Implementation pitfalls

- We learned about various cryptographic primitives and the provable security approach, saw many secure constructions.
- You are almost ready to employ this knowledge in practice.
- Let us review some common mistakes one needs to be aware of and avoid when implementing cryptographic protocols.

1

Always remember to

- Use widely accepted and believed to be secure building blocks (e.g. AES).
- Use provably secure (under reasonable assumptions) constructions (e.g. $CBC).
- Do not assume that the schemes provide security properties other than what is proven about them (e.g. encryption does not provide authenticity).
- Realize that the use of a provably secure scheme does not guarantee that the entire system will be secure.
- Make sure that you implement exactly the scheme that was proven secure.

2

Not using the right primitives

- ATM-based passive optical networks commonly use a block cipher called CHURN. It's key size is 8 bits and it's block size is 4 bits!

Using the constructs without security proofs

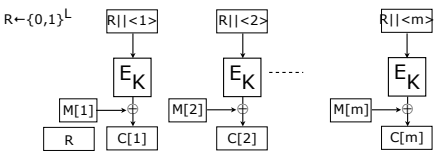
- The use of the ECB mode and the Plain RSA encryption is still very common.

3

Not considering the security bounds

Consider the encryption algorithm of a scheme CTRS$[L]

Let $E:\{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher.



One can prove that for every A making q queries there exists B s.t.

$$\mathbf{Adv}^{\text{ind-cpa}}_{CTRS\$[L]}(A) \leq \mathbf{Adv}^{\text{prf}}_{E}(B) + \frac{q^2}{2^{L+1}}$$

Is CTRS$ secure?

4

$$\mathbf{Adv}^{\text{ind-cpa}}_{CTRS\$[L]}(A) \leq \mathbf{Adv}^{\text{prf}}_{E}(B) + \frac{q^2}{2^{L+1}}$$

- WEP protocol for IEEE 802.11 wireless networks uses a scheme like CTRS$ with L=24,40,64 or 80.
- Assume L=24 and q=4096. Then the last term becomes 1/2 and no security is guaranteed by the bound!

5

Not using the right tool

- It is tempting to believe that encryption provide some authenticity.
- The first versions of the SSH protocol, IPsec specification and the WEP protocol did not use message authentication codes, and thus were subject to certain attacks.

Not implementing exactly the provable-secure schemes

- A slightest tweak to a provably-secure scheme can make it insecure
- Diebold voting machines encrypted the votes with CBC$, but used all-zero string as an IV.
- Microsoft Word and Excel used CBCS$, but did not pick a new random R each time.

6

Random numbers

- It is usually straightforward to implement the pseudo-code descriptions in C or Java.
- However, how do you implement commands like $K \xleftarrow{\$} \{0,1\}^k$?
- The C offers a built-in random number generator, that works roughly as this

```
                    32-bit number
procedure srand(seed)  | function rand()
    state = seed;      |     state = ((state * 1103515245) + 12345)
                       |         mod 2147483648;
                       |     return state
                                          2^31
```

7

- So one can implement $K \xleftarrow{\$} \{0,1\}^k$ as follows

```
algorithm K          | function keygen()
    K <-$ {0,1}^128  |     key[0] = rand(); key[1] = rand();
    return K         |     key[2] = rand(); key[3] = rand();
                     |     return key
```

- But looking at how rand() works we notice that

$$\begin{aligned}
\texttt{key[1]} &= ((\texttt{key[0]} \cdot 1103515245) + 12345) \bmod 2^{31} \\
\texttt{key[2]} &= ((((\texttt{key[0]} \cdot 1103515245) + 12345) \cdot 1103515245) + \\
&\quad 12345) \bmod 2^{31} \\
\texttt{key[3]} &= ((((((\texttt{key[0]} \cdot 1103515245) + 12345) \cdot 1103515245) + \\
&\quad 12345) \cdot 1103515245) + 12345) \bmod 2^{31}
\end{aligned}$$

- This means that there are still only 2^{32} possibilities for the key.

8

- The Netscape browser tried to do better:

```
procedure NetscapeRandSetup()
    pid = process ID;
    ppid = parent process ID;
    seconds = current time of day
        (seconds);
    microseconds = current time of day
        (microseconds);
    x = concatenation of pid, ppid,
        seconds, microseconds;
    NSseed = SHA1(x);

function NetscapeGetRand()
    rv = SHA1(NSseed);
    NSseed = NSseed + 1 mod 2^160;
    return rv;
```

- This can be used as

```
algorithm K
    K <-$- {0,1}^128
    return K

function keygen();
    NetscapeRandSetup();
    tmp = NetscapeGetRand();
    key = first 128-bits of tmp;
    return key
```

- Despite the reasonable properties of SHA1 and the 160-bit output of the generator, an adversary can learn or guess x.

9

Randomness for encryption

- Designers of SSH, IPsec, SSL all assumed that the last blocks of the ciphertexts in CBC can be used as IVs for the next ciphertexts.

Combining the schemes

- Recall that it is insecure in general to apply the Encrypt-and-MAC paradigm in order to achieve both privacy and authenticity.

Key management

- All users of the WEP encryption protocol use the same symmetric key.
- The key for the secure votes encryption in Diebold machines is hardwired in the code:

```
#define DESKEY ((des_key*)"F2654hD4")
```

10

Reference

- Y. Kohno "Implementation pitfalls". Available at http://www.cse.ucsd.edu/~mihir/cse107/yoshi.pdf

11