**CS 261: Computer Security** **Fall 2007**

Nov 13: Cryptography

*Lecturer: David Wagner* *Scribe: Gunho Lee*

# 1 How to solve security problem?

Here is a trajectory how people (and I) think about this problem.

## 1.1 Security is a cryptographic problem.

Many people thought that "crypto will save the world"

- Many people excited about 'Applied cryptography' in mid-90s. It shows bunch of fancy crypto protocol
- One good example of those protocol is gambling. There is physical protocol to prevent cheating in offline gamble. How can we ensure somebody is not cheating in online? Here is an simple example: Coin-toss.

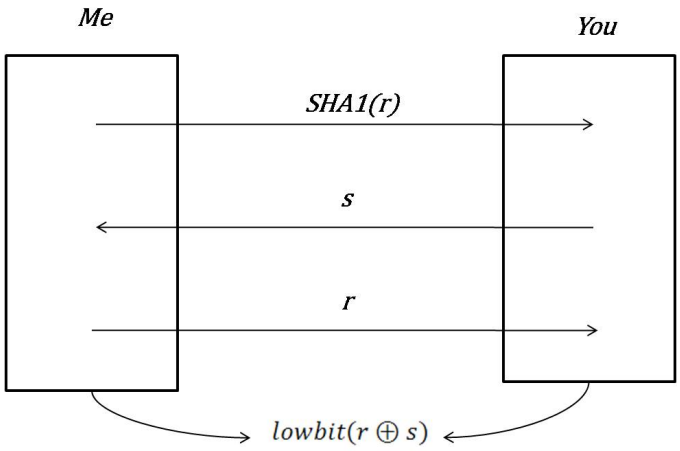


**Figure 1.** Coin-toss protocol

**r** is random value picked by Me

**s** is random value picked by You

I send /em SHA1(r) to you, and you send $s$ to me. Once we've done, I reveal $r$ then both compute $lowbit(r \oplus s)$. Now, the *lowbit* is the side of coin.

The protocol above:

1. Prevent ME from changing mind
   because *SHA1* hash is collision resistant, I can't come up with any value with same hash value.
2. Keep my number secret
   because *SHA1* hash is one-way hash function, you can't figure out what is my number.

We don't need to trust 3rd party player.

- Poker is more difficult than this simple example, but possible.
- Other examples include distributed computation, voting, etc ...
- Privacy is protected not by law, but by Math!! It is exciting! But ...

## 1.2 Security is a system problem

It turns out that the problem is human, not math. People misuse the system.

- Crypto only solves tiny piece of the problem. Majority of security problem was buffer overrun, abusing/misusing/misconfiguration ...
- If you think cryptography will solve the problem, you don't understand both problem & cryptography.

## 1.3 Security is an economics / incentives problem

Recently, people realized that incentive really does matter in security.

- Difference of bank system between US and UK is a good example. In case of fraud dispute, customer should prove it is a fraud in UK. In opposite, bank should prove it is not a fraud in US. It sounds like that UK case is better from the point of bank's view. In reality, UK banks have less incentive to make system secure, and lose more money due to fraud.
- Many software companies don't have great incentive to make their software secure, because customers don't like the situation that security feature bothers them.
- SSL is well made from the technical/cryptographic point of view. It prevents eavesdropping, but most information is stolen at the endpoint (compromised client / fake site). SSL does not prevent phishing or pharming. However, SSL is promoted because it is good for websites or browser vendor to claim that "security does matter for us!".

### 1.4 Crypto protocol is solving the wrong problem!

Nobody uses elegant cryptographic protocol. They just trust servers. Most concern is in client side.

## 2 ATM card system

### 2.1 1st generation

ATM card holds $(acc\#, E_k(PIN))$, where $K$ is secret key, and $PIN$ is chosen by user.

- Online ATM : send $E_k(PIN)$ to server to verify
- Offline ATM : ATM has tamper resistant HW with $K$, to decrypt $PIN$

Because account number was not tied to $PIN$, bad guy can replace account number with other's number, and draw money using his own $PIN$.

