Hunting crypto secrets in SAP systems

Martin Gallo,

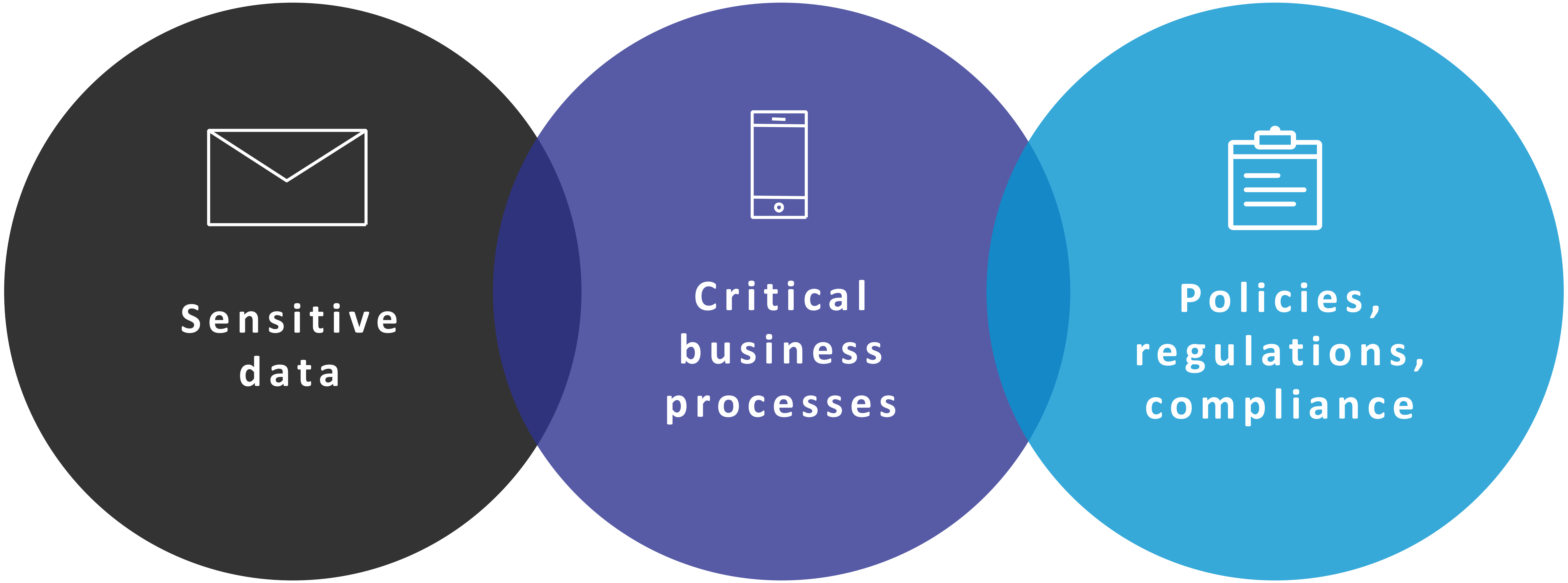
Product Owner/Security Researcher

AGENDA

- Problem definition
- Cryptographic material
- Personal Security Environment (PSE)
- SSO credentials (cred_v2)
- Local Protection Store (LPS)
- Putting everything together
- Recommendations
- Conclusions

SECUREAUTH + CORE SECURITY

Problem: Secure business processes and data



Sensitive data

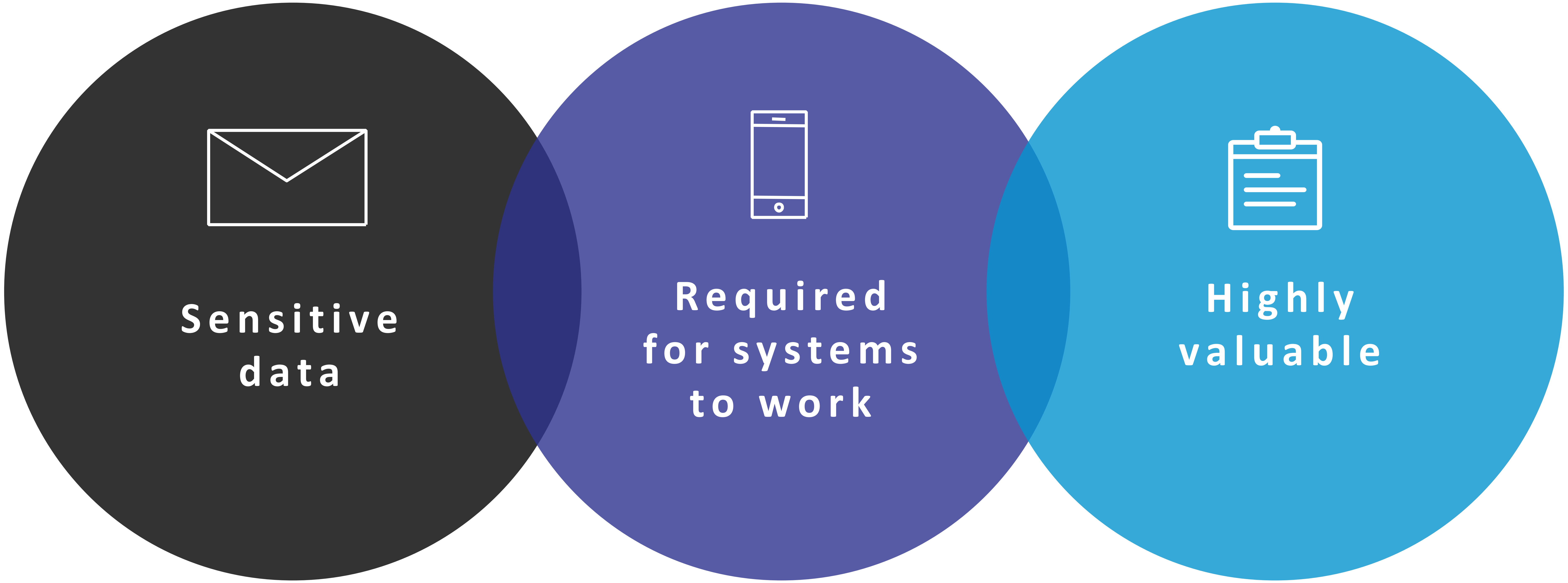
Critical business processes

Policies, regulations, compliance

Solution: Crypto all the things!

- Encrypt data at transit
  - Secure communication paths
- Encrypt data at rest
  - Encrypt databases and stores
- Strong authentication
  - Auth protocols, SSO, etc.
- Integrity
- Digital signature

New problem: Secure crypto material



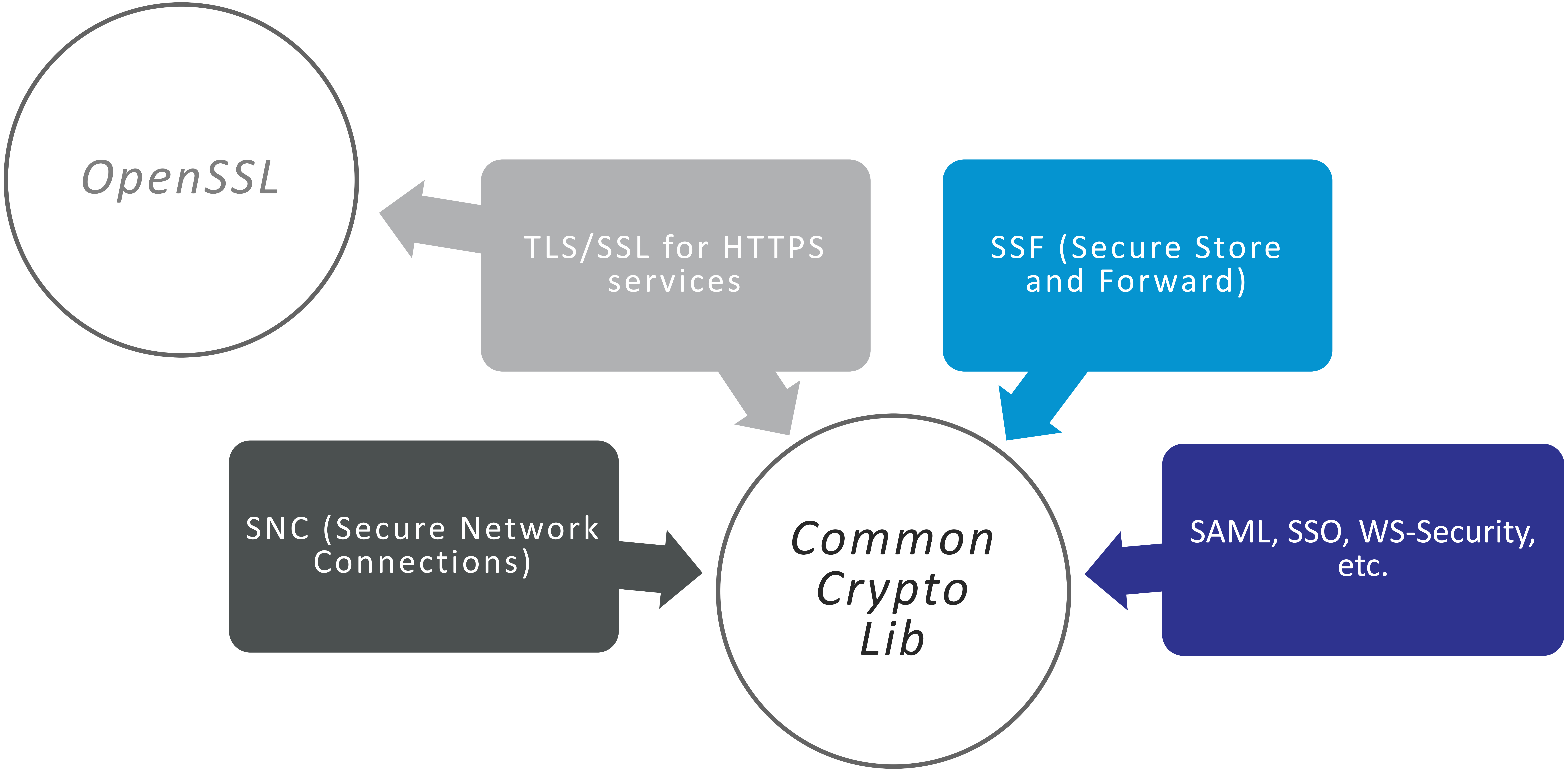
Sensitive data

Required for systems to work

Highly valuable

Cryptographic material in SAP environments

Cryptographic libraries



OpenSSL

TLS/SSL for HTTPS services

SSF (Secure Store and Forward)

SNC (Secure Network Connections)

Common Crypto Lib

SAML, SSO, WS-Security, etc.

