Cryptography for Smalltalkers

Martin Kobetic
Cincom Smalltalk Development

ESUG 2004

To get the VW presentation

- in the open repository in a bundle called Presentations-MK, class ESUG04Crypto. To load it and open it up in a recent version of VW just run this:
- | profile |profile := Store.ConnectionProfile newname: 'open repository';driverClassName: 'PostgreSQLEXDIConnection'; environment: 'store.cincomsmalltalk.com:5432_store_public'; userName: 'guest'; password:'guest';yourself.

- Store.RepositoryManager addRepository: profile;Store.DbRegistry connectTo: profile.Store.Bundle newestVersionWithName: 'Presentations-MK') loadSrc.ESUG04Crypto open

Cryptographic Objectives

- confidentiality
  - encryption
- integrity
  - message authentication codes (MAC)
- authentication
  - signatures

Encryption

- $E(P) = C$ & $D(C) = P$
- symmetric (secret) key ciphers
  - $E_K(P) = C$ & $D_K(C) = P$
- asymmetric (public) key ciphers
  - $E_{K1}(P) = C$ & $D_{K2}(C) = P$
- one-time pad

Secret Key Ciphers

- bulk data encryption
- built from simple, fast operations (xor, shift, x + y mod n, ...)
- two fundamental classes
  - stream ciphers (RC4)
  - block ciphers (DES/AES)

Secret Key Ciphers

```
key := ‘secret key’ asByteArray.
alice := ARC4 key: key.
msg := ‘Hello’ asByteArrayEncoding: #utf_8.
msg := alice encrypt: ‘Hello’
msg asString.

bob := ARC4 key: key.
bob decryptInPlace: msg from: 1 to: 4.
msg asStringEncoding: #utf_8.
```

Stream Ciphers

- time-varying transformation on individual plain-text digits

  key-stream generator: $k_1, k_2, k_3, \ldots.$
  State S, NextState(S), Output(S)

  E: $c_i = p_i \text{ xor } k_i$
  D: $p_i = c_i \text{ xor } k_i$

- Pike, A5, RC4, SEAL
- key reuse is catastrophic!
  (a xor k) xor (b xor k) = a xor b

RC4 (1992)

- leaked trade secret of RSA Security (1994)
- 256 byte S-Box; 2 counters i=j=0

S-Box initialization:

```
$S = 0, ..., 255$
$K$ = 256B of replicated key
for $i=0$ to 255:
  $j = (j + S_i + K_i) \bmod 256$
  swap $S_i$ and $S_j$
```

next key-stream byte:

```
$i = (i + 1) \bmod 256$
$j = (j+S_i) \bmod 256$
swap $S_i$ and $S_j$
$t = (S_i + S_j) \bmod 256$
$K = S_t$
```

RC4

```
alice := ARC4 key: key.
msg := alice encrypt: 'Hello' asByteArray.
msg asHexString.

bob := ARC4 key: key.
(bob decrypt: msg) asString
```

Block Ciphers

- fixed transformation on blocks of plaintext (e.g 64, 128 bits)
- basic transformation applied in rounds
- DES, IDEA, CAST, Blowfish, RC2, RC5

DES (1977)

- csrc.nist.gov: FIPS PUB 46 (1977)
- FIPS PUB 46-3 (1999)
  - triple DES still approved
  - single DES legacy systems only
- 64 bit block size
- 56 bit key (64 bits with parity)
- 16 rounds using 48 bit subkeys

Block Ciphers - Padding

```
key := ‘secret8B’ asByteArray.
alice := DES key: key.
alice encrypt: ‘Hello World!’ asByteArray.

alice := BlockPadding on: DES new.
alice setKey: key.
(alice encrypt: ‘Hello World!’ asByteArray) asString.
```

Block Ciphers - Padding

- must be reversible
- pad with bits “100…0”
- pad with padding size (1-8)
  - aka PKCS#5 padding
- ciphertext stealing
  - different for different modes (ECB, CBC)
- some modes don’t need padding

Block Ciphers - ECB

- electronic codebook mode
- $C_i = E_k(P_i)$
- $P_i = D_k(C_i)$
- don’t use !

Block Ciphers - CBC

- cipher block chaining mode
- $C_i = E_k(P_i \text{ xor } C_{i-1})$
- $P_i = C_{i-1} \text{ xor } D_k(C_i)$
- initialization vector (IV)
  - isn’t secret but unique, random
  - timestamp, nonce, random nr

Block Ciphers - CBC

```
alice := CipherBlockChaining
            on: DES new
            iv: ‘nonce 8B’ asByteArray.
alice setKey: ‘secret8B’ asByteArray.
msg  := ‘a block a block ’ asByteArray.
msg := alice encrypt: msg.
msg asString
```

Block Ciphers - CBC

```
alice := DES newBP_CBC.
alice setKey: 'secret8B' asByteArray.
alice setIV: 'nonce 8B' asByteArray.
msg := 'Hello World!' asByteArray.
msg := alice encrypt: msg.
msg asString.
```

Block Ciphers - OFB

- output feedback mode
- $S_i = E_k(S_{i-1})$
- $C_i = P_i \text{ xor } S_i$
- $P_i = C_i \text{ xor } S_i$
- like synchronous stream cipher

```
(OutputFeeback on: DES new)
      setKey: ‘secret8B’ asByteArray;
      setIV: ‘nonce 8B’ asByteArray
```

Block Ciphers - CTR

- counter mode
- $S_i := E_k(\text{Nonce} \,||\, i)$
- $C_i = P_i \text{ xor } S_i$
- $P_i = C_i \text{ xor } S_i$
- OFB variant

Block Ciphers - CFB

- cipher feedback mode
- $C_i = P_i \text{ xor } E_k(C_{i-1})$
- $P_i = C_i \text{ xor } E_k(C_{i-1})$
- like self-synchronizing stream cipher

```
(CipherFeeback on: DES new)
    setKey: ‘secret8B’ asByteArray;
    setIV: ‘nonce 8B’ asByteArray
```

Block Ciphers - Mixing

- interleaving
  - parallelizing “chained” modes
- multiple encryption with single cipher
  - double encryption – no good
  - 3EDE (inner/outer CBC)
- cascading different ciphers

Block Ciphers - Mixing

```
des3 := TrippleEDEOuterCBC
            first: DES new
            second: DES new
            third: DES new.
des3 := DES new3EDE_CBC.
des3 setKey: ’24bytes for 3 keys’ asByteArray.
des3 setIV: ‘nonce 8B’ asByteArray.
```

AES (2001)

- NIST FIPS PUB 197 (2001) - Rijndael
- 15 submissions (1998)
- 5 finalists: MARS, Serpent, Twofish, RC6
- modes: ECB, CBC, CFB, OFB, CTR
- block size 128 bits
- key sizes 128, 192, 256 bits
- 10, 12, 14 rounds

Blowfish (1993)

- http://www.counterpane.com/blowfish.html
- block size 64-bits
- variable key size 32-448 bits
- not patented, royalty-free
- 2 parts: key expansion & data encryption
- 16 rounds, key dependent S-Boxes

Books

- Anderson: Security Engineering
- Ferguson, Schneier: Practical Cryptography
- Kahn: The Codebreakers, …
- Menezes, van Oorschot, Vanstone: Handbook of Applied Cryptography
- Schneier, B: Applied Cryptography