

# **Personal inFo eXchange (PFX)**

**Brian Beckman, Ph.D.  
Microsoft® Corporation**

# Outline

- ◆ The Problem
- ◆ Scenario Walkthrough
- ◆ Design & Analysis
- ◆ Summary

# Freedom for Personal Info!

- ◆ User's Identity on the Net is ...
  - ✧ Private keys
  - ✧ User's Certificates
  - ✧ Certificates of recent transaction partners
  - ✧ Assorted Secrets
- ◆ Currently, user is locked into one browser & machine & app set ("Platform")
- ◆ ==> Multiple Identity Crisis!

# Personal Information Exchange (PFX) Is:

- ◆ Platform-independent transfer syntax for personal info
- ◆ Secure protocols & policies
- ◆ Convenience
  - \* Easy management of PFX-specific keys
  - \* Transport independence
  - \* Low-security password modes
  - \* Online private-key (re)generation
- ◆ Forward Compatibility
  - \* Old software will process new formats

# Two Worlds

## ◆ Online

- \* Client may be an Internet “Toaster”
  - \* No hard drive
  - \* Maybe even no floppy
  - \* No smart card reader in next year or two
- \* Secrets have to be transported on the Net!

## ◆ Offline

- \* For platforms with floppy drives
- \* Transport secrets on floppy
- \* User physically protects the secrets
- \* Reduces encryption strength requirements

# A Third World

## ◆ Secure Hardware Tokens

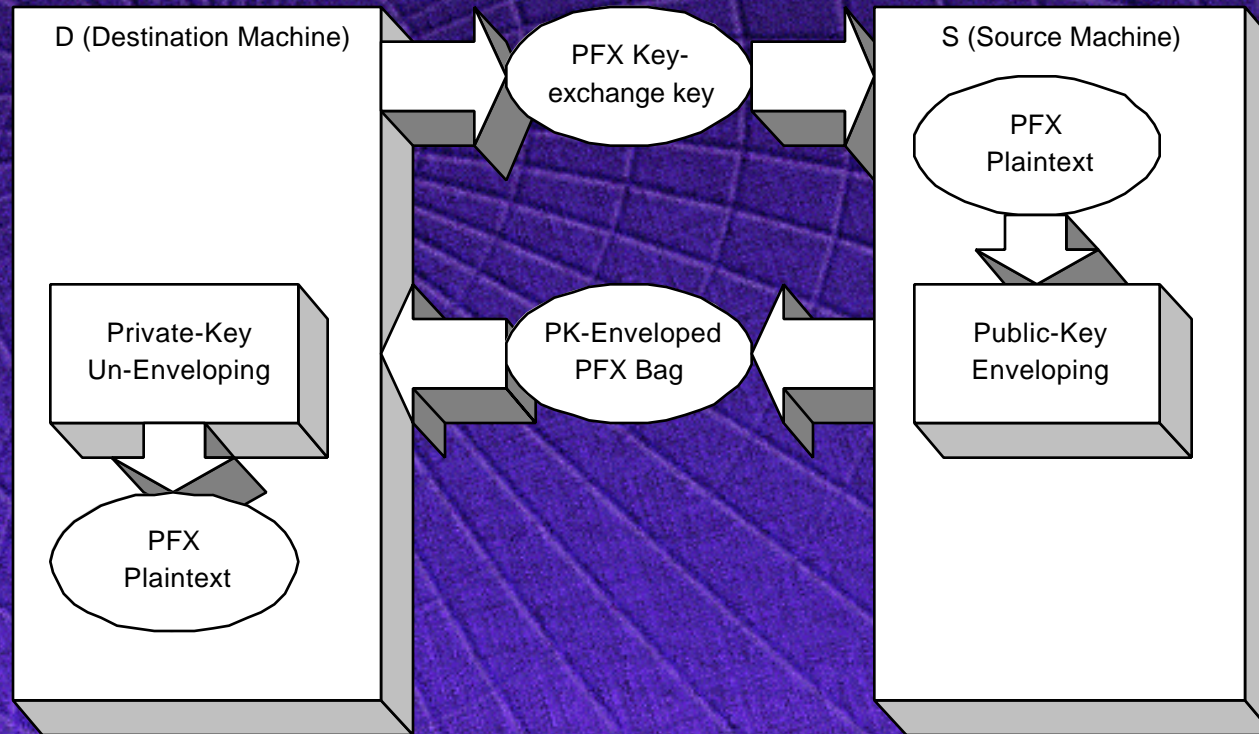
- \* Smart Cards contain private keys
- \* Fat Cards hold other private data

