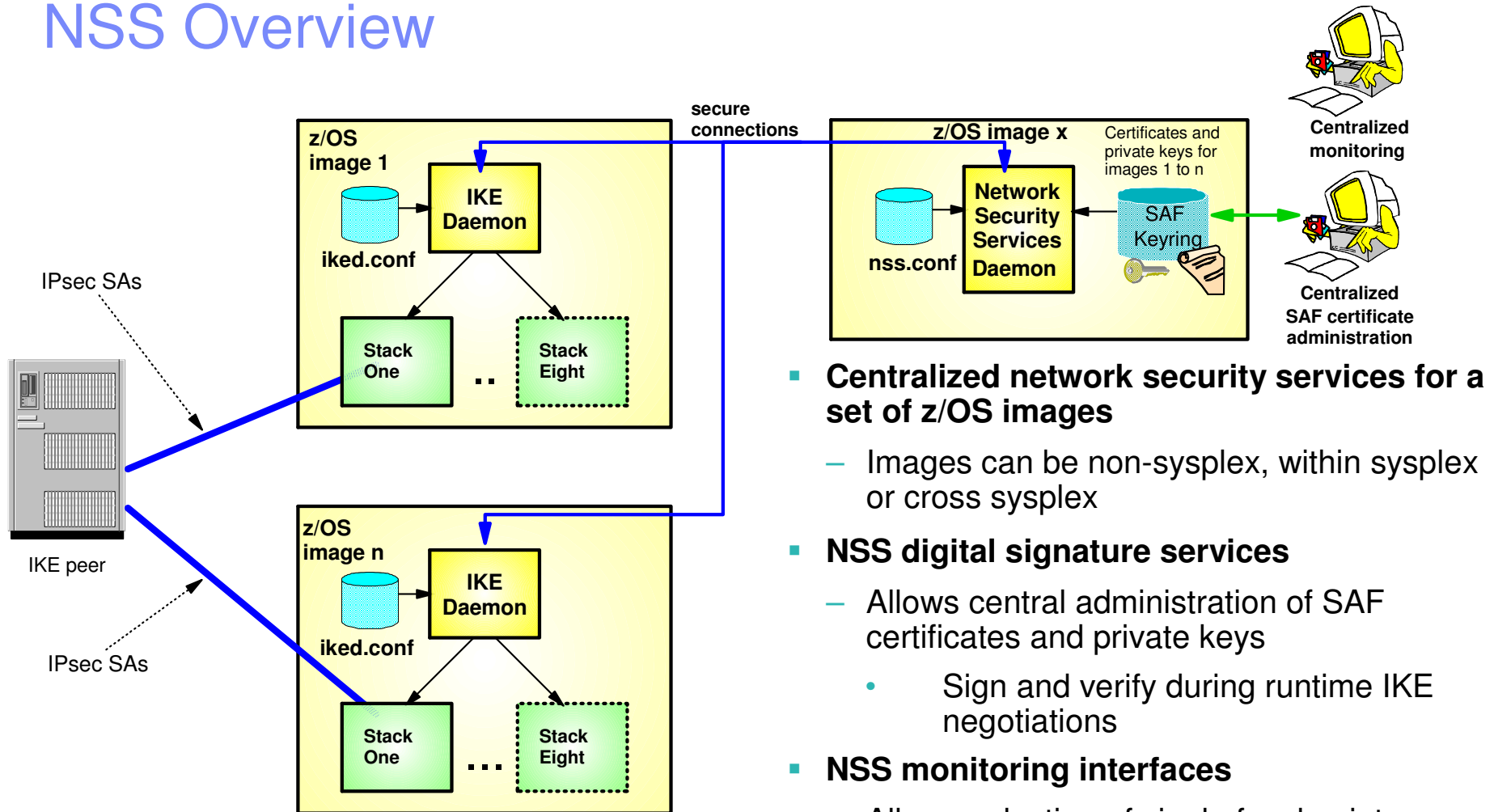
Network Security Services



Agenda

- Background
 - Cryptography basics
 - Digital certificates
 - IP Security (IPSec) / Internet Key Exchange (IKE)
- Certificate use in the IKE
 - Authentication
 - Negotiation modes
- Network Security Services Daemon (NSSD)
 - **Overview**
 - Digital signature operations
 - Advanced certificate services
 - Certificate revocation lists
 - Certificate trust chains
 - IKEv2 certificate encoding types
- Implementing the NSSD solution

NSS Overview



NSS role extended in z/OS V1R12

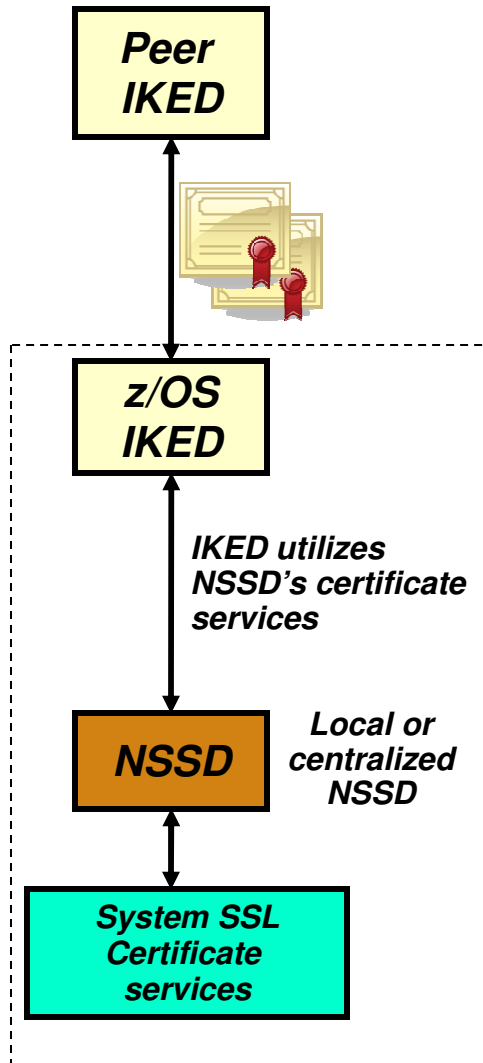
- NSS is required for z/OS V1R12 advanced certificate support
 - ▶ Certificate Revocation List
 - ▶ Certificate Trust Chain
- NSS is required for ALL IKEv2 certificate services

- **Centralized network security services for a set of z/OS images**
 - Images can be non-sysplex, within sysplex or cross sysplex
- **NSS digital signature services**
 - Allows central administration of SAF certificates and private keys
 - Sign and verify during runtime IKE negotiations
- **NSS monitoring interfaces**
 - Allows selection of single focal point as IPsec management hub
 - ipsec command for administrator
 - Network Monitor Interface (NMI) for management application