OpenSSL

- Standard open source cryptographic library
- Used for TLS/SSL communications in
  - SAP HANA
  - SAP BusinessObjects BI
  - …
- Protection of keys and certificates is well known
  - PKCS#5-7-8-11-12, PEM/DER, etc.
- **In SAP HANA deprecated starting with SP09**
- Not the focus of this talk

Migration from OpenSSL to CommonCryptoLib - SAP Note 2093286

CommonCryptoLib

- Cryptographic library for all SAP components
  - SAP Netweaver
  - SAP HANA
  - Auxiliary components, services and tools
- Replaces all old libraries (full backward compatible)
  - SECUDE
  - SAP Security Library (SAPSECULIB)
  - SAP Cryptographic Library (SAPCRYPTOLIB)
  - Secure Login Library
- FIPS 140-2 crypto kernel available
- Interface for Hardware Security Module (HSM)

Central Note for CommonCryptoLib 8 (SAPCRYPTOLIB) - SAP Note 1848999

CommonCryptoLib use cases

Communication paths

- TLS/SSL for HTTPS
- SNC for RFC, Diag, Router, etc.
- WS-Security for SOAP

Authentication

- NW SSO
- SAML, JWT, SAP Logon
- Kerberos/SPNEGO

Digital Signature and encryption (SSF)

- Human Capital Management
- Production Planning – Process Industry
- Product Data Management
- SAP ArchiveLink Content Server

Central Note for CommonCryptoLib 8 (SAPCRYPTOLIB) - SAP Note 1848999

Cryptographic material

Passwords

- *CODVN**
- *RFC*
- *HDB User store*
- *SAP GUI Shortcuts*
- *Configuration files*

Certificates and Keys

- **HTTPS/TLS/SSL**
- **SNC**
- **SSF**
- **SAML**
- **JSON Web Token (JWT)**
- **SAP Logon tickets**

Private Keys

- *ABAP/Java Secure Storage*
- *SAP HANA Server-Side Data Encryption*

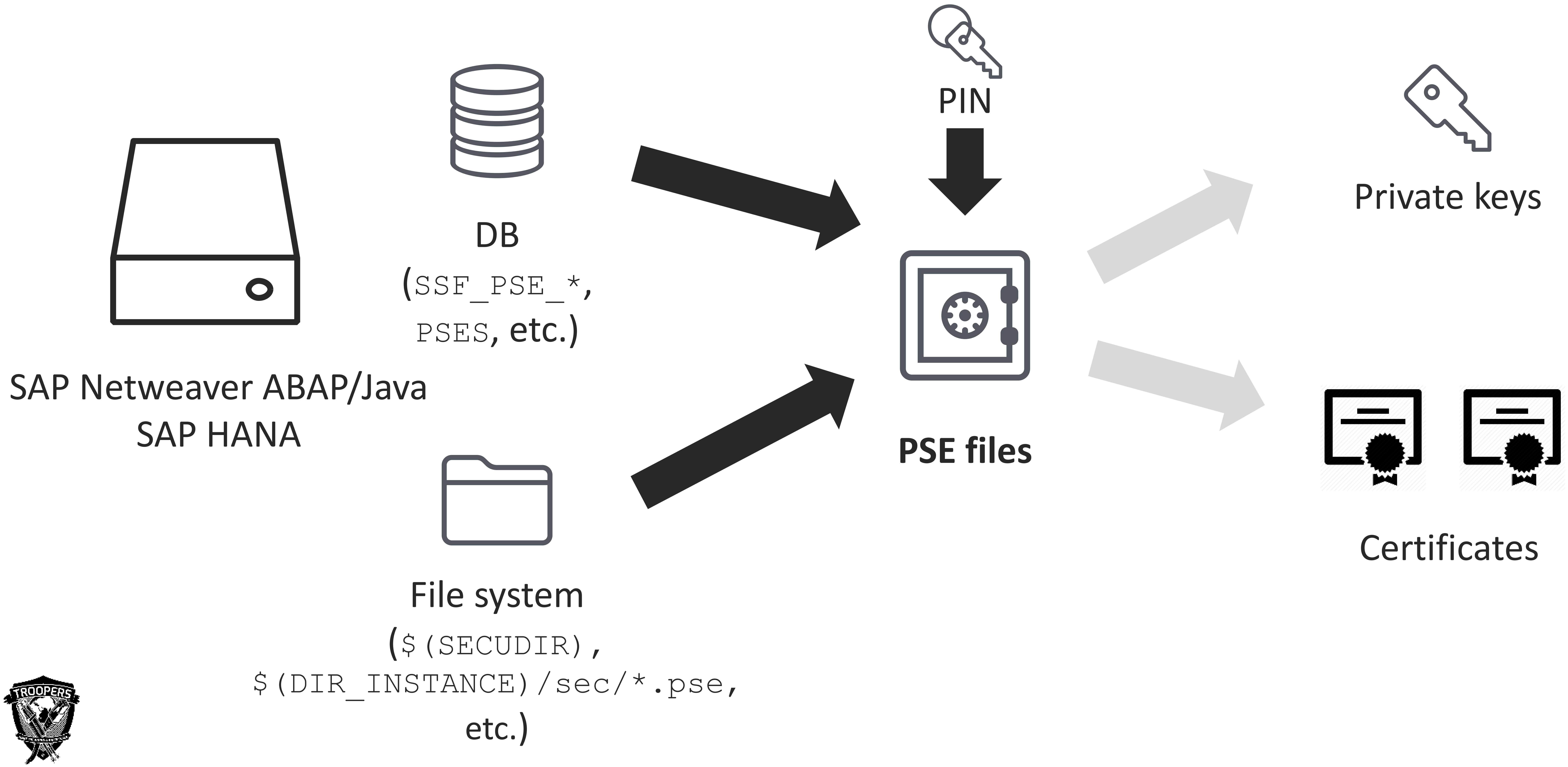
ERPScan's All your passwords belong to us – Confidence 2014

Personal Security Environment (PSE)

PSE

- Storage format for cryptographic objects
  - X.509 Certificates
  - **Private Keys**
  - Certificate revocation lists
- Defined as part of SecuDE(Security Development Environment)
- Similar to PKCS#12

PSE



SAP Netweaver ABAP/Java
SAP HANA

DB
(`SSF_PSE_*`,
`PSES`, etc.)

File system
(`$(SECUDIR)`,
`$(DIR_INSTANCE)/sec/*.pse`,
etc.)

PIN

PSE files

Private keys

Certificates

SecuDE

- Portable general-purpose security toolkit
- Developed by GMD
- Define several cryptographic libraries and utilities
- Included definition of PSE (Personal Security Environment)
- Provided ASN.1 definitions as well as a reference implementation