## ◆ When?

- \* Not soon enough
- \* We can do reasonably well now in SW
  - \* Be compatible with hardware later
- \* Hardware will need key recovery, anyway
  - \* PFX is a candidate format & protocol

# Scenario: Public-Key Privacy Mode

PFX Exchange from S (Source Machine) to D (Destination Machine) in Public-Key Privacy Mode



# Scenario Specifics

- ◆ User's client auth private key "V" is safely transported from one machine to another
- ◆ Preconditions:
  - \* Two machines: S — source and D — destination
  - \* "V" exists on S (Proof: SSL client auth succeeds)
  - \* "V" does not exist on D (Proof: SSL client auth fails)
  - \* S has D's (trustified) public key-exchange key [PKEK]
    - \* Note: PKEK is D's key, not the user's key

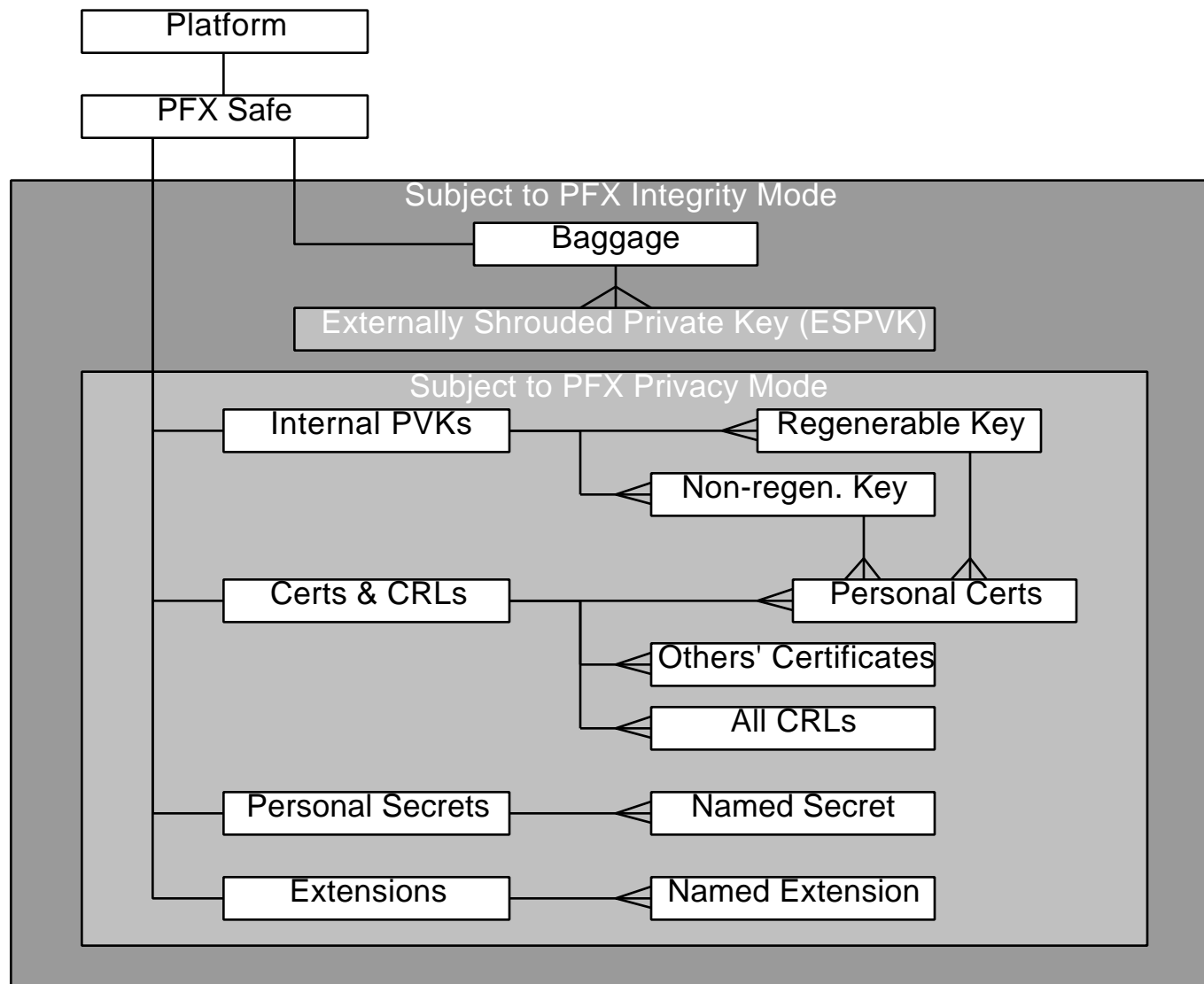
# Scenario: Steps

- ◆ 1: Transport PKEK [from D] to S
- ◆ 2: Create, encrypt and sign a PFX bag containing “V”
  - ✧ Signed with S’s signature key for integrity
  - ✧ Enveloped to D with PKEK
- ◆ 3: Transport PFX bag to D
- ◆ 4: Verify integrity and decrypt PFX bag on D
- ◆ 5: Install “V” on D



