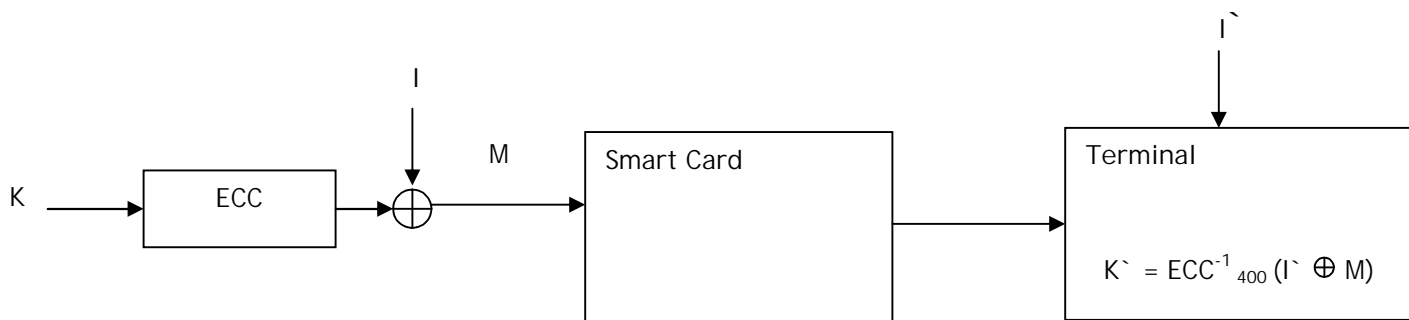


Review from last class - Biometrics to generate keys

Biometric deficiencies

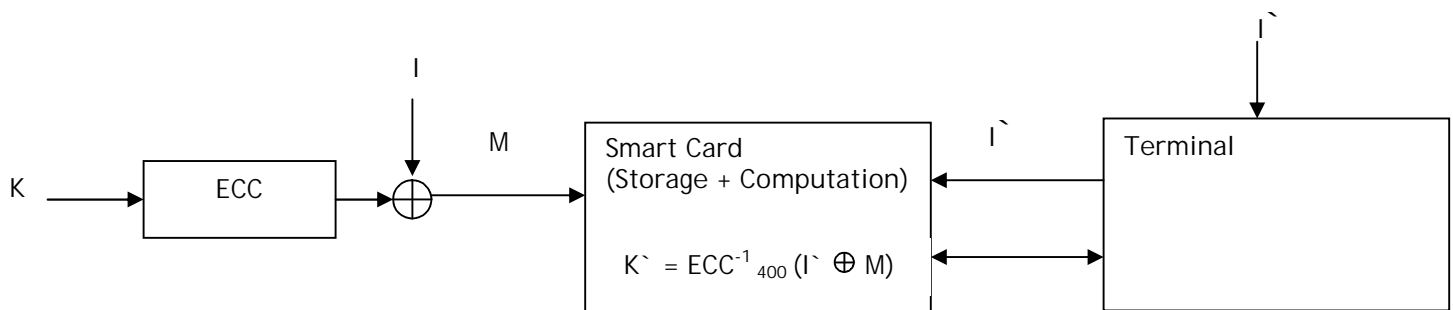
- One biometric for life
- Biometrics can be stolen e.g. by using photography
- Biometric reading are noisy (this can be fixed using ECC)

Naïve Scheme



This scheme cannot be used in public. Consider a scenario with a merchant and a customer who is making a payment using his smart card. Since the terminal knows K^{\wedge} the merchant can produce another smart card. The problem here is that the smart card is being used simply as storage.

Solution



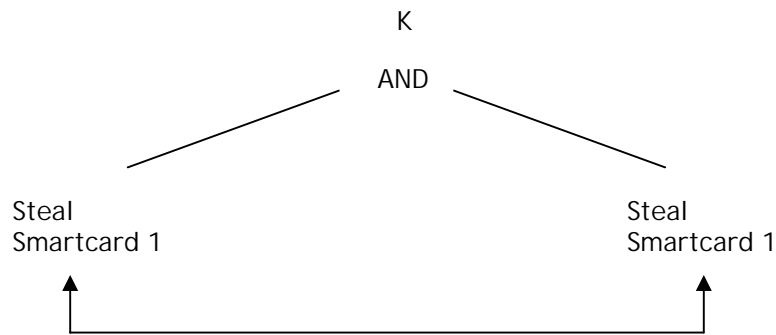
Deletes K^{\wedge} from memory after use

Idea is that nothing secret should ever leave the smart card.

Why is smart card + biometrics better than just using two smart cards?

It's often as easy to steal two smart cards as it is to steal one, since the user will usually keep them together. Stealing an iris, though, requires a completely different attack from stealing a smartcard.

