PKCS #5 v2.0:
Password-Based Cryptography Standard

Burt Kaliski, RSA Laboratories
PKCS Workshop, 29 September 1999

Outline

- Background
- PKCS #5 v1.5
- PKCS #5 v2.0
- Generating salt
- Possible future work

Background

- Cryptography with a password ...
  - encryption
  - message authentication
  - identification, key establishment
- … has some peculiar problems:
  - passwords are not conventional keys
  - nor are they very “random”

RSA SECURITY

General Model

- **Password-based key derivation:**

  $$key = \mathrm{PBKDF}\,(password,\ salt,\ count)$$

- ***salt*** **prevents dictionary attack**
- ***count*** **complicates search**
- ***key*** **is applied to conventional cryptosystem**

PKCS #5 v1.5

- **Password-Based Encryption Standard**
  - published November 1993
- **Two encryption schemes:**
  - MD2 with DES-CBC
  - MD5 with DES-CBC

PKCS #5 v1.5 Encryption Scheme

Encryption operation

1. Generate salt S

2. Hash with password to derive key, IV:

$$K \,||\, IV = \text{Hash}^{count}(P \,||\, S)$$

3. Pad message and encrypt:

$$EM = M \,||\, pad$$
$$C = \text{DES-CBC}\,(K, IV, EM)$$

- S, *count*, C sent to recipient

Decryption is similar

RSA SECURITY

Limitations of v1.5 Scheme

- **Algorithm restrictions:**
  - only two hash functions
  - only one underlying encryption scheme
    - includes padding, assumes CBC mode
- **Theoretical deficiencies:**
  - no formal model or security proof for KDF
  - construction has “entropy bottleneck”
- **Fixed maximum length for keys**

Enhancements Before v2.0

- **RSA Data Security extensions:**
  - SHA-1 hash function
  - RC2-CBC encryption
- **PKCS #12 password-based schemes**

PKCS Workshops ’97, ’98

- Discussion of PKCS #5 improvements, proposed draft
- Conclusions:
  1. New key derivation function to be specified
  2. Modular encryption, message authentication schemes
     - parameters for underlying scheme (e.g., IV) managed separately

RSA SECURITY

PKCS #5 v2.0

- Password-Based *Cryptography* Standard
  - published March 1999
- Several techniques:
  - extended v1.5 and new KDF
  - extended v1.5 and new modular encryption schemes
  - new modular message authentication scheme
- New schemes are recommended for new applications

New Key Derivation Function

PBKDF2 (P, S, c, $dkLen$)

1. Compute blocks $T_1,\ldots,T_l$ by iterated construction:

$$U_1 = G(P, S \,||\, \mathrm{Int}(i)),\ U_2 = G(P, U_1),\ \ldots,\ U_c = G(P, U_{c-1})$$

$$T_i = U_1 \oplus U_2 \oplus \cdots \oplus U_c$$

2. Output first $dkLen$ octets of $T_1 \,||\, \cdots \,||\, T_l$

G is underlying pseudorandom function

RSA SECURITY

Motivation for PBKDF2

- "Belt-and-suspenders" approach:
  - U_i values are computed recursively to remove a degree of parallelism
    - different than PKCS '98 proposal, which computed them independently as $U_j = G\ (P,\ S\ ||\ \mathrm{Int}\ (i)\ ||\ \mathrm{Int}\ (j))$
  - XOR reduces concerns about the recursion degenerating into a small set of values
- (Potentially) provably secure under reasonable assumptions on pseudorandom function G
- Variable length through varying i