Agenda

- Background
 - Cryptography basics
 - Digital certificates
 - IP Security (IPSec) / Internet Key Exchange (IKE)
- Certificate use in the IKE
 - Authentication
 - Negotiation modes
- Network Security Services Daemon (NSSD)
 - Overview
 - **Digital signature operations**
 - Advanced certificate services
 - Certificate revocation lists
 - Certificate trust chains
 - IKEv2 certificate encoding types
- Implementing the NSSD solution

IKE Digital Signature operations with NSSD



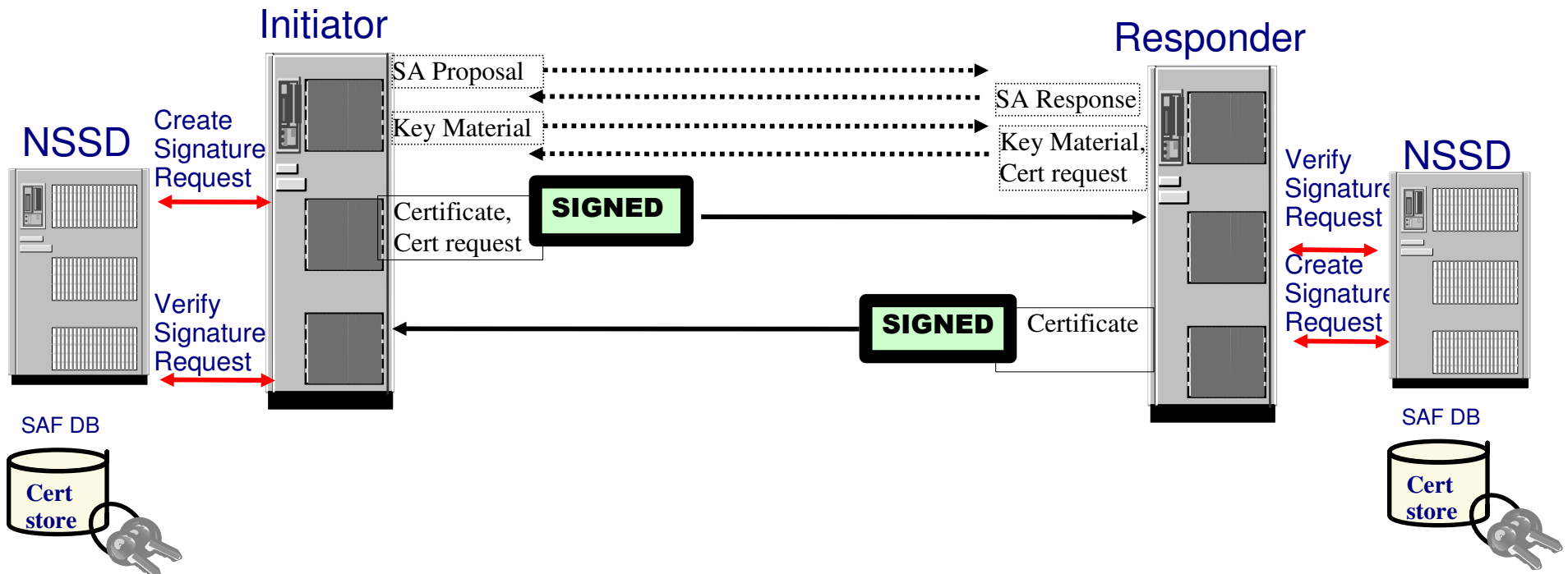
- **The IKE protocol for digital signature authentication requires the peers to exchange digital signatures**
 - The certificates used to verify the signatures are exchanged between IKE peers in certificate payloads in the IKE messages.

- **z/OS must perform digital signature creation and verification**
 - Can be performed locally by IKED for RSA signatures
 - IKED can request signature creation and verification from NSS daemon (NSSD)

- **Network Security Services (NSS) IPsec discipline certificate services**
 - **Create** a digital signature using a private key associated with the local IKED certificate
 - **Verify** a digital signature using the public key from the remote IKED certificate

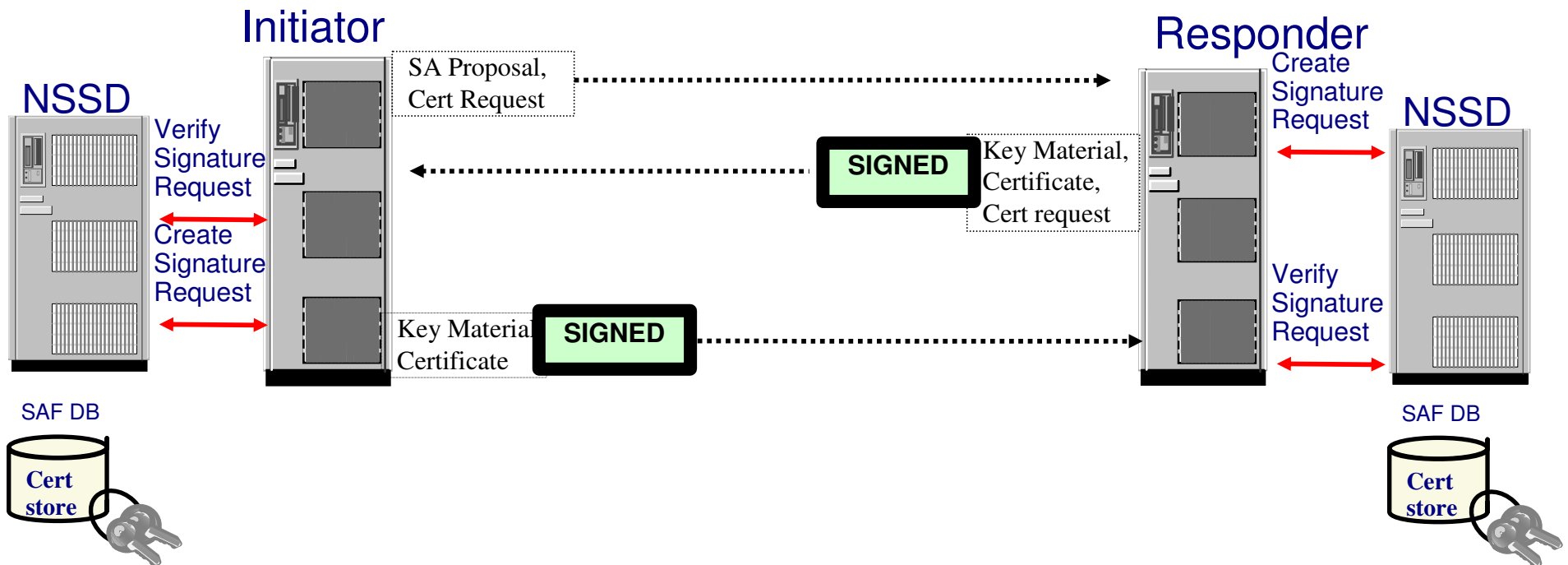
NSSD role in IKEv1 Main Mode Negotiation