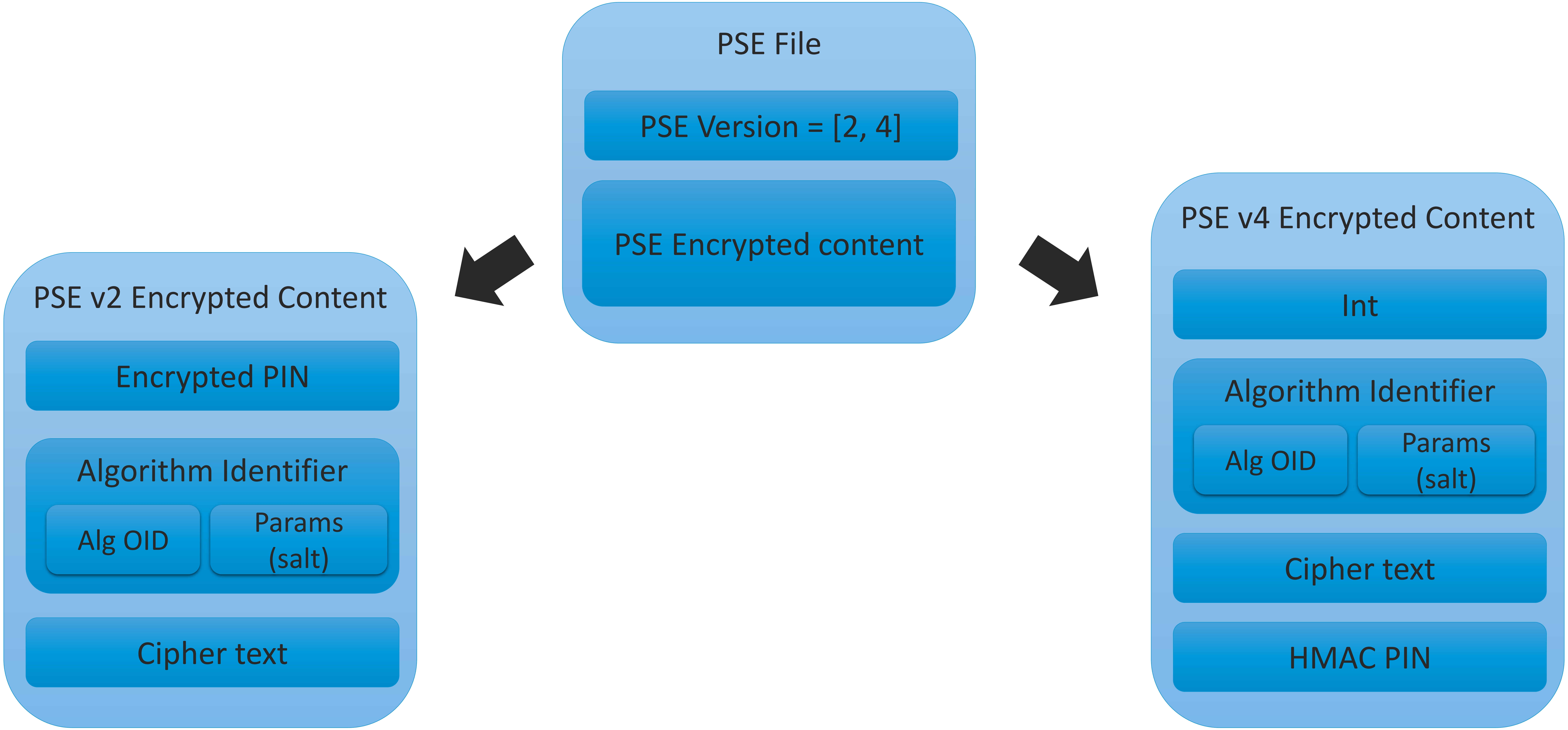
PSE locations

- File system
  - $(SECUDIR)
  - $(DIR_INSTANCE)/sec/*.pse
- SAP Netweaver ABAP Database
  - Table SSF_PSE_T/SSF_PSE_D (data)
  - Table SSF_PSE_H (metadata)
- SAP HANA (>=SPS10)
  - In-database storage
  - CREATE/ALTER/DROP/SET/UNSET PSE statements
  - CERTIFICATES/PSE_CERTIFICATES/PSES views/tables

PSE file format

- Main versions
  - v2: default (since beginning of times?)
  - v4: added in SAPCRYPTOLIB 555pl19 (May 2007)
- ASN.1 structure
- **PIN**-protected
- **Encrypted** with PKCS#12/5
  - PBE1 (*PKCS#12*) / PBE2
  - PKCS#7 Padding
- Optional Local Protection Storage (LPS)
- *Key strength depends on PIN complexity/entropy*

PSE file format



PSE File

PSE Version = [2, 4]

PSE Encrypted content

PSE v2 Encrypted Content

Encrypted PIN

Algorithm Identifier

Alg OID

Params (salt)

Cipher text

PSE v4 Encrypted Content

Int

Algorithm Identifier

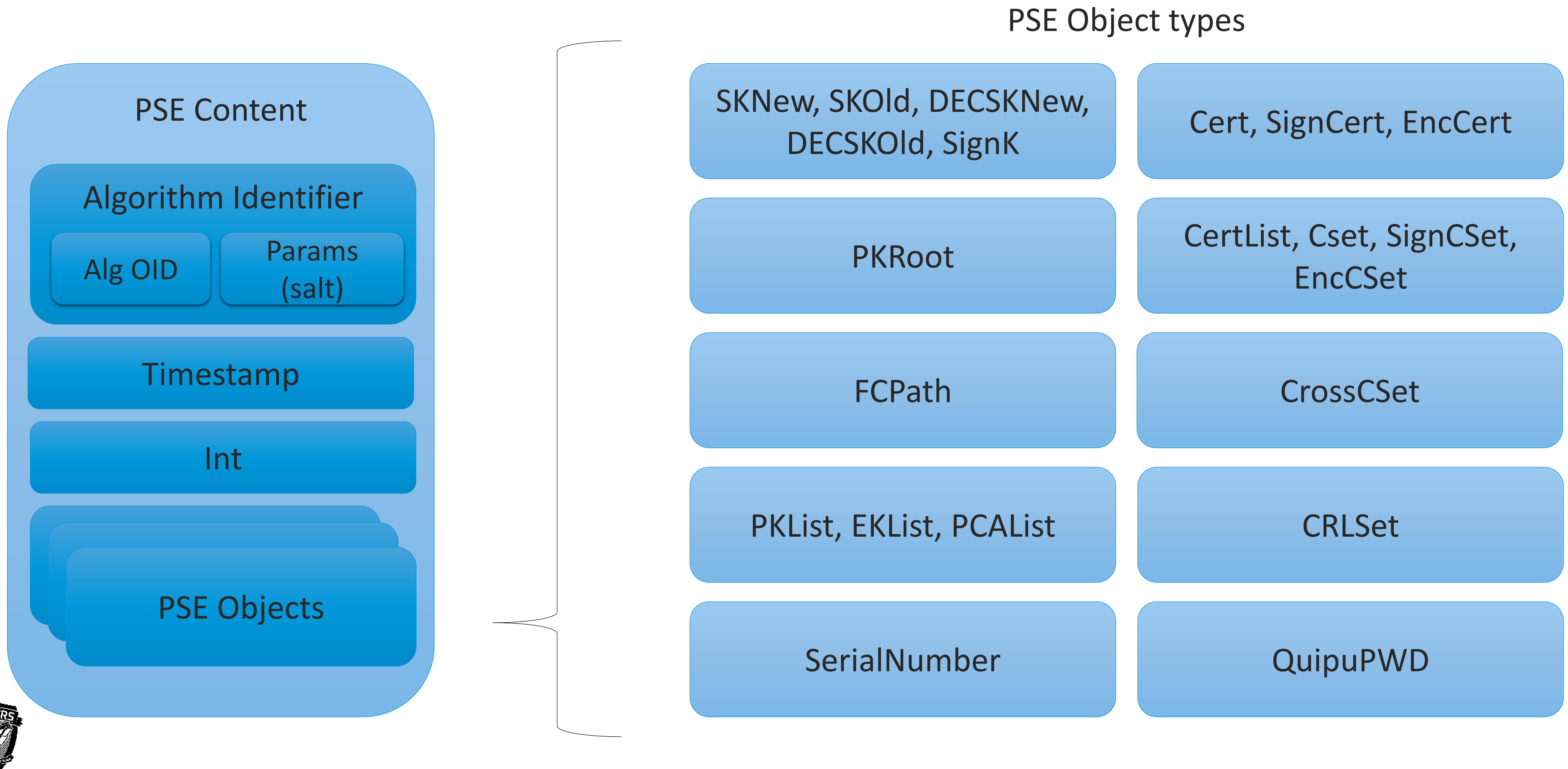
Alg OID

Params (salt)

Cipher text

HMAC PIN

PSE content format



PSE Content

Algorithm Identifier

Alg OID

Params (salt)

Timestamp

Int

PSE Objects

PSE Object types

SKNew, SKOld, DECSKNew, DECSKOld, SignK

Cert, SignCert, EncCert

PKRoot

CertList, Cset, SignCSet, EncCSet

FCPath

CrossCSet

PKList, EKList, PCAList

CRLSet

SerialNumber

QuipuPWD

PSE decryption algorithms

PBES1-3DES-SHA1

```
DerivedKey, IV = PBKDF1(SHA1, Iterations, Salt, PIN)

EncryptedPIN = 3DES(DerivedKey, IV, PIN)
PSEContent = 3DES(DerivedKey, IV, PSEEncCont)
```

