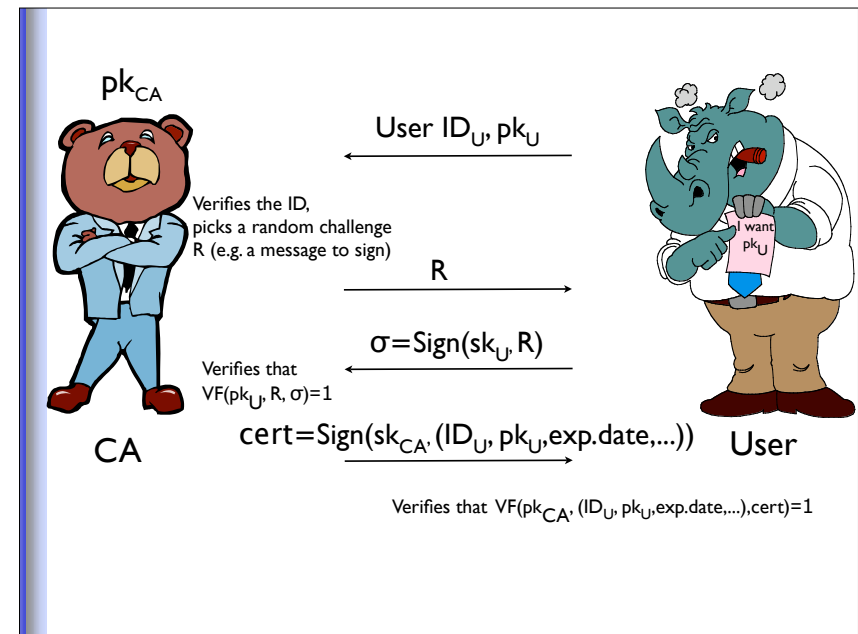


CS 4803 Computer and Network Security

Alexandra (Sasha) Boldyreva
PKI, secret key sharing,
implementation pitfalls .

1



2

- The PKI also
 - makes the certified public keys, the corresponding identities and the certificates public
 - maintains the public certificate revocation list (CRL)
- The PKI may be hierarchical with CAs certifying other CAs.
- X.509 is the standard for digital certificates developed by the International Telecommunications Union (ITU).

3

References

- Internet X.509 Public Key Infrastructure - Certificate Management Protocol (CMP). Internet draft. Available from <http://www.ietf.org/internet-drafts/draft-ietf-pkix-rfc2510bis-09.txt>
- C. Ellison and B. Schneier, "Ten risks of PKI." Available at <http://www.schneier.com/paper-pki.html> (linked from the class web page)

4

Secret key sharing

- Security of all symmetric and asymmetric schemes relies on secrecy of a secret key.
- How to make a secret key "more secret"?
- An idea: let's split a secret key K and store the shares in different places (e.g. on n different computers), such that
 - any t shares allow to reconstruct K
 - if $t-1$ computers become compromised, we are still fine in that no one can learn anything about K from $t-1$ shares
- To do any harm an adversary must compromise t computers
- This is (t,n) -secret sharing scheme.

5

Shamir's secret sharing

- Let p be a large prime.
- To (t,n) share a secret $z \in \mathbb{Z}_p$:
 - Choose $t-1$ random elements of \mathbb{Z}_p : a_1, \dots, a_{t-1} . Let $a_0 = z$. View these as the coefficients of a polynomial f of degree $t-1$, meaning $f(x) = a_0 + a_1x + \dots + a_{t-1}x^{t-1}$
 - Store $y_i = f(i)$ on each computer $i = 1, \dots, n$.
- To recover the secret given t pairs (i, y_i) for $i \in S$ use the Lagrange interpolation to find:

$$z = a_0 = f(0) = \sum_{i \in S} y_i \prod_{j \in S, j \neq i} \frac{-j}{i-j}$$

- The scheme is unconditionally secure ↘ Can be pre-computed

6

- There are several weaknesses of the Shamir's secret sharing protocol:
 - if some parties cheat during the secret reconstruction, the secret cannot be recovered and others cannot detect cheating
 - the dealer needs to be trusted
- A verifiable secret sharing protocol allows to overcome these difficulties
- It is also desirable that parties be able to perform secret-key operations (decryption or signing) such that no party holds the whole secret key at any time
- Threshold schemes allow to achieve this

7

(2,2) Visual secret sharing

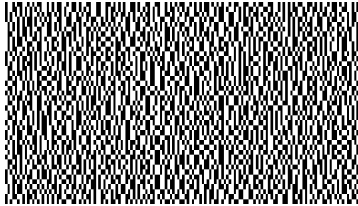
- Let's consider a protocol to $(2,2)$ - share a black-and-white image:
 - for each pixel compute the shares as follows:

pixel		share #1	share #2	superposition of the two shares
□	$p = .5$			
	$p = .5$			
■	$p = .5$			
	$p = .5$			

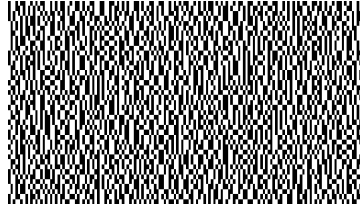
8

An example

Share 1



Share 2



The result? See in class

9

References

- Secret Sharing
 - David Wagner's lecture notes. Available from <http://www.cs.berkeley.edu/~daw/teaching/cs276-s04/22.pdf>
- Visual cryptography
 - Doug Stinson's visual cryptography page. <http://www.cacr.math.uwaterloo.ca/~dstinson/visual.html>

10

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.

11

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.

12

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.

13

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 a variation of CBCS\$, but did not pick a new random R each time.

14

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
```

$\leftarrow 2^{31}$

15

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

```
algorithm  $\mathcal{K}$  | function keygen()
   $K \xleftarrow{\$} \{0, 1\}^{128}$  | 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

```
key[1] = ((key[0] · 1103515245) + 12345) mod 231
key[2] = (((key[0] · 1103515245) + 12345) · 1103515245) +
         12345) mod 231
key[3] = ((((((key[0] · 1103515245) + 12345) · 1103515245) +
         12345) · 1103515245) + 12345) mod 231
```

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

16

- 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 2160;
  return rv;

```

- This can be used as

```

algorithm K
  K ←s {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.

17

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")
```

18

Reference

- Y. Kohno "Implementation pitfalls". Available at <http://www.cse.ucsd.edu/~mihir/cse107/yoshi.pdf> (linked from the class web page).

19