1. Initiator sends a proposal to peer
2. Responder replies with selected proposal to peer
3. Initiator sends keying material to peer
4. Responder replies with keying material and certificate request to peer
5. **CreateSignatureRequest** - Initiator sends request to NSSD and awaits response
6. Initiator sends signed message, certificate and certificate request to peer
7. **VerifySignatureRequest** - Responder sends request to NSSD and awaits response
8. **CreateSignatureRequest** - Responder sends request to NSSD and awaits response
9. Responder sends signed message, certificate and certificate request to peer
10. **VerifySignatureRequest** – Initiator sends request to NSSD and awaits response



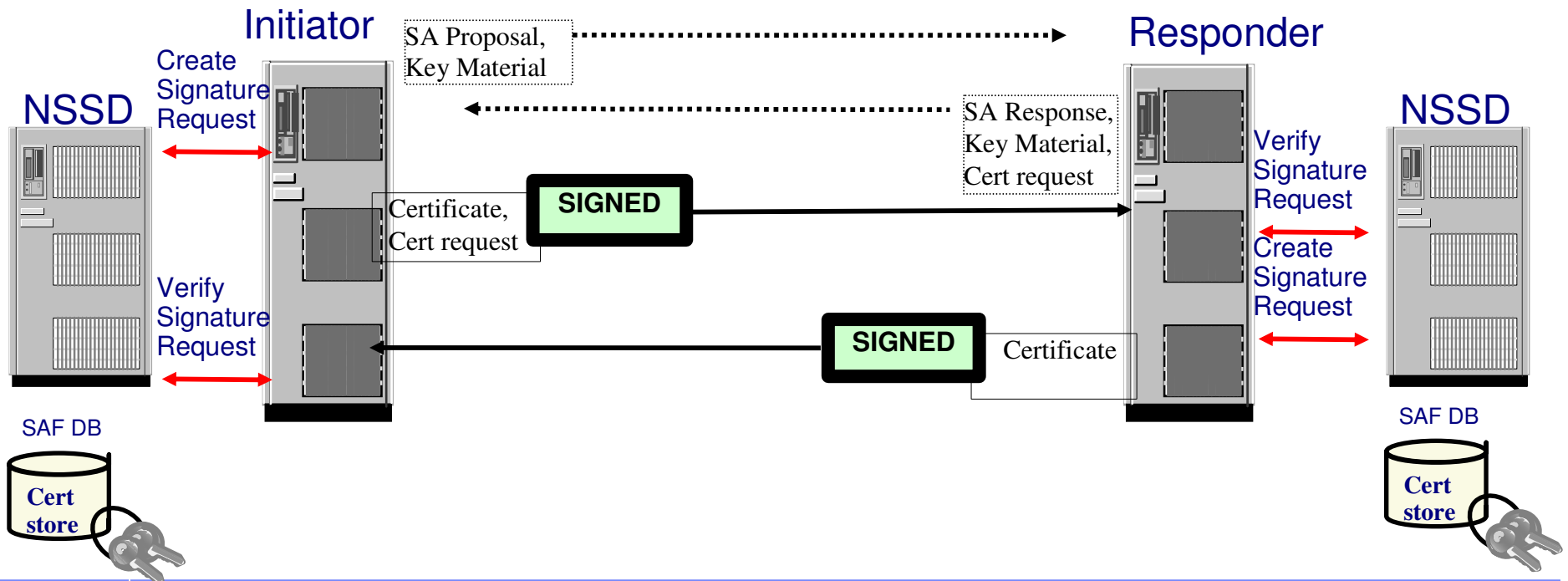
NSSD role in IKEv1 Aggressive Mode Negotiations

1. Initiator sends a proposal along with a certificate request to peer
2. **CreateSignatureRequest** - Responder sends request to NSSD and awaits response
3. Responder sends signed message including certificate and certificate request to peer
4. **VerifySignatureRequest** - Initiator sends request to NSSD and awaits response
5. **CreateSignatureRequest** - Initiator sends request to NSSD and awaits response
6. Initiator sends signed message including certificate to peer
7. **VerifySignatureRequest** - Responder sends request to NSSD and awaits response



NSSD role in IKEv2 Mode Negotiations

1. Initiator sends a proposal and keying material to peer
2. Responder replies with selected proposal, keying material and certificate request to peer
3. **CreateSignatureRequest** - Initiator sends request to NSSD and awaits response
4. Initiator sends signed message, certificate and certificate request to peer
5. **VerifySignatureRequest** - Responder sends request to NSSD and awaits response
6. **CreateSignatureRequest** - Responder sends request to NSSD and awaits response
7. Responder sends signed message, certificate and certificate request to peer
8. **VerifySignatureRequest** - Initiator sends request to NSSD and awaits response



Agenda

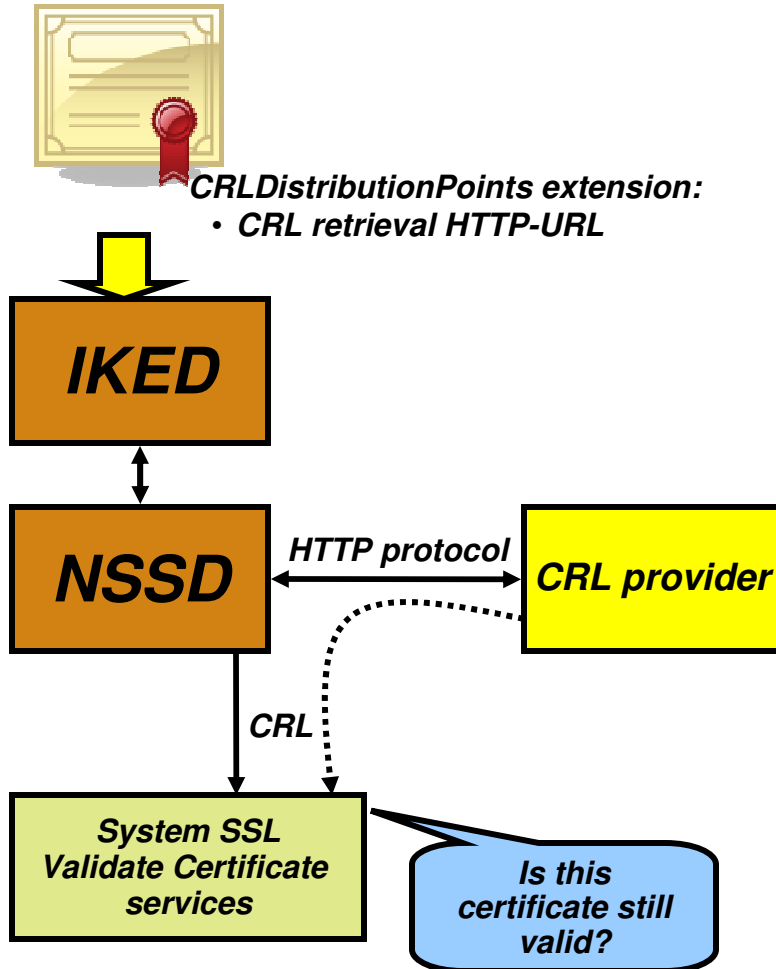
- Background
 - Cryptography basics
 - Digital certificates
 - IP Security (IPSec) / Internet Key Exchange (IKE)
- Certificate use in the IKE
 - Authentication
 - Negotiation modes
- Network Security Services Daemon (NSSD)
 - Overview
 - Digital signature operations
 - Advanced certificate services
 - **Certificate revocation lists**
 - Certificate trust chains
 - IKEv2 certificate encoding types
- Implementing the NSSD solution