# Design and Analysis

# Data Model



# PFX Modes

## ◆ Privacy Modes

- ✧ Public-key Privacy Mode
- ✧ Password Privacy Mode

## ◆ Integrity Modes

- ✧ Public-key Integrity Mode
- ✧ Password Integrity Mode

## ◆ Private-key (re)generation

- ✧ Works only for private keys
- ✧ Requires a well-known, 'magic-mirror' server

# PFX Privacy Modes

## ◆ Public-Key Mode

- \* Each platform has a key-exch key pair
- \* Encrypt PFX bag from source to dest. platform
- \* Best available security
- \* Email your PFX bag (or store on server)

## ◆ Password Mode

- \* Don't have Key Exchange Key of dest. platform?
- \* Derive symmetric key from username & password
- \* Don't email if including private keys

# PFX Integrity Modes

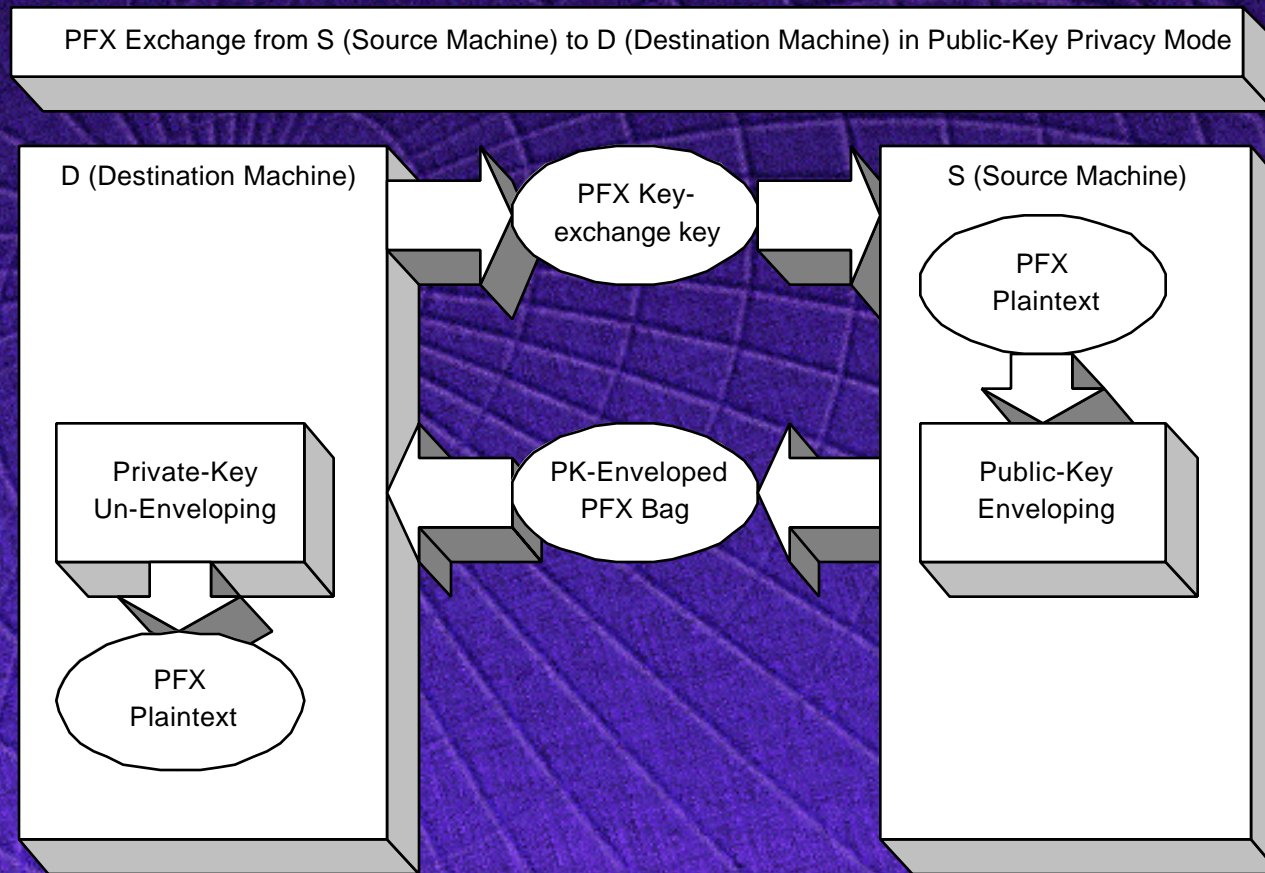
- ◆ Prevents Adversary from tampering
- ◆ Public-Key Mode
  - \* Each platform has a trusted signature key pair
  - \* Hash and sign PFX bag on source platform
- ◆ Password Mode
  - \* Don't have Signature Key on source platform?
  - \* Derive MAC key from username & password
  - \* Hash bag with MAC key