** PBKDF1 as defined in PKCS#12*

PBES2-AES256-SHA1/SHA256

PBES2 based on standard PKCS#5

PSE encryption overview

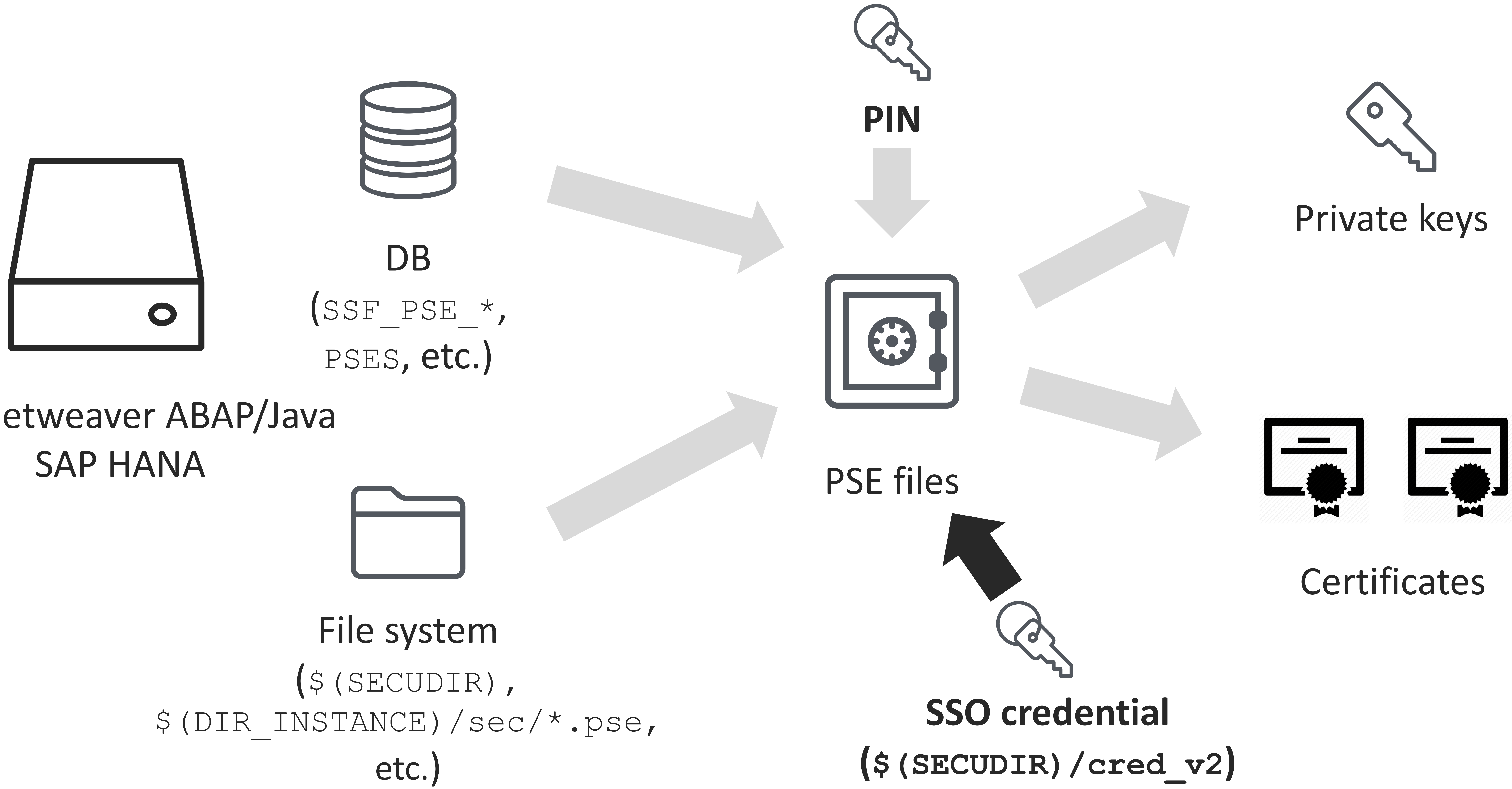
Encryption mechanism	Default iterations	Key Strenght	CommonCryptoLib version
PBES1-3DES-SHA1 (*PKCS#12*)	2048 (10000 in >= 8.5.15)	168 bits	< 8.5.15
PBES2-AES256-SHA1	10000	256 bits	>= 8.5.15 (Aug 2017)
PBES2-AES256-SHA256	10000	256 bits	>= 8.5.15 (Aug 2017)

SSO Credential (cred_v2)

SSO Credential (cred_v2)

- Single-sign-on experience for PSE files
- Storage of PIN to decrypt PSEs
- ASN.1 structure
- **Encrypted** for the current username
  - 3DES/AES/DPAPI
- Optional Local Protection Storage (LPS)
- *Renders PSE encryption ineffective if note secured properly*

SSO Credential (cred_v2)



SAP Netweaver ABAP/Java
SAP HANA

DB
(`SSF_PSE_*`,
`PSES`, etc.)

File system
(`$(SECUDIR)`,
`$(DIR_INSTANCE)/sec/*.pse`,
etc.)

PIN

PSE files

SSO credential
(`$(SECUDIR)/cred_v2`)

Private keys

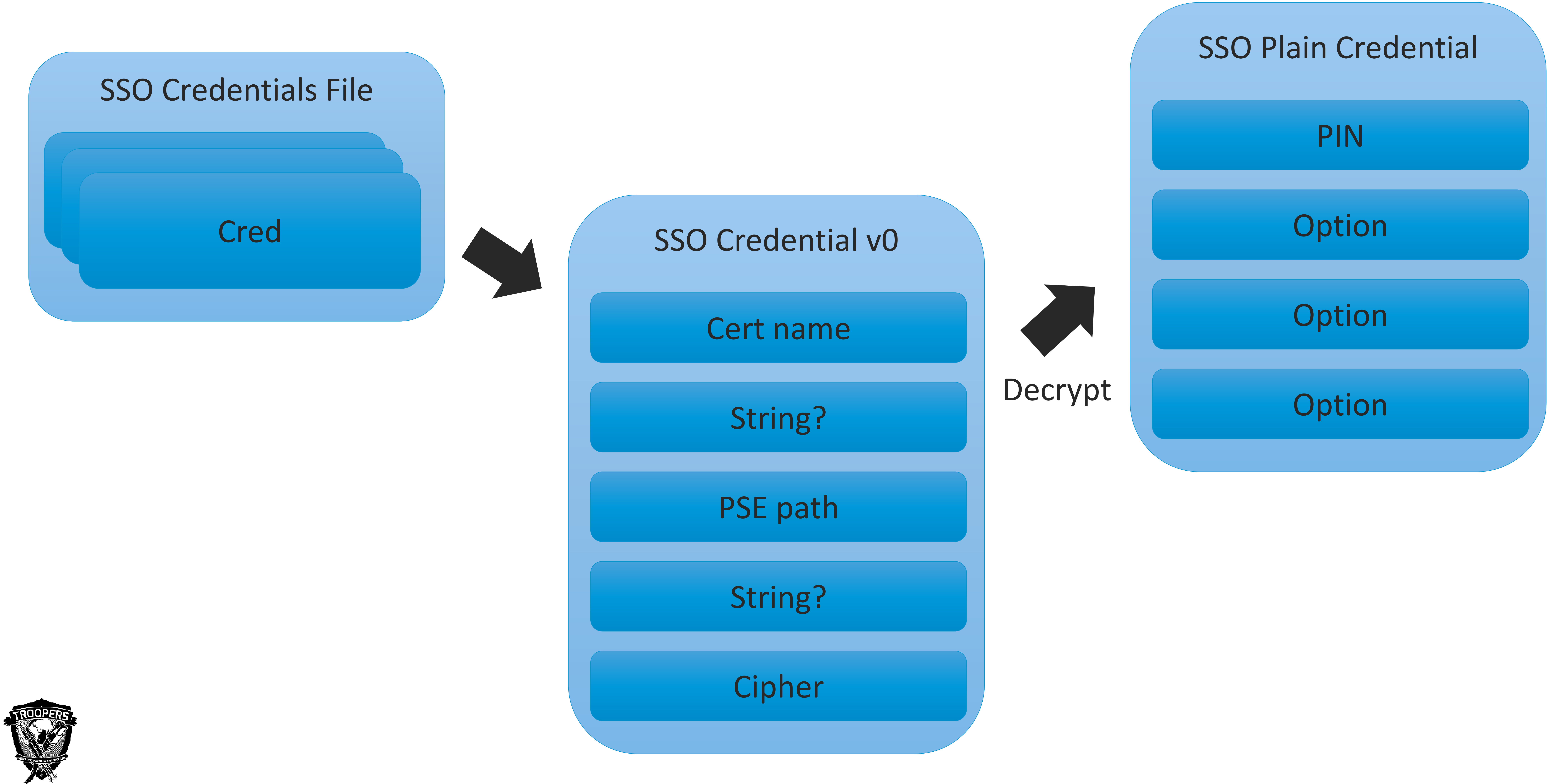
Certificates

SSO Credential file format v0

Version 0 format

- Default in CCL version < 8.5.15
- PIN encrypted with 3DES
- Encryption key obtained from
  - **Hardcoded** string
  - Formatted with the username

SSO Credential file format v0



SSO Credentials File

Cred

SSO Credential v0

Cert name

String?

PSE path

String?

Cipher

Decrypt

SSO Plain Credential

PIN

Option

Option

Option

SSO Credential v0 decryption algorithm

```
IV = “00000000”
Key = “<fixed key>” % username

PIN = 3DES(Key[:24], IV, EncryptedPIN)
```