Certificate revocation

- **A Certificate Revocation List is a list of certificates that have been revoked or are no longer valid.**
 - CRLs are digitally signed by issuing certificate authority
- **Common reasons a certificate might be revoked:**
 - Private key has been compromised
 - Termination of an affiliation
 - Employment
 - Membership
 - Certificate no longer valid for stated purpose
- **Certificate revocations should be taken into account when checking the validity of a certificate**



z/OS Communications Server CRL support



- When IPsec authenticates a digital signature, it needs to ensure that the certificate associated with the signing private key is still valid
- NSSD will retrieve CRLs using information in the CRLDistributionPoints extension in a certificate
 - HTTP-URLs only
 - Retrieval of CRLs from LDAP servers not supported
- NSSD will pass CRLs to z/OS System SSL services
- System SSL will validate the certificate against the CRL to ensure the certificate has not been revoked
- **IKED depends on NSSD for this function**

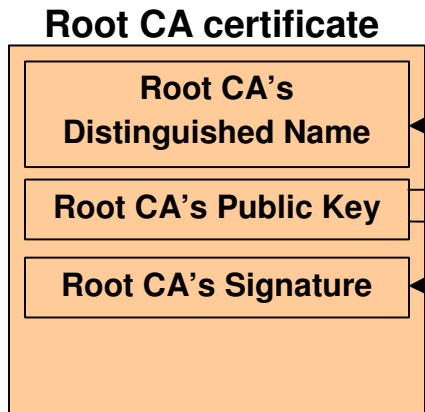
CRL processing controls

- **IKED controls the level of CRL checking enforced by NSSD**
 - IKED retrieves the revocation checking level to be performed from the RevocationChecking parameter on the **KeyExchangePolicy** or **KeyExchangeAction** IPsec policy statement
 - The revocation checking level is passed to NSS on the signature validation request
- **NSSD supports three levels of CRL checking**
 - None – Do no check revocation information
 - Loose – Check revocation information if it can be obtained; otherwise, assume valid.
 - Strict – Always check revocation information. If it cannot be obtained assume not valid
- **Retrieved CRLs are cached based on the NSSD URLCacheInterval parameter**
 - The CRL is removed from the cache when
 - The nextUpdate time in the CRL is reached
 - The URLCacheInterval is reached
 - The MODIFY nssd,REFRESH command is issued

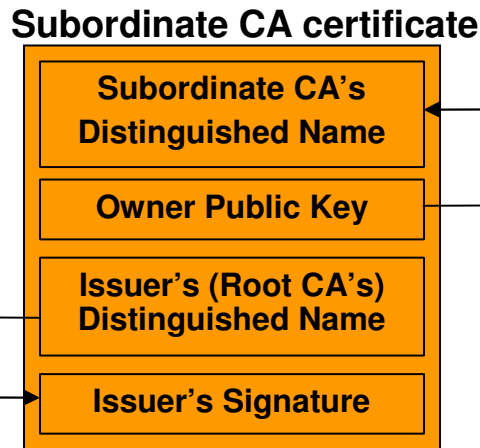
Agenda

- Background
 - Cryptography basics
 - Digital certificates
 - IP Security (IPSec) / Internet Key Exchange (IKE)
- Certificate use in the IKE
 - Authentication
 - Negotiation modes
- Network Security Services Daemon (NSSD)
 - Overview
 - Digital signature operations
 - Advanced certificate services
 - Certificate revocation
 - **Certificate trust chains**
 - IKEv2 certificate encoding types
- Implementing the NSSD solution

Certificate trust chains



- A **digital certificate** is issued (signed) by a trusted certificate authority or is self-signed
- A **certificate authority** can be a root certificate authority or a subordinate certificate authority.

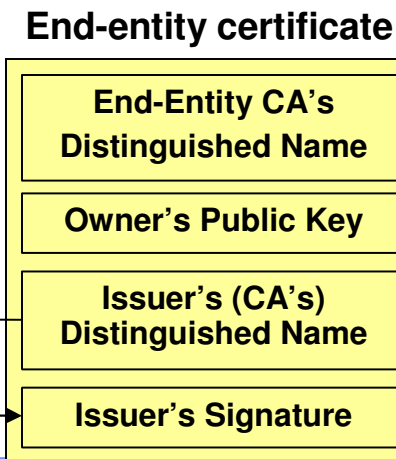


- Each certificate authority has its own public/private key pair that is bound by its own certificate.