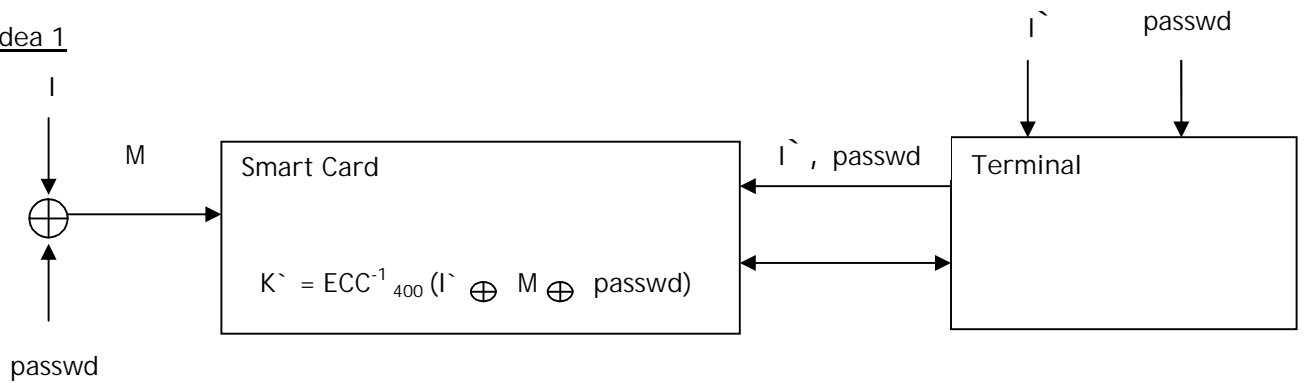
Attack Tree



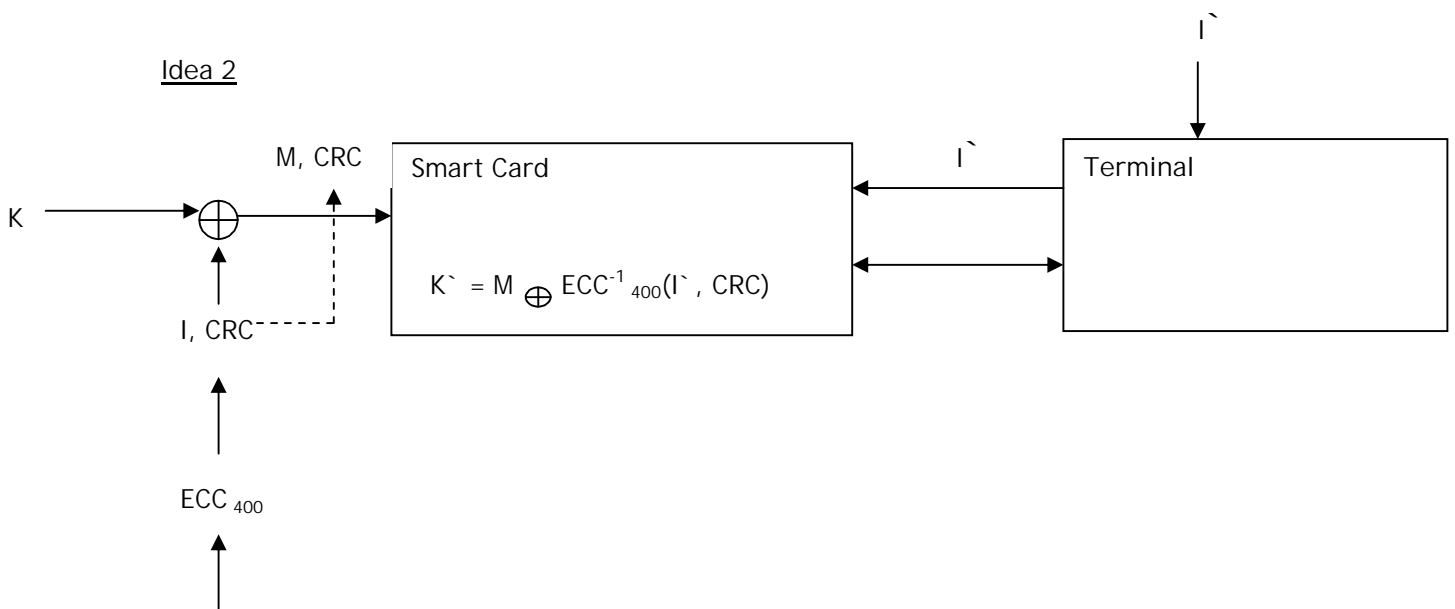
Steps to steal the two items are exactly the same and There is a good chance that they can be stolen simultaneously

3 Factor Authentication - add something you know

Idea 1