# PFX Modes (cont.)

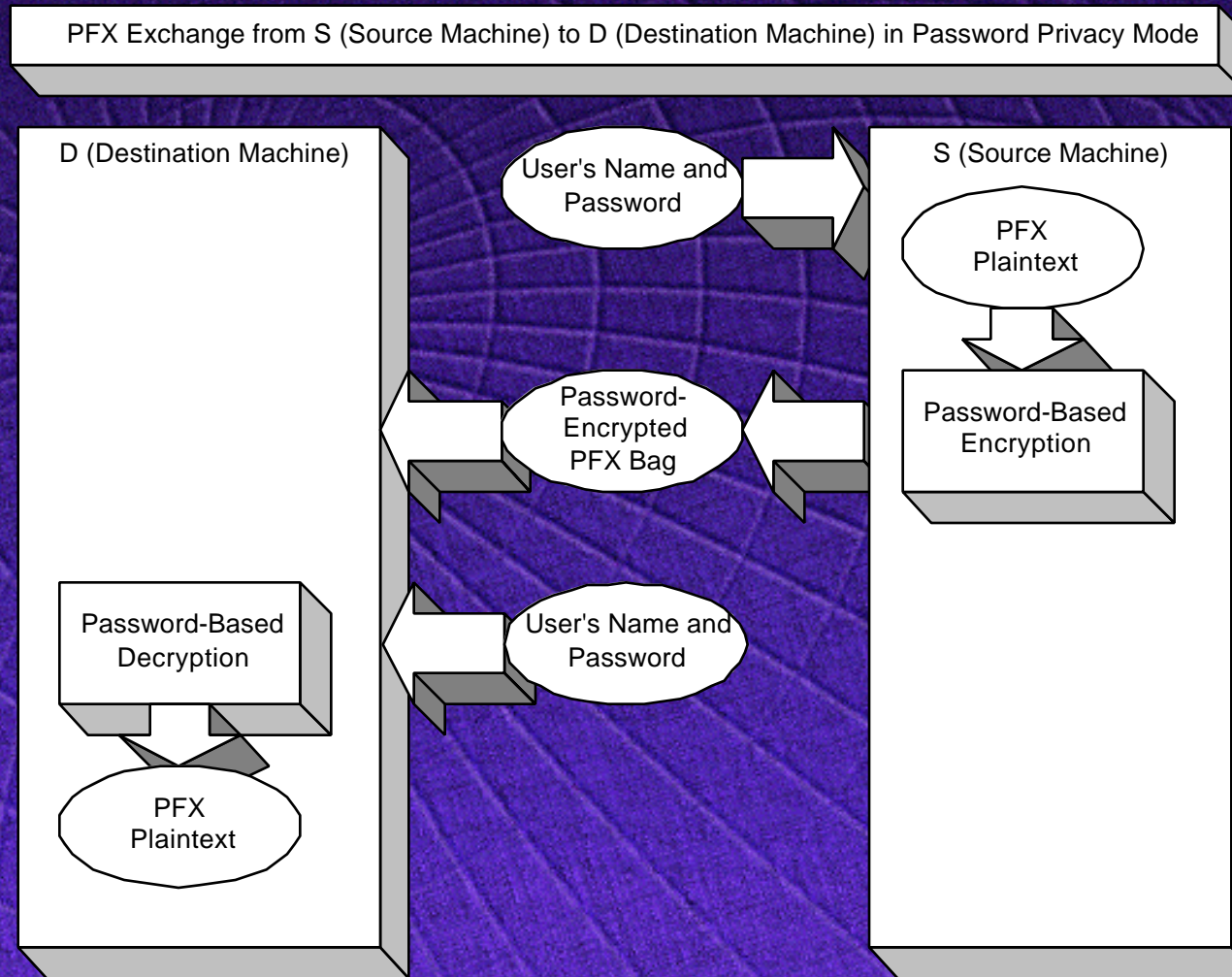
## ◆ Private Key (re)Generation

- \* Create same private keys on multiple platforms
- \* Must gen keys every time, including first
- \* Online 'magic-mirror' server impedes dictionary attacks
- \* Server does not learn user's secrets
- \* Server injects entropy from eavesdropper's perspective
- \* No server entropy from perspective of dictionary attacker!
- \* Essential part of PFX's Online World
  - \* Since online transport of password-protected private keys is strongly discouraged