Chain of trust

Verify signature

Verify signature

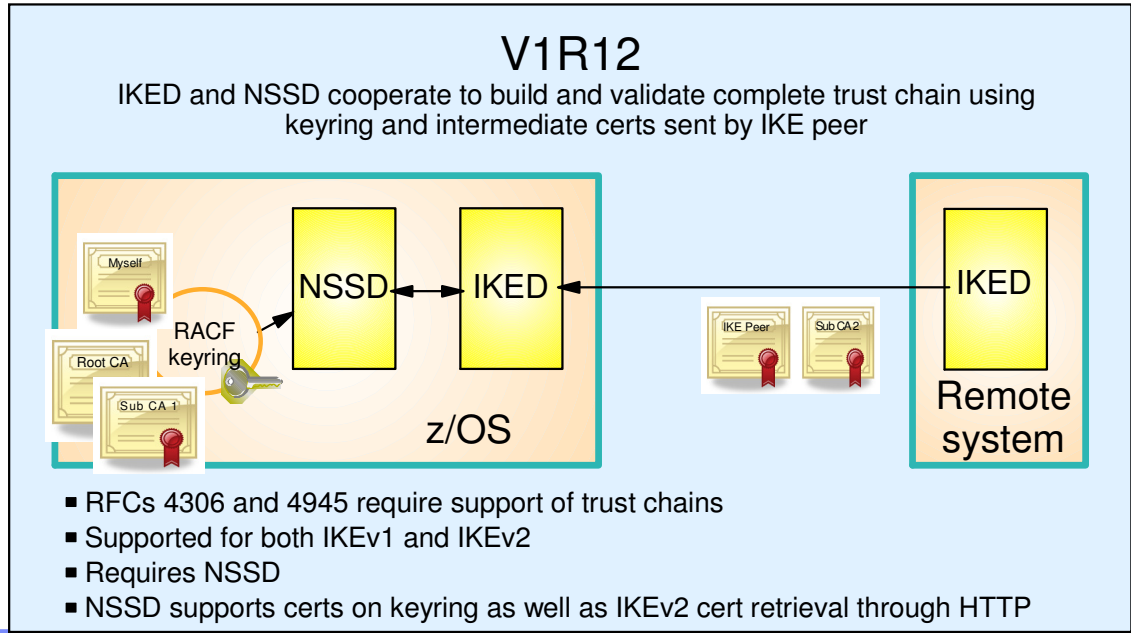
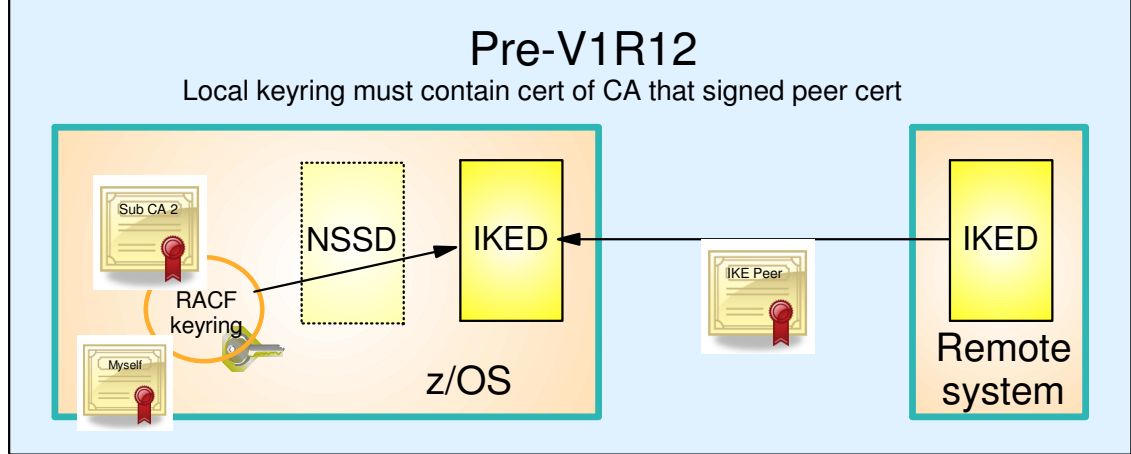
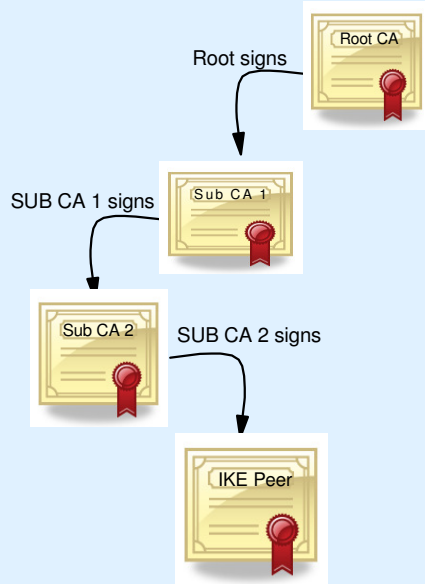


- A **subordinate certificate authority** is one that has been delegated the responsibility to issue (sign) certificates on behalf of another certificate authority.
 - For example, an enterprise can use subordinate CAs to allow geographic regions and departments to manage their own certificates

IPSec support for certificate trust chains

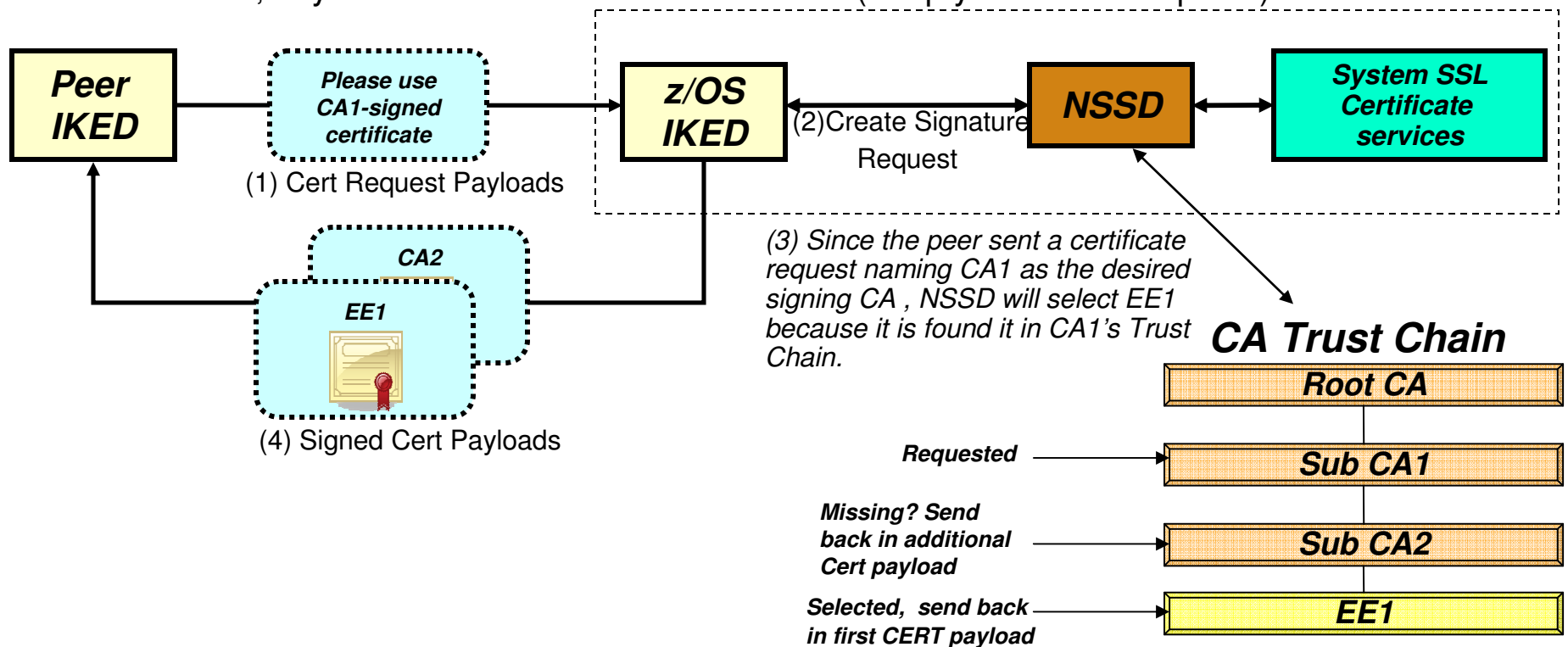
Eases administrative requirements by reducing the number of subordinate CA certificates needed on IKE keyrings