SSO Credential file format v0

Version 0 format

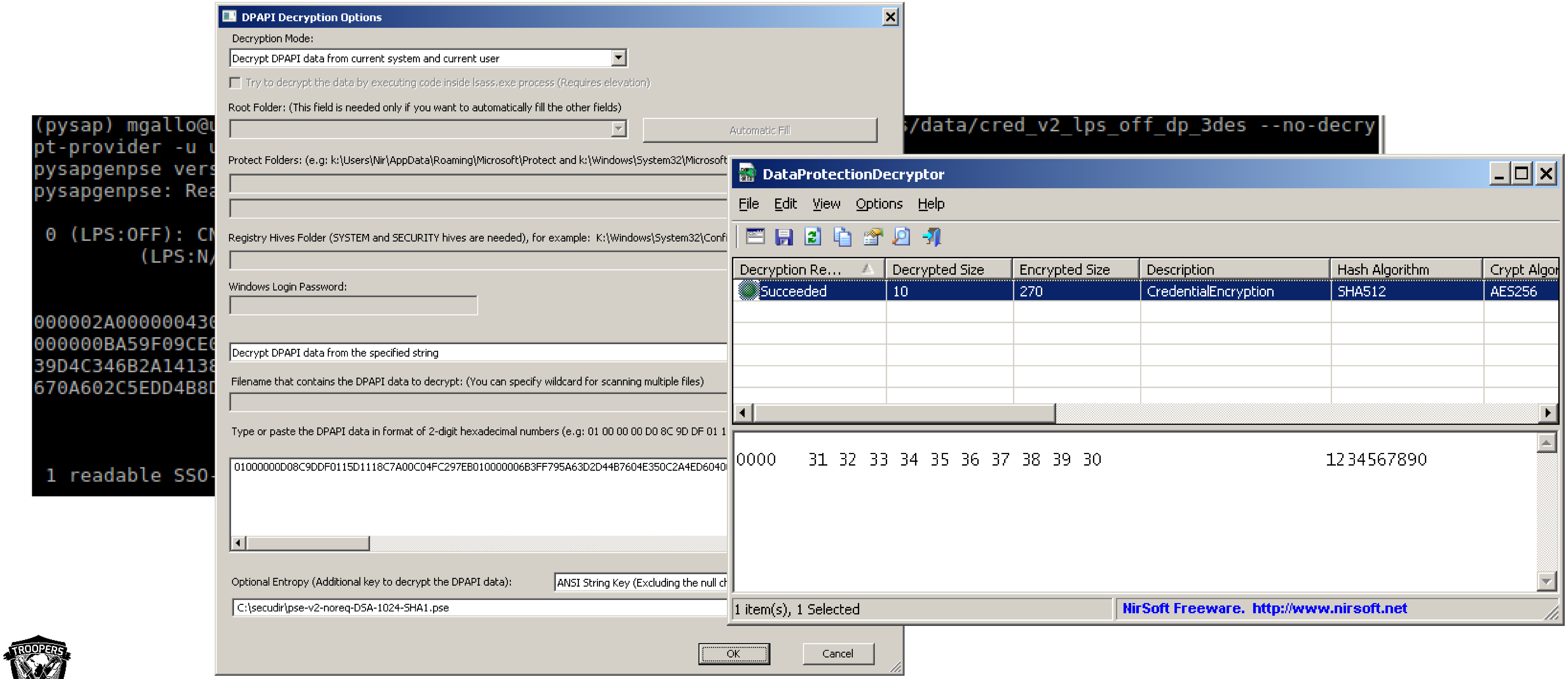
- On Windows platforms, uses DP API by default
  - Same encryption algorithm/key derivation
  - Encrypted blob in the file is **encrypted with Windows DP API**
    - Additional entropy is the PSE path

Windows DP API

- Data Protection API provided by Windows
- Available since Windows 2000
- Designed for symmetric encryption of asymmetric private keys
- Security relies on access to Windows’ user account
- Some research and attacks published between 2010-2012
- Offline decryption tools available

MSDN - Data Encryption and Decryption Functions

Windows DP API



```
(pysap) mgallo@u                                              /data/cred_v2_lps_off_dp_3des --no-decry
pt-provider -u u
pysapgenpse vers
pysapgenpse: Rea

 0 (LPS:OFF): CN
        (LPS:N/


000002A000000043
000000BA59F09CE
39D4C346B2A14138
670A602C5EDD4B8D


 1 readable SSO-
```

DPAPI Decryption Options

Decryption Mode: Decrypt DPAPI data from current system and current user

□ Try to decrypt the data by executing code inside lsass.exe process (Requires elevation)

Root Folder: (This field is needed only if you want to automatically fill the other fields)

Protect Folders: (e.g: k:\Users\Nir\AppData\Roaming\Microsoft\Protect and k:\Windows\System32\Microsoft

Registry Hives Folder (SYSTEM and SECURITY hives are needed), for example: K:\Windows\System32\Confi

Windows Login Password:

Decrypt DPAPI data from the specified string

Filename that contains the DPAPI data to decrypt: (You can specify wildcard for scanning multiple files)

Type or paste the DPAPI data in format of 2-digit hexadecimal numbers (e.g: 01 00 00 00 D0 8C 9D DF 01 1

```
01000000D08C9DDF0115D1118C7A00C04FC297EB010000006B3FF795A63D2D44B7604E350C2A4ED6040
```

Optional Entropy (Additional key to decrypt the DPAPI data): ANSI String Key (Excluding the null ch

C:\secudir\pse-v2-noreq-DSA-1024-SHA1.pse

DataProtectionDecryptor

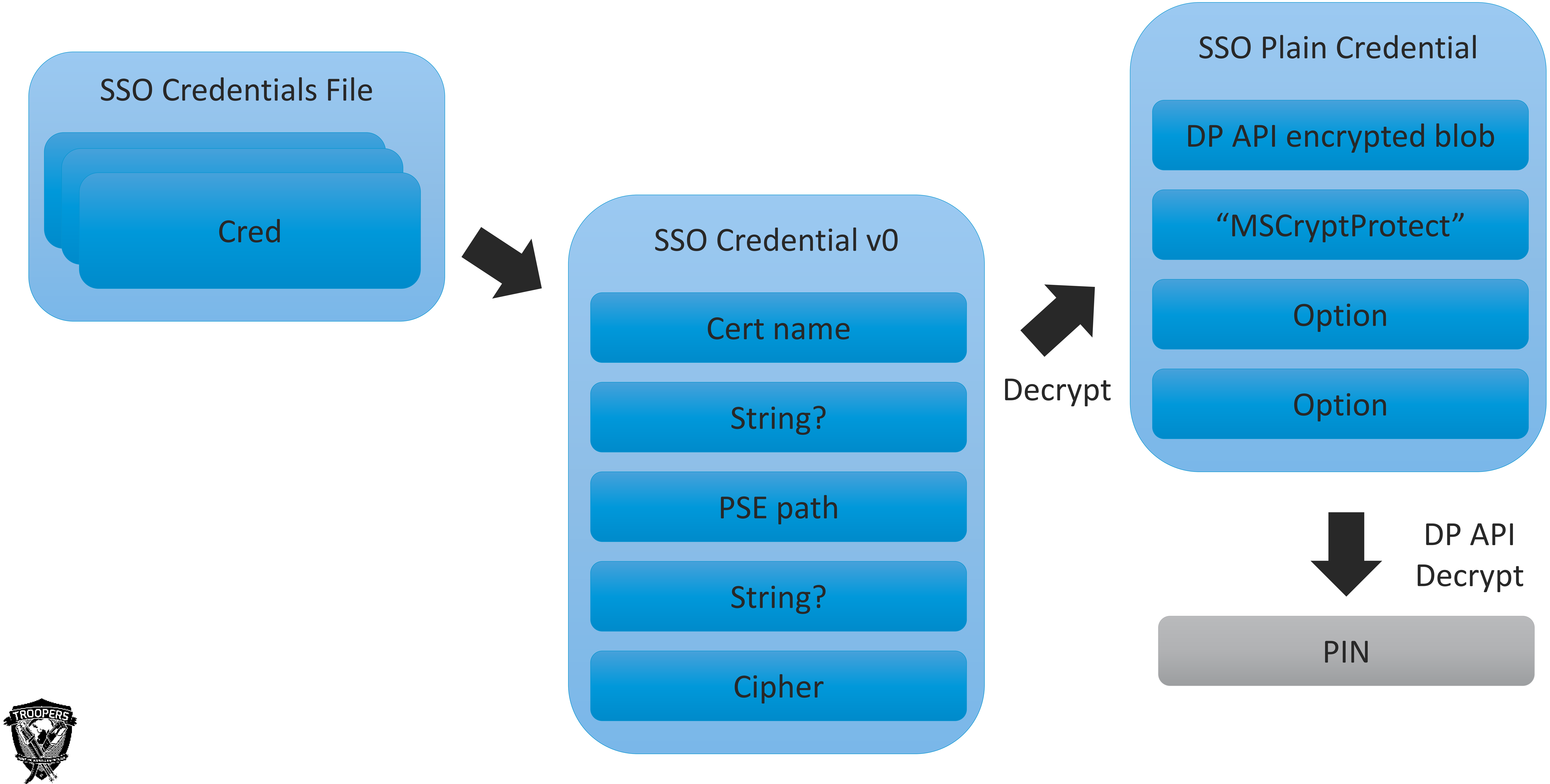
Decryption Re...	Decrypted Size	Encrypted Size	Description	Hash Algorithm	Crypt Algor
Succeeded	10	270	CredentialEncryption	SHA512	AES256

```
0000    31 32 33 34 35 36 37 38 39 30                    1234567890
```

1 item(s), 1 Selected

NirSoft Freeware. http://www.nirsoft.net

SSO Credential file format v0



SSO Credentials File

Cred

SSO Credential v0

Cert name

String?

PSE path

String?