### 2.2 2nd generation

Now, $PIN$ is fixed function of account number like this $f(DES_k(acc\#))$, where $f$ is decimalization function (64bit string to 4 digit number)

- ATM card holds *(acc#, offset)*, where *offset* is used to calculate *PIN* from user-input value
- In this case, Insider of bank can see $PIN$. Moreover, there was a case that a bank only uses 3 $PIN$ for all accounts. This fact was kept secret because bank had no time to fix it.
- It became public at 2005, at that time they no longer uses passive ATM card. Now they are using smartcard with CPU and storage.

### 2.3 Vulnerabilities of ATM system

- False terminals (phishing)
- Man-in-the-middle attack (trusted path problem)
- When calculating $PIN := f(DES_k(acc\#))+$ *offset*, Function $f$ is transforming hex digit to decimal digit. Because it is not uniform, it increases the chance of guessing PIN
- Crypto imperfect - DES (56bit version) is not quite good

# 3 Common failure modes

## 3.1 Bad random number generator

Early Netscape used random number based on process id and time of the day, which is guessable.

## 3.2 Homebrew crypto

Use standard algorithm. Don't make your own!

## 3.3 Omitting authentication

Encrypting only does not help!

## 3.4 Key management failure

System is well defined in most cases. Short-cut makes problem.

## 3.5 Implementation bugs

Here is my own experience.

- RC4 is a stream cipher. $c = m \oplus RC4(k)$
- Problem occurred when the file size is large. Source code was very short and fit into one page, but couldn't find where is the bug.
- Finally, I found that the following macro is the cause.

```
1 #define SWAP(x, y) \
2         x ^= y; \
3         y ^= x; \
4         x ^= y;
```

**Listing 1.** swap macro

- This macro does swap without temporary variable.

  If call $SWAP\ (a[i], a[j])$ when $i = j$, it replace $a[i]$ with 0.

  Eventually, every vector is replace with 0.

  Hence, message is XORed with 0 and actually not encrypted after certain amount of operation.

- Kerberos had similar bugs for several years.

## 3.6 Side channel attack

- Suppose that RSA decryption is implemented in HW using square & multiply method.

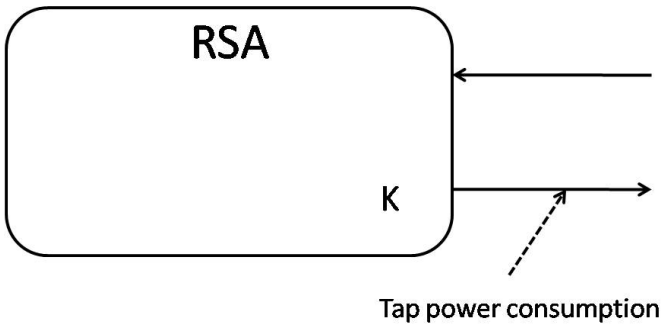


**Figure 2.** RSA

- Plain text is calculated like this : $p = c^d \mod n$
- Tap the wire to see power consumption of the HW.

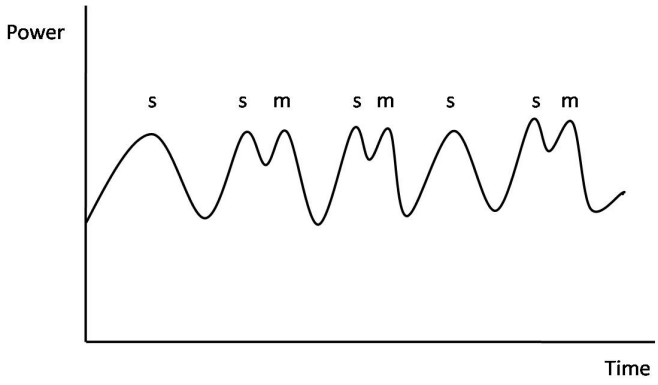


**Figure 3.** Side channel attack

- In the figure above,
  - single peak means s(square), which represents digit 0.
  - twin peak means s(square) and m(multiply), which represents digit 1.

  Hence, the adversary can read the decryption key : 01101...