Given the following certificate hierarchy:



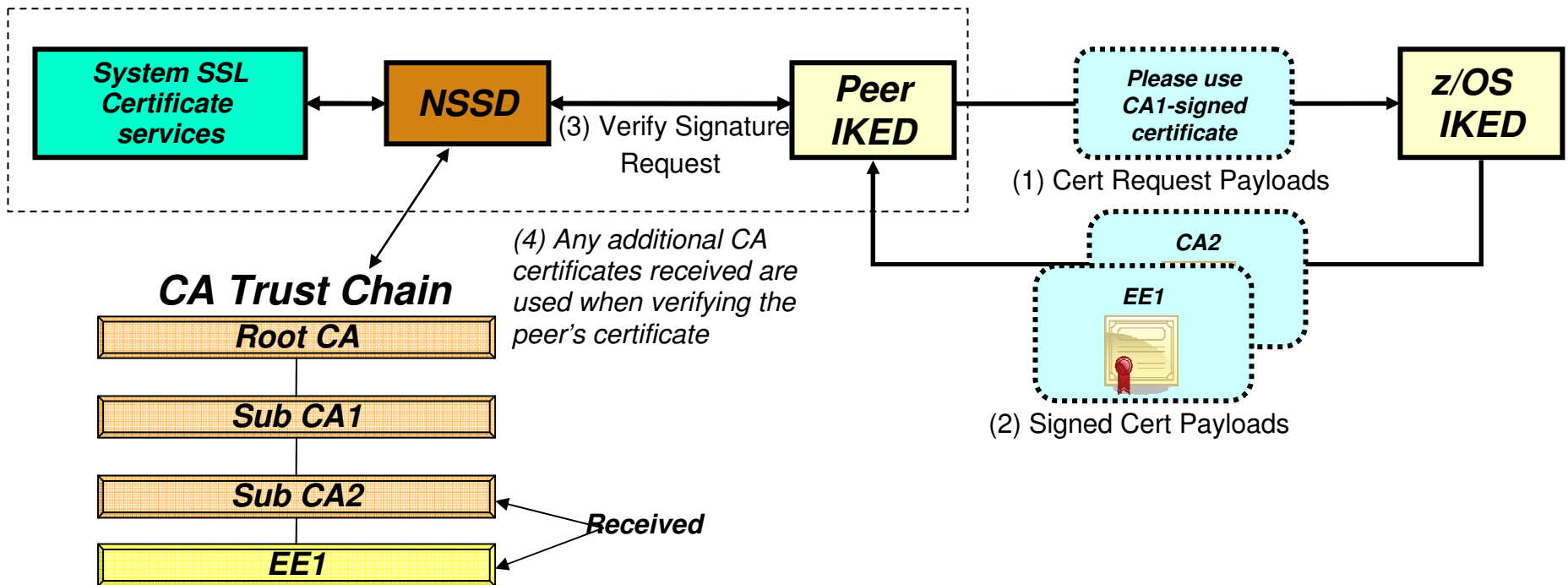
Using Trust Chains to select an end entity certificate with NSSD

- **When receiving certificate request payloads, NSSD attempts to select a certificate within the trust chain of the requested CA**
 - Trust chain support will additionally return the necessary set of subordinate CA certificates needed to fill in any gaps between the named CA and the end entry certificate
- **If a certificate cannot be found within the trust chain, the handling is determined by the IKE protocol version**
 - For IKEv1, the negotiation fails
 - For IKEv2, any viable certificate found is selected (“empty certificate request”)



Using Trust Chains during signature verification with NSSD

- **When receiving certificate payloads to use in verifying a signature.**
 - The first certificate payload contains an end-entity certificate
 - Any other certificate payloads received containing CA certificates are used to complete the CA trust chain



Agenda

- Background
 - Cryptography basics
 - Digital certificates
 - IP Security (IPSec) / Internet Key Exchange (IKE)
- Certificate use in the IKE
 - Authentication
 - Negotiation modes
- Network Security Services Daemon (NSSD)
 - Overview
 - Digital signature operations
 - Advanced certificate services
 - Certificate revocation lists
 - Certificate trust chains
 - **IKEv2 certificate encoding types**
- Implementing the NSSD solution

IKEv2 certificate encoding types - Hash and URL encoding of certificate and certificate bundles

- **IKEv2 support includes new certificate payloads encoded using hash and URL encoding**
 - Hash and URL encoding is an alternative to transmitting the actual certificates
 - Amount of data carried on the IKE certificate payload flow is reduced
 - Offsets the potential for increased data introduced by Certificate Trust Chain support
- **The URL points to the location of either a certificate or “certificate bundle” in a binary file on HTTP server**
 - Facilitates creation of a shared public key infrastructure by using HTTP servers as digital certificate repositories
 - Certificates / certificate bundles can be retrieved using HTTP protocols by anyone who knows the URL and has network access to the HTTP server

Support for new certificate repository formats

- **New certificate repository formats introduced with this support**
 - Binary file that contains a DER-encoded X.509 certificate
 - Binary file that contains an “X.509 certificate bundle”
- **An X.509 certificate bundle is a binary file that contains a collection of some or all of the following:**
 - End-entity certificates
 - CA certificates
 - Certificate revocation lists
 - Used if HTTP server named in CRLDistributionPoints extension cannot be reached
- **New z/OS certbundle UNIX command creates X.509 certificate bundle files containing**
 - X.509 certificates retrieved from the key ring (identified by label)
 - CRLs from z/OS UNIX files
- **Output bundle files can be placed on an HTTP server**
 - Add a corresponding CertificateBundleURL statement to the NSSD configuration file, with the URL of the bundle file