Cipher

Decrypt

SSO Plain Credential

DP API encrypted blob

“MSCryptProtect”

Option

Option

DP API Decrypt

PIN

SSO Credential v0 decryption algorithm

```
IV = “00000000”
Key = “<fixed key>” % username

DPAPIEncryptedBlob = 3DES(Key[:24], IV, EncryptedPIN)

PIN = DPAPIUnprotect(DPAPIEncryptedBlob, PSEPath)
```

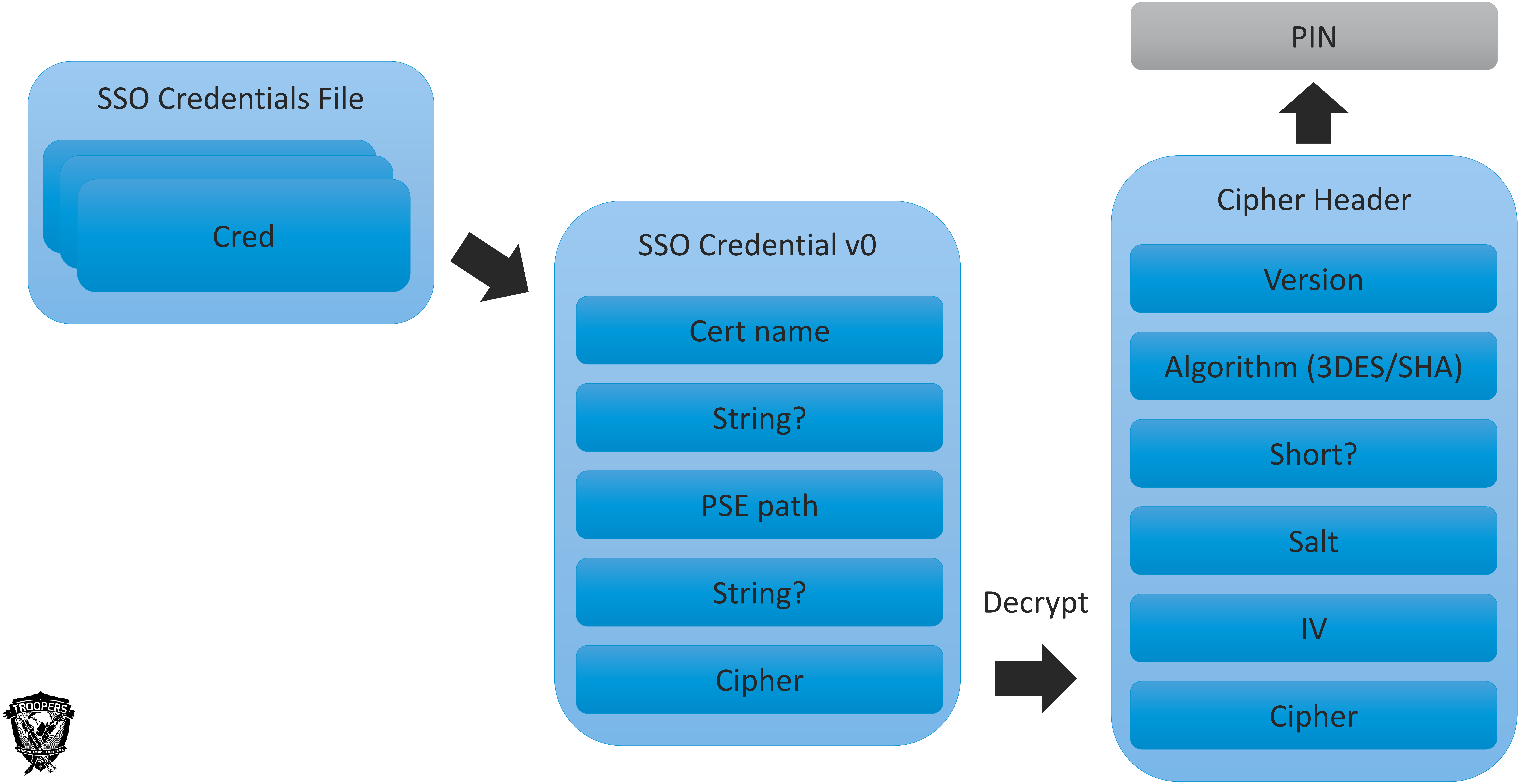
SSO Credential file format v1

Version 1 format

- Added in CCL version 8.5.15 (May 2017)
- PIN encrypted with 3DES or AES256
  - Configurable in CCL format
- Encryption key obtained from
  - **Hardcoded** key, derived using SHA256 and XORed with **hardcoded** key
- Salt and IV stored in credential file

CommonCryptoLib 8.5: Configuration Profile Parameters – SAP Note 2338952

SSO Credential file format v1



SSO Credential v1 decryption algorithm

```
Key = “<fixed key>” % username
DerivedKey = DeriveKeyFnc-SHA256+XOR(Key, FixedXORKey1,
Version, Algorithm, Short?, Salt)

AlmostPlain = 3DES(DerivedKey, IV, EncryptedPIN)

PIN = XOR(AlmostPlain, FixedXORKey2)
```

SSO Credential encryption

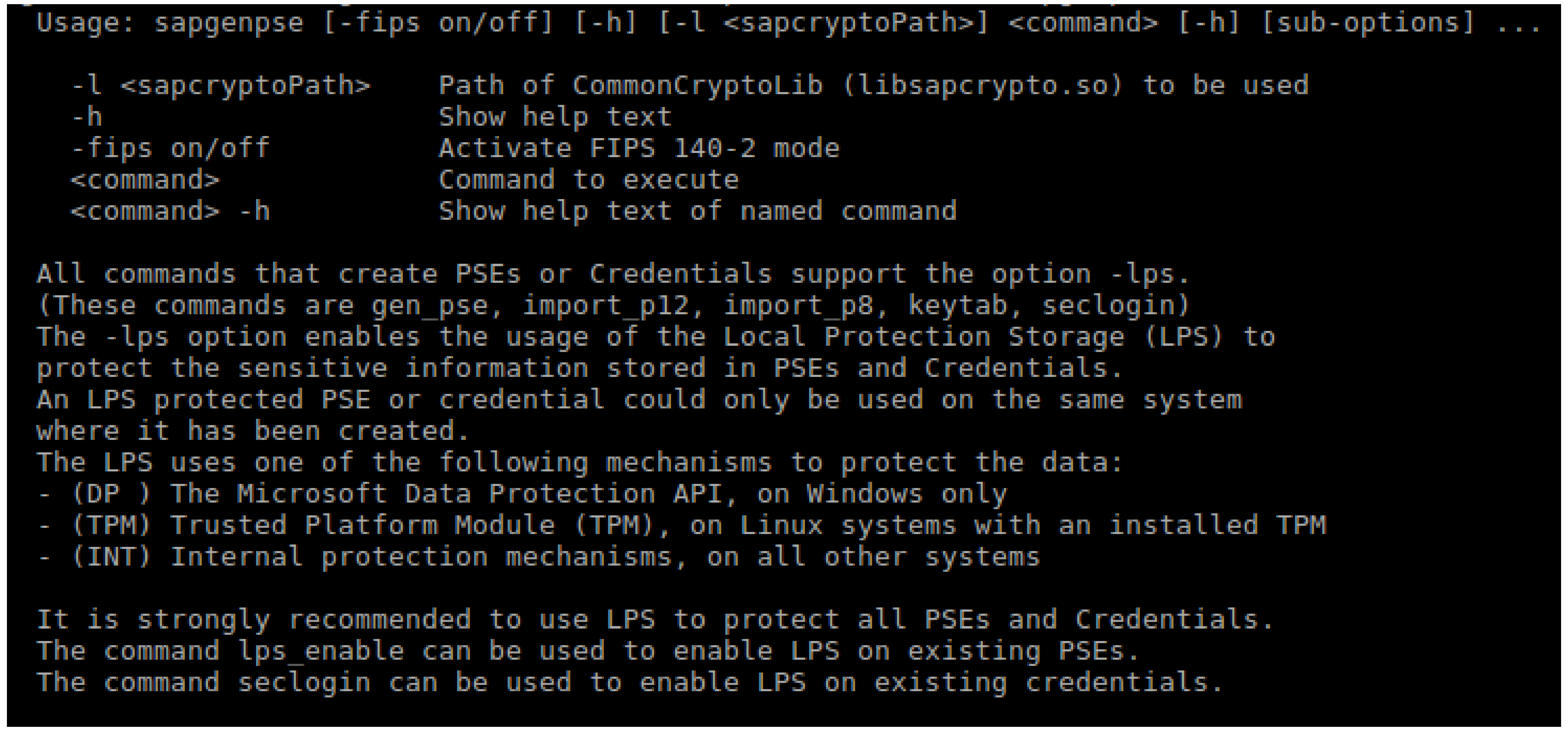
Version	Encryption mechanism	Encryption Algorithm	Encryption Key	Key Strength	CommonCrypto Lib version
0	Simple	3DES	Formatted with username from **hardcoded** key in CCL, null IV	168 bits *	< 8.5.15
	Simple (Windows only)	3DES + DP API	Formatted with username from hardcoded key in CCL, null IV, **encrypted** with DP API (AES256)	**256 bits**	< 8.5.15
1	With Header	3DES	Derived from **hardcoded** key in CCL using SHA256 + XOR key, salt and IV stored	168 bits *	>= 8.5.15 (Aug 2017)
	With Header	AES256	Derived from **hardcoded** key in CCL using SHA256 + XOR key, salt and IV stored	256 bits *	>= 8.5.15 (Aug 2017)

* Not effective key strength as key is hardcoded/fixed

Local Protection Store (LPS)

Local Protection Store (LPS)

- Advanced protection for both credentials and PSE files
- Added in SAPCRYPTOLIB
- Three working modes
  - DP API on Windows
  - TPM on Linux
  - INT or FALLBACK on Linux
    - If TPM not available