# Public-Key Privacy Mode

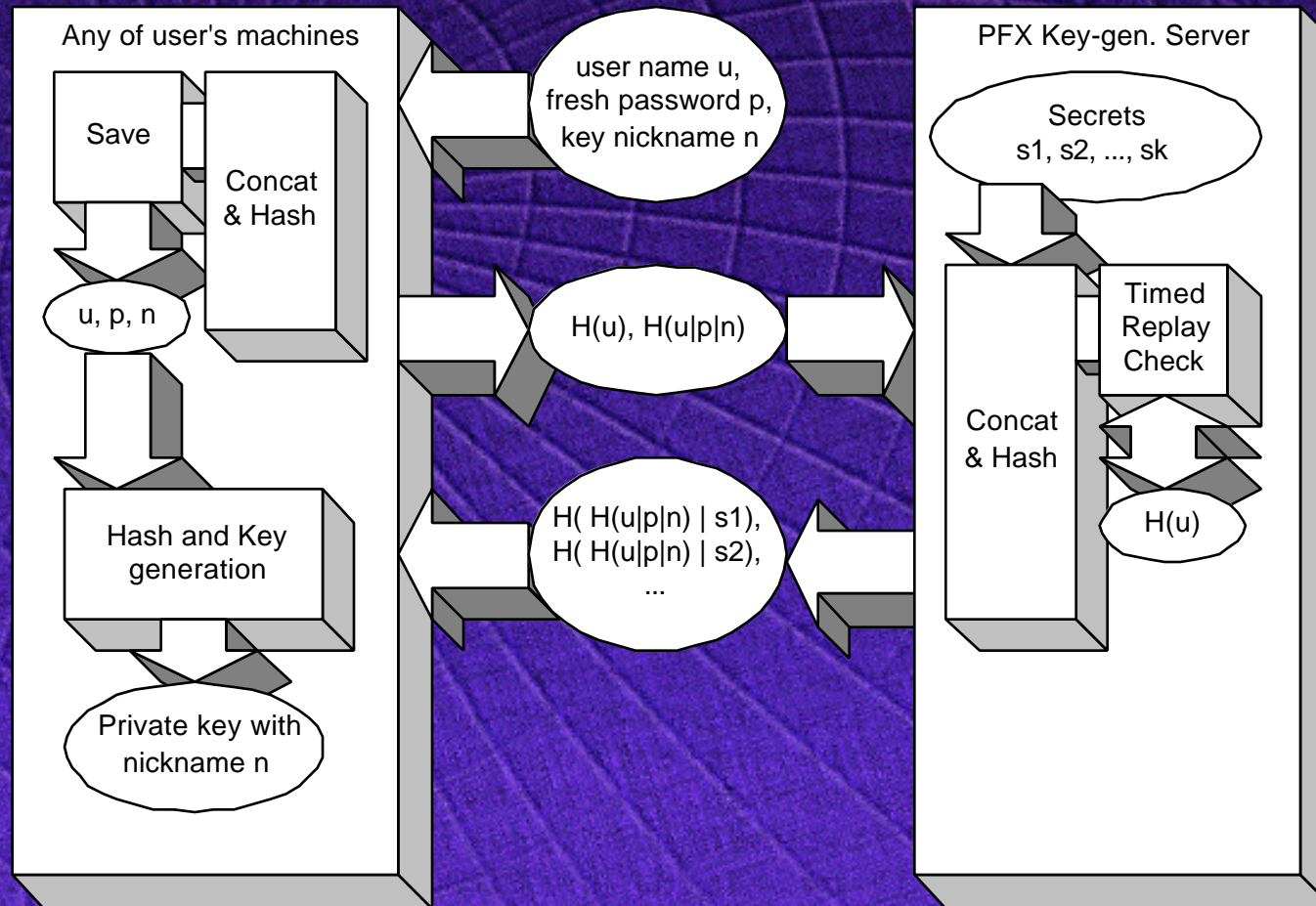


# Password Privacy Mode



# Private-Key (re)Generation

PFX Key (re)generation using the 'Magic-Mirror' Server



# PFX Data Format

- ◆ Integrity and Privacy wrappers are PKCS #7 *ContentInfo*
- ◆ Outer wrapper, *ContentType* options:
  - \* *SignedData* = Public-key Integrity Mode
  - \* *data* = Password Integrity Mode
- ◆ Inner Wrapper *ContentType* options:
  - \* *EnvelopedData* = Public-key Privacy Mode
  - \* *EncryptedData* = Password Privacy Mode

# Note on wrapping order

- ◆ Integrity protects baggage, version, and privacy wrapper
- ◆ NOT to be interpreted as a ‘signature of encrypted data’
  - \* Normally not allowed in protocols because of contract law
  - \* ‘Signature’ here is with a platform key, not a human signature key
  - \* There is no reason for a human to inspect the signature preimage

# Wrapper details

```
PFX ::= SEQUENCE {  
    safeMAC      DigestInfo  OPTIONAL  
    macSalt      BIT STRING  OPTIONAL,  
    authSafe     ContentInfo -- signedData or data  
}
```

```
AuthenticatedSafe ::= SEQUENCE {  
    version      Version      DEFAULT v1,  
    transportMode TransportMode DEFAULT off-lineTransport,  
  
    privacySalt  BIT STRING   OPTIONAL,  
    baggage      Baggage      OPTIONAL, -- No encryption  
    safe         ContentInfo -- encryptedData or envelopedData  
}
```

# PFX Data Format (cont.)

## ◆ Inner contents mimic the data model

SafeContents ::= SET OF SafeBag