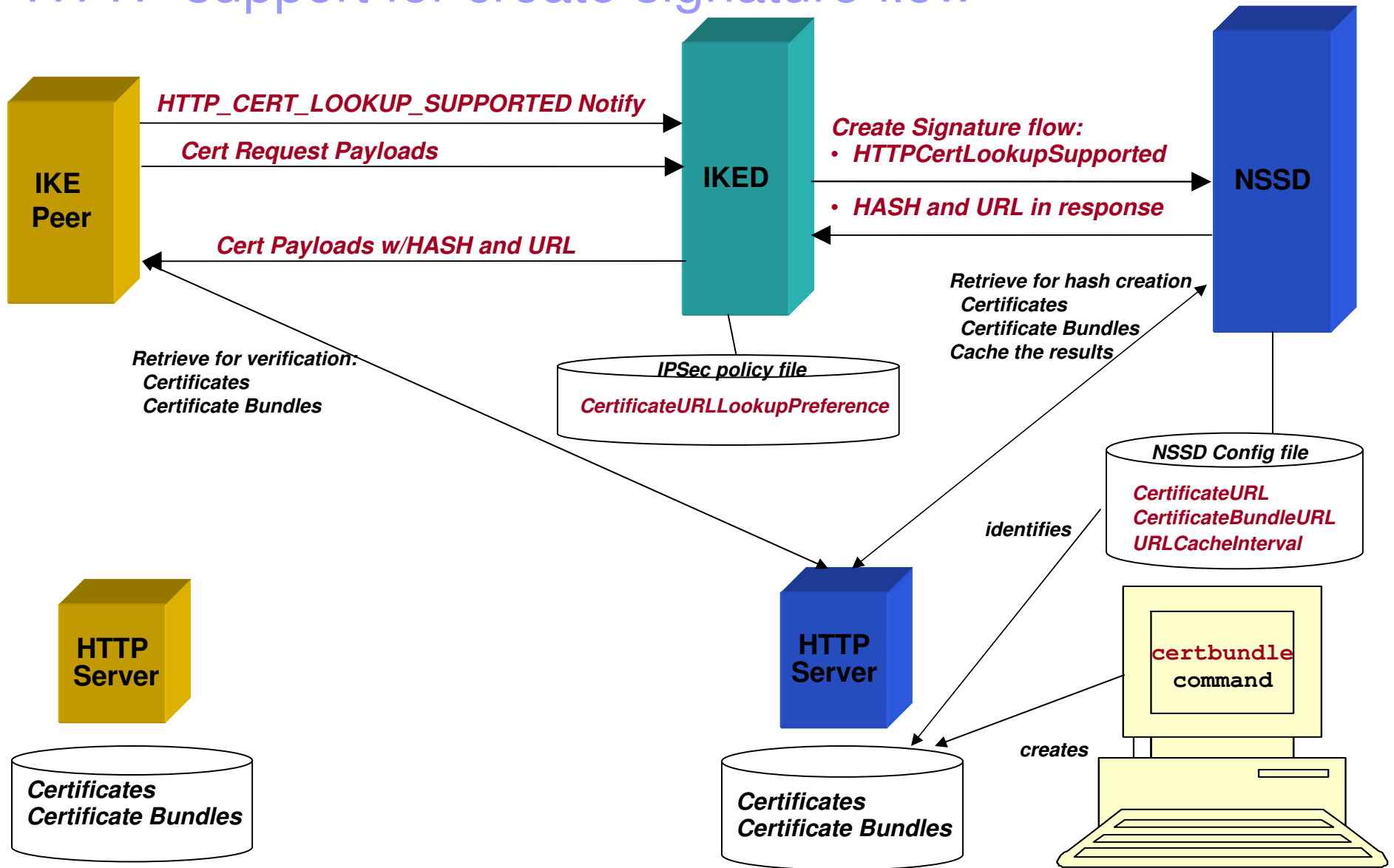
IKEv2 processing of hash and URL encoding

- **IKEv2 flows include transmission of new certificate payloads in IKE messages**
 - Certificate payloads can be encoded in several ways, including:
 - **Encoding 4** is the entire binary X.509 certificate
 - Can be quite long (several hundred bytes)
 - **Encoding 12** is the hash of a certificate, plus the URL where the certificate resides
 - **Encoding 13** is the hash of a certificate bundle, plus the URL of the bundle
 - These payloads flow encrypted in the IKEv2 messages
 - After the receiver retrieves the certificate, it verifies the integrity of the certificate by recreating the hash and comparing it to the hash value from the certificate payload which is digitally signed by IKE peer
- **IKEv2 peers indicate support for encodings 12 and 13 to each other**
 - Notify type *HTTP_CERT_LOOKUP_SUPPORTED*
- **Use of Hash and URL encodings has an effect on performance**
 - Reduces the size of the IKE messages
 - May increase the actual time to process
 - Latency introduced by certificate / certificate bundle retrieval from the HTTP server
 - Caching retrieved URL data helps reduce this time