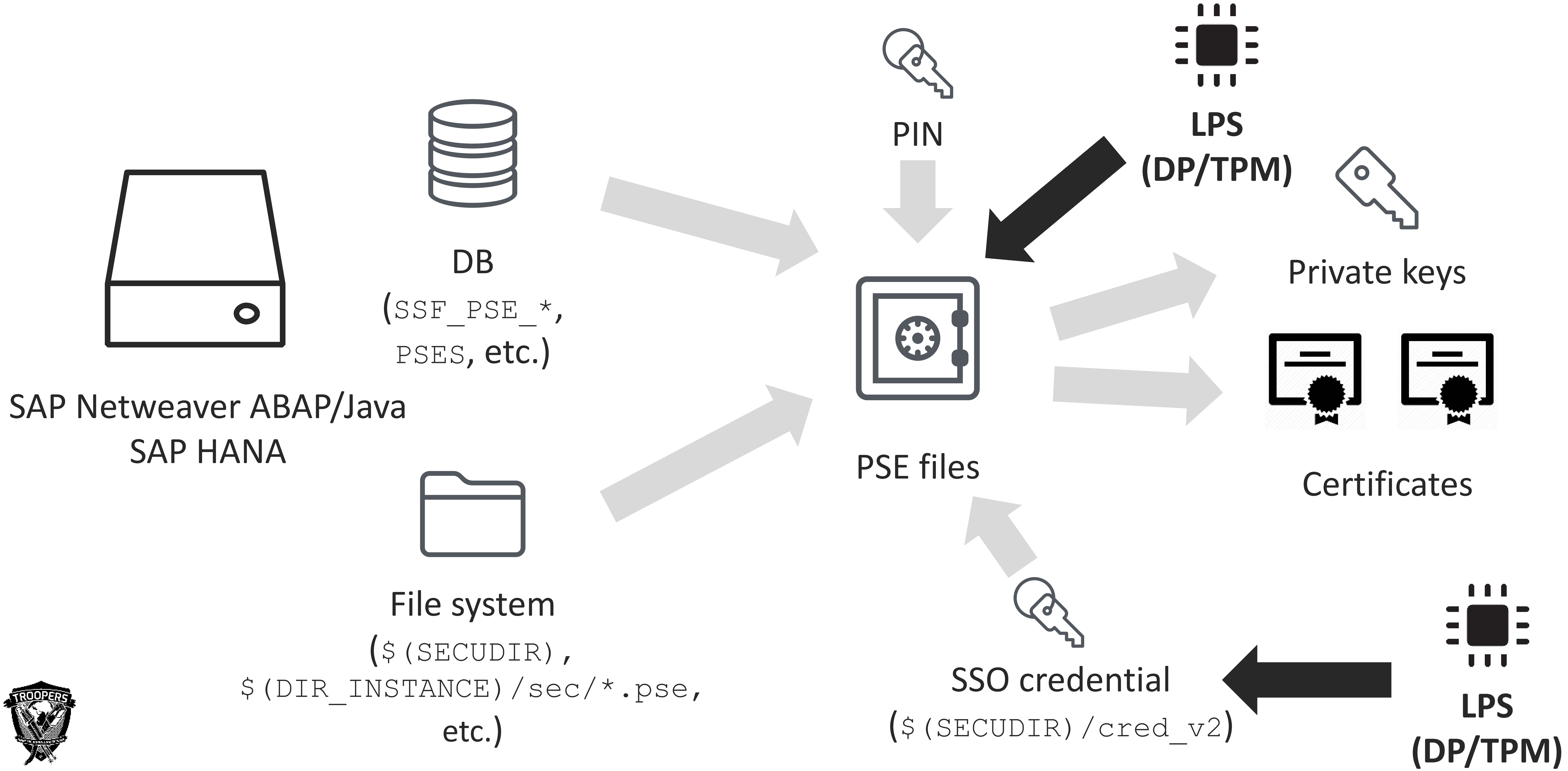
```
Usage: sapgenpse [-fips on/off] [-h] [-l <sapcryptoPath>] <command> [-h] [sub-options] ...

  -l <sapcryptoPath>    Path of CommonCryptoLib (libsapcrypto.so) to be used
  -h                    Show help text
  -fips on/off          Activate FIPS 140-2 mode
  <command>             Command to execute
  <command> -h          Show help text of named command

All commands that create PSEs or Credentials support the option -lps.
(These commands are gen_pse, import_p12, import_p8, keytab, seclogin)
The -lps option enables the usage of the Local Protection Storage (LPS) to
protect the sensitive information stored in PSEs and Credentials.
An LPS protected PSE or credential could only be used on the same system
where it has been created.
The LPS uses one of the following mechanisms to protect the data:
- (DP ) The Microsoft Data Protection API, on Windows only
- (TPM) Trusted Platform Module (TPM), on Linux systems with an installed TPM
- (INT) Internal protection mechanisms, on all other systems

It is strongly recommended to use LPS to protect all PSEs and Credentials.
The command lps_enable can be used to enable LPS on existing PSEs.
The command seclogin can be used to enable LPS on existing credentials.
```

Local Protection Store (LPS)



SAP Netweaver ABAP/Java
SAP HANA

DB
(`SSF_PSE_*`, `PSES`, etc.)

File system
(`$(SECUDIR)`, `$(DIR_INSTANCE)/sec/*.pse`, etc.)

PIN

LPS (DP/TPM)

PSE files

Private keys

Certificates

SSO credential
(`$(SECUDIR)/cred_v2`)

LPS (DP/TPM)

LPS for PSE/SSO Credential

INT/FALLBACK mode

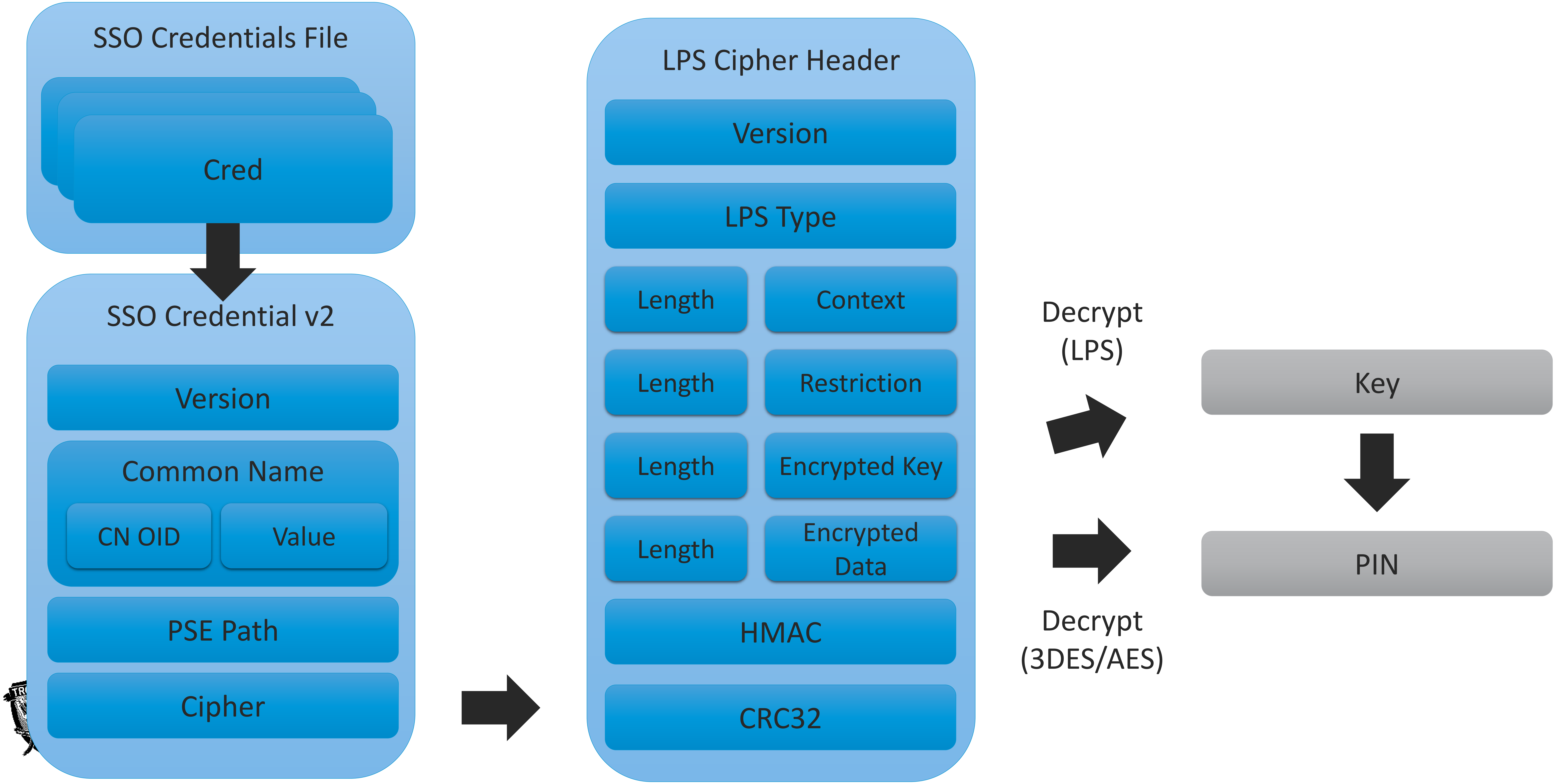
- PIN encrypted with AES256
- Encryption key obtained from
  - **Context string** encrypted with a key
  - Key derived from **hardcoded** key using SHA1 and HMAC-SHA1