RSA SECURITY

New Encryption Scheme

Encryption operation

1. Select salt S, iteration count c, key length $dkLen$
2. Apply KDF to derive key

$$DK = \mathrm{KDF}\ (P, S, c, dkLen)$$

3. Apply underlying encryption scheme

$$C = \mathrm{Enc}_{DK}\ (M)$$

   – parameters such as IV selected as part of underlying scheme

Decryption is similar

Message Authentication Scheme

MAC generation operation

1. Select salt S, iteration count c, key length $dkLen$

2. Apply KDF to derive key

$$DK = \text{KDF}\,(P, S, c, dkLen)$$

3. Apply underlying message authentication scheme

$$T = \text{MAC}_{DK}\,(M)$$

MAC verification is similar

RSA SECURITY

Addressing the Limitations

- **Algorithm restrictions:**
  - arbitrary (iterated) pseudorandom function
  - arbitrary underlying encryption scheme
- **Theoretical deficiencies:**
  - formal model / (potential) security proof for KDF
  - construction still has “entropy bottleneck”
    - but can support wider hash function
    - not a practical problem for passwords
- **Large maximum length for keys**

Supporting Techniques

- **Pseudorandom functions:**
  - HMAC-SHA-1 where message is index
- **Encryption schemes:**
  - DES, DES-EDE3, RC2, RC5
    - all in CBC mode with PKCS #5 v1.5 padding
  - DES-EDE2, DESX, RC4 could potentially be added
- **Message authentication schemes:**
  - HMAC-SHA-1

RSA SECURITY

ASN.1 Syntax

- Key derivation functions
  - only PBKDF2
- Encryption schemes
  - PBES1 and PBES2
- Message authentication scheme (and pseudorandom function)
  - PBMAC1

PBKDF2

- **Generic OID:**
  - id-pbkdf2 ::= pkcs-5.12
- **Parameters:**
  - PBKDF2-params ::= SEQUENCE {
    salt CHOICE {
      specified OCTET STRING,
      otherSource AlgID {{PBKDF2-SaltSources}} },
    iterationCount INTEGER (1..MAX),
    keyLength INTEGER (1..MAX) OPTIONAL,
    prf AlgID {{PBKDF2-PRFs}}
      DEFAULT algid-hmacWithSHA1 }

RSA SECURITY

PBES1

- Specific OIDs as in v1.5:
  - pbeWithMD2AndDES-CBC ::= pkcs-5.1
  - pbeWithMD5AndDES-CBC ::= pkcs-5.3
  - …
  - pbeWithSHA1AndRC2-CBC ::= pkcs-5.11
- Parameters:
  - PBEParameter ::= SEQUENCE {
    salt OCTET STRING SIZE (8),
    iterationCount INTEGER }

PBES2

- Generic OID:
  - id-pbes2 ::= pkcs-5.13
- Parameters:
  - PBES2-params ::= SEQUENCE {
    kdf AlgID {{PBES2-KDFs}},
    enc AlgID {{PBES2-Encs}} }

PBMAC1

- Generic OID:
  - id-pbmac1 ::= pkcs-5.14
- Parameters:
  - PBMAC1-params ::= SEQUENCE {
    kdf AlgID {{PBMAC1-KDFs}},
    mac AlgID {{PBMAC1-MACs}} }

Generating Salt

- Primary purpose of salt is to increase difficulty of precomputation attacks
- Secondary purpose is to separate keys generated at different times
- A random salt assures the one who generated it that these goals are achieved, but not necessarily the one who receives it
  - i.e., the party decrypting a ciphertext or verifying a MAC is not assured that separate keys were employed

RSA SECURITY

Exploiting Ambiguity

- Suppose a password is employed for two algorithms with different key lengths, and the salt does not distinguish between them
- Then for a given salt, the key for one algorithm will be a prefix of the key for the other
- Suppose also that an opponent can solve for the shorter key by a chosen ciphertext attack
- Then the opponent can also solve for the longer key by guessing the rest of it: “divide-and-conquer”
- Similar concerns for other kinds of interaction

RSA SECURITY

Salt Recommendations

- If interactions are not a concern (e.g., password is always employed with the same algorithm), then a random salt is sufficient
  - at least 64 bits recommended
- If they are, then the salt should also contain some structure, e.g., an algorithm identifier and/or a sequence number that can be checked by the party receiving the key

Possible Future Work

- **Structure for salt value**
  - basically, key derivation parameters for KDF
- **"Pepper" variants where part of salt is secret**
  - several references in literature
- **Public-key password-based techniques**
  - password-based entity authentication and key establishment
  - password-based private-key downloading