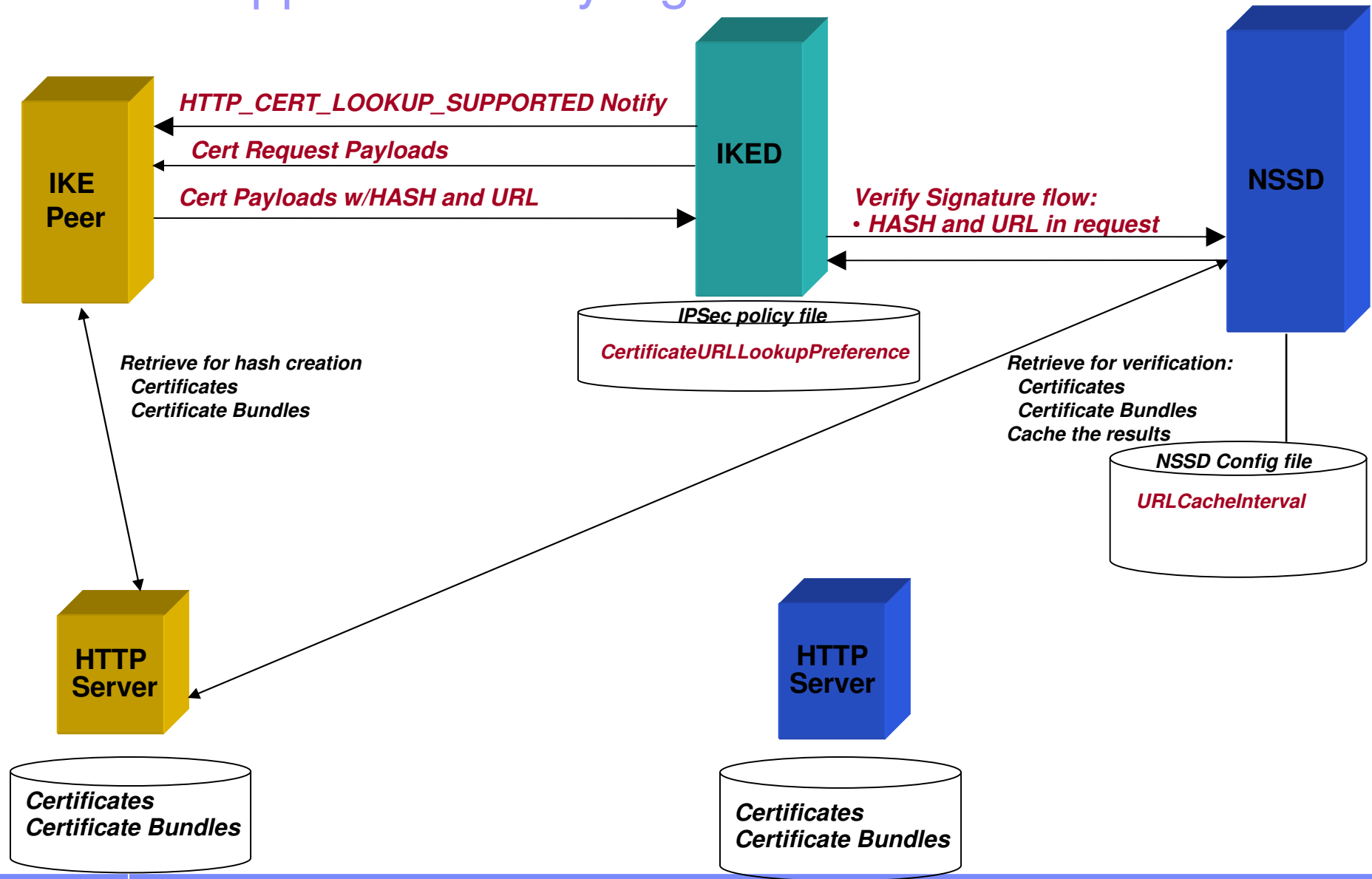
HTTP processing controls

- **Enabling / disabling HTTP lookup**
 - **CertificateURLLookupPreference** keyword of KeyExchangeAction
 - *Allow* - send and request URLs
 - *Tolerate* - send URLs, don't request
 - *Disallow* - do not send or request
 - Also available on KeyExchangePolicy
 - Sets the stack level default
- **NSSD configuration file URL data specifications**
 - **CertificateURL** and **CertificateBundleURL** statements
 - Define the label of a local certificate on the key ring, and the corresponding URL of the certificate or bundle file to be used by peer nodes for HTTP retrieval
 - **URLCacheInterval** statement
 - Defines the number of minutes that retrieved URL data is cached
 - Default: 1 week
 - 0 means do not cache
 - All of these statements can be modified dynamically
 - `MODIFY nssd,REFRESH`

HTTP support for create signature flow



HTTP support for verify signature flow



z/OS® V1R12 Communications Server

Implementing the NSSD solution



Implementing Certificate Management in NSSD

Basic configuration for creating and verifying digital signatures

- 1 Configure the Network Security Server (nssd.conf)
- 2 Configure NSS client stacks (iked.conf)
- 3 Configure AT-TLS policy for the IKED NSS client
- 4 Configure AT-TLS policy for NSSD
- 5 Install permit filter rules for the IKED NSS client (outbound TCP, remote port 4159)
- 6 Install permit filter rules for NSSD (inbound TCP, local port 4159)
- 7 Authorize IKED NSS clients to specific NSS services and certificates using SAF profiles
- 8 Optional support for retrieval of CRLs from web server
 - URL Cache interval
 - Update IKE policy to indicate level of CRL checking (revocation level)
 - Install permit filter rules for reaching HTTP server (outbound TCP, remote port 80)
- 9 Create NSSD keyring
- 10 Install end entity certificates at NSSD host

The Configuration Assistant for z/OS Communications Server can assist with steps 1-8 above.

Implementing hash and URL encoding support in NSSD



- 1 Install permit filter rules for reaching HTTP server (outbound TCP, remote port 80)
- 2 To store certificates on a web server:
 - Export end entity certificates from NSSD keyring to an MVS dataset (using DER-encoding)
 - FTP DER-encoded certificates to a web server
 - Update NSSD configuration with URL of certificate
- 3 To store certificate bundles on a web server
 - Use certbundle command to populate a certificate bundle with one or more of the following:
 - End entity certificate
 - Intermediate CA certificates
 - CRLs
 - FTP certificate bundle to a web server
 - Update NSSD configuration with URL of certificate bundle
- 4 Update IPsec policy to indicate level of HTTP support

Summary

The table below summarizes the support for digital certificates in IKED and NSSD and identifies the supported configurations

Function	IKEv1 Local	IKEv1 with NSSD	IKEv2 with NSSD
Rivest-Shamir-Adleman (RSA) Digital Signature algorithm	✓	✓	✓
Certificate Revocation Lists		✓	✓
Locate a certificate within a CA's trust chain		✓	✓
Send missing CA certificates to a peer		✓	✓
Use received CA certificates when validating a peer's signature		✓	✓
Hash and URL encoding for certificate payload			✓
Treat as having an empty certificate request payload			✓
Elliptic Curve Digital Signature Algorithm (ECDSA)			✓

For more information

URL	Content
http://www.twitter.com/IBM_Commserver 	IBM Communications Server Twitter Feed
http://www.facebook.com/IBMCommserver 	IBM Communications Server Facebook Fan Page
http://www.ibm.com/systems/z/	IBM System z in general
http://www.ibm.com/systems/z/hardware/networking/	IBM Mainframe System z networking
http://www.ibm.com/software/network/commserver/	IBM Software Communications Server products
http://www.ibm.com/software/network/commserver/zos/	IBM z/OS Communications Server
http://www.ibm.com/software/network/commserver/z_lin/	IBM Communications Server for Linux on System z
http://www.ibm.com/software/network/ccl/	IBM Communication Controller for Linux on System z
http://www.ibm.com/software/network/commserver/library/	IBM Communications Server library
http://www.redbooks.ibm.com	ITSO Redbooks
http://www.ibm.com/software/network/commserver/zos/support/	IBM z/OS Communications Server technical Support – including TechNotes from service
http://www.ibm.com/support/techdocs/atmastr.nsf/Web/TechDocs	Technical support documentation from Washington Systems Center (techdocs, flashes, presentations, white papers, etc.)
http://www.rfc-editor.org/rfcsearch.html	Request For Comments (RFC)
http://www.ibm.com/systems/z/os/zos/bkserv/	IBM z/OS Internet library – PDF files of all z/OS manuals including CommServer