LPS for PSE/SSO Credential

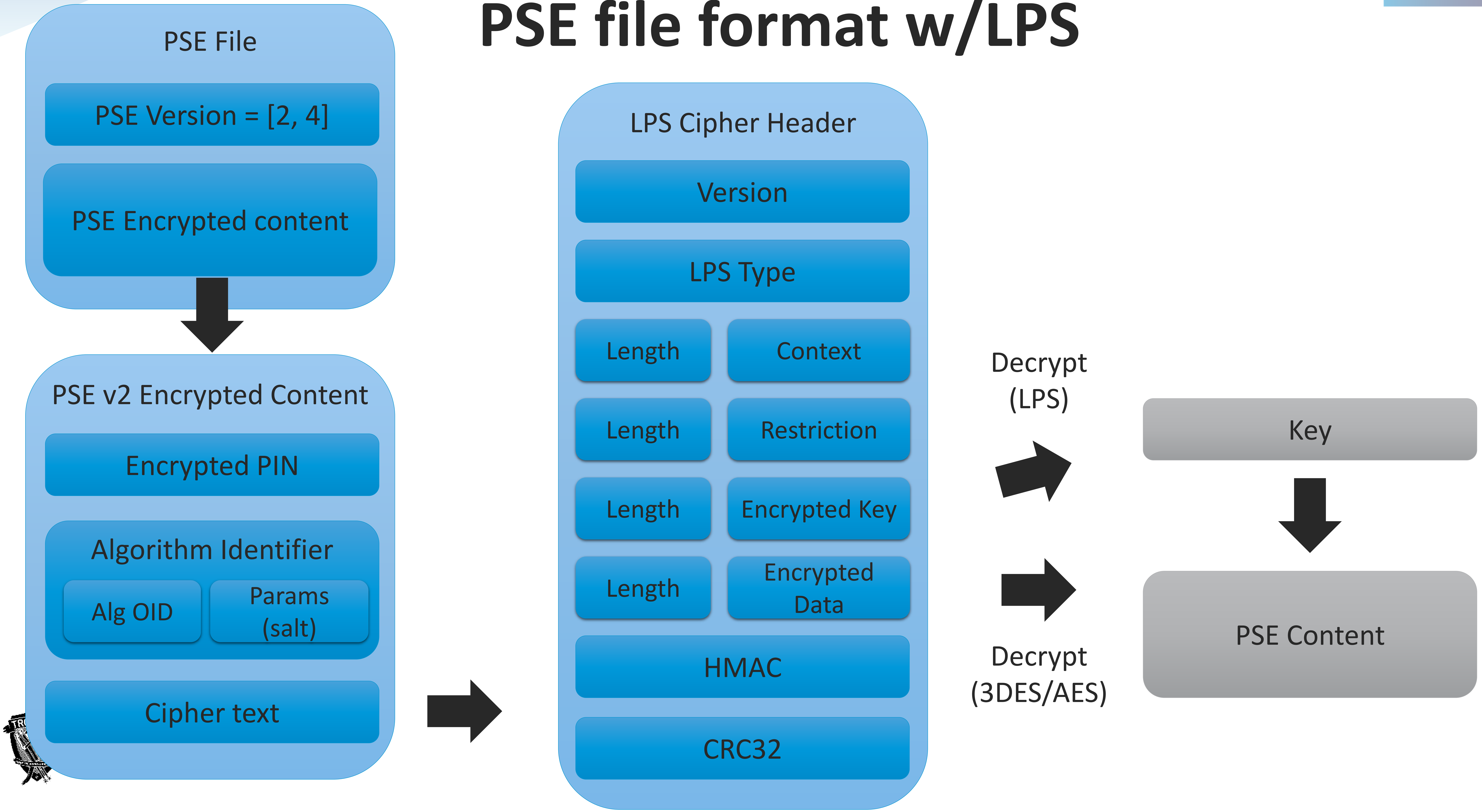
DP API/TPM mode

- PIN encrypted with AES256
- Encryption key obtained from
  - Encrypted blob in file is encrypted with **Windows DP API**
  - Encrypted blob in file is encrypted with **TPM API**
- Null IV

SSO Credential file format v2 w/LPS



PSE file format w/LPS



PSE File
- PSE Version = [2, 4]
- PSE Encrypted content

PSE v2 Encrypted Content
- Encrypted PIN
- Algorithm Identifier
  - Alg OID
  - Params (salt)
- Cipher text

LPS Cipher Header
- Version
- LPS Type
- Length | Context
- Length | Restriction
- Length | Encrypted Key
- Length | Encrypted Data
- HMAC
- CRC32

Decrypt (LPS)

Decrypt (3DES/AES)

Key

PSE Content

LPS decryption algorithm

```
Key = LPSDecrypt(Context, EncryptedKey)
IV = “00000000”

Plain = AES-256(Key, IV, EncryptedData)
```

LPS decryption algorithm

INT/FALLBACK mode LPSDecrypt

```
DerivedKey1 = SHA-1(FixedKey)
DerivedKey2 = HMAC-SHA1(DerivedKey1, Context)
IV = “00000000”

DerivedKey = AES-256(DerivedKey2[:16], IV, EncryptedKey)
```

LPS for PSE/SSO Credential encryption

Version	Encryption mechanism	Encryption Algorithm	Encryption Key	Key Strength	CommonCrypto Lib version
2	LPS - FALLBACK (Linux only)	AES256	Context string encrypted with **hardcoded** key in CCL, null IV	256 bits *	>= ???
	LPS - DP API (Windows only)	AES256	Encrypted with DP API, null IV	**256 bits**	>= ???
	LPS - TPM (Linux only)	AES256	Encrypted with TPM, null IV	**256 bits**	>= ???

* Not effective key strength as key is hardcoded/fixed

Putting everything together

Attack scenario 1

- Attacker is able to obtain PSE(s)
  - Sysadmin/BASIS not handling it properly
  - Compromising a system with `<sid>adm` or `root` permissions
  - Abusing miss-configured permissions/authorizations
  - Accessing PSE-related tables (e.g. SQL Injection)
  - …
- No SSO credentials available…

Attack scenario 1

- PSE not protected with LPS
  - Off-line crack PIN via brute force or dictionary attack

- PSE protected with LPS
  - DP API mode
    - Local access under the user account
    - If Domain account, look for recovery or backup keys in AD
  - TPM mode
    - Local access under the user account
  - Fallback mode
    - Off-line crack PIN via brute force or dictionary attack

Attack scenario 2

- Attacker is able to obtain PSE(s)
- Attacker is able to SSO credentials
  - Sysadmin/BASIS not handling it properly
  - Compromising a system with `<sid>adm` or `root` permissions
  - Abusing miss-configured permissions/authorizations
  - …

Attack scenario 2

- SSO credential not protected with LPS
  - Off-line decrypt using hardcoded keys
  - DP API mode
    - Local access under the user account
    - If Domain account, look for recovery or backup keys in AD

Attack scenario 2

- SSO credential protected with LPS
  - DP API mode
    - Local access under the user account
    - If Domain account, look for recovery or backup keys in AD
  - TPM mode
    - Local access under the user account
  - Fallback mode
    - Off-line decrypt using hardcoded keys

Practical tools

New open source tool pysap release coming!

- Support for reading and decrypting SSO Credential files
    - Version 0, 1 , 2
- Support for reading and decrypting PSE files
    - Version 2
- Support for decrypting LPS-protected SSO Credential files
    - DP API mode support (on local machine)
    - INT/FALLBACK mode support

Working on PSE cracking

- John the Ripper plugin?
- Hashcat?

https://github.com/CoreSecurity/pysap

Business Impact

Attacker with access to PSE files can

- Decrypt encrypted DB data
  - Credit cards (HCM, FI)
  - Material/product management (PLM)
  - Payroll data (HCM)
- Forge digital signatures
  - Perform bank transactions (BCM)
  - Quality management

Business Impact

Attacker with access to PSE files can

- Inspect network traffic
- Relay or intercept traffic
  - Man-in-the-middle attacks
  - Server impersonation attacks
- Modify trust relationships

Recommendations

Recommendations

- **Know** your own crypto material
  - Where/how are you using it
- **Understand** distribution mechanisms
  - ABAP PSE replication
  - HANA in-database storage
- Apply **key management** processes
  - For both PSEs and SSO credentials
  - Either stored in the filesystem or database
  - Acceptable key rotation policies

Recommendations

- Store PSEs always **encrypted**
- Use **strong** PINs
  - Randomly generated password/key
  - Passphrase
- **Enable LPS** for both PSEs and SSO credentials
  - DP API on Windows-based systems
  - Deploy TPM on Linux-based systems
  - *Avoid Fallback LPS mode*

Recommendations

- Use always **latest** CommonCryptoLib version
  - SAP note 1848999
- Configure **strong** algorithms
  - CCL profile file - SAP Note 2338952
  - PSE encryption
    - ccl/pse_encryption_iterations >= 10000
    - ccl/pse_encryption_algorithm = PBES2-AES256-SHA256
  - SSO Credentials encryption
    - ccl/credential_encryption_algorithm = AES256
- **Re-encrypt** old PSEs with newer algorithms

Conclusions

Conclusions

CRYPTO IS HARD

Just setting encryption is not enough if crypto material is not protected

KNOWLEDGE IS POWER

Understand the protection mechanisms available and the **actual** security level they provide

PRACTICAL ATTACKS

Attackers can leverage this in a practical way as post-exploitation activities

Thanks to
Troopers crew, Joris, Euge!

SECUREAUTH + CORE SECURITY

THANK YOU

mgallo@coresecurity.com
@martingalloar