SafeBag ::= SEQUENCE {  
    safeBagType    OID,  
    safeBagContent [0] EXPLICIT ANY DEFINED BY safeBagType  
}

pkcs-12BagIds OID ::= {pkcs-12 3}

keyBagID        OID ::= {pkcs-12BagIds 1}

certAndCRLBagID OID ::= {pkcs-12BagIds 2}

secretBagId     OID ::= {pkcs-12BagIds 3}

# Private Keys

KeyBag ::= SET OF PrivateKey

PrivateKey ::= SEQUENCE {  
    pvkData    PVKSupportingData,  
    pkcs8data  PrivateKeyInfo -- import from PKCS #8  
}

PVKSupportingData ::= SEQUENCE {  
    assocCerts  SET OF Thumbprints,  
    regenerable  BOOLEAN DEFAULT FALSE,  
    nickname    BMPString, -- Straight UNICODE  
    pvkAdditional PvkAdditional OPTIONAL  
}

Thumbprint ::= DetachedDigest

# APIs

- ◆ **Proposal includes APIs**

PfxExportCreate

PfxExportGetPDU

PfxImportOpen

PfxImportSave

PfxEnumItems

PfxItemExists

PfxRemoveItem

PfxCloseHandle

PfxGetLastError

- ◆ **Anyone can write PFX import / export utilities**
- ◆ **Cross-platform: NOT just Win32**
- ◆ **See the docs...**

# Summary

- ◆ PFX is an interoperability protocol and data format to move a user's private information safely from one platform to another
- ◆ It's been submitted to the W3C and to RSA as an open standard
- ◆ There's been significant security review
  - \* Implement now!
  - \* There's enough to go on to get started
- ◆ Read the spec and comment ([cryptoapi@listserv.msn.com](mailto:cryptoapi@listserv.msn.com))



**Questions?**

# *Microsoft*<sup>®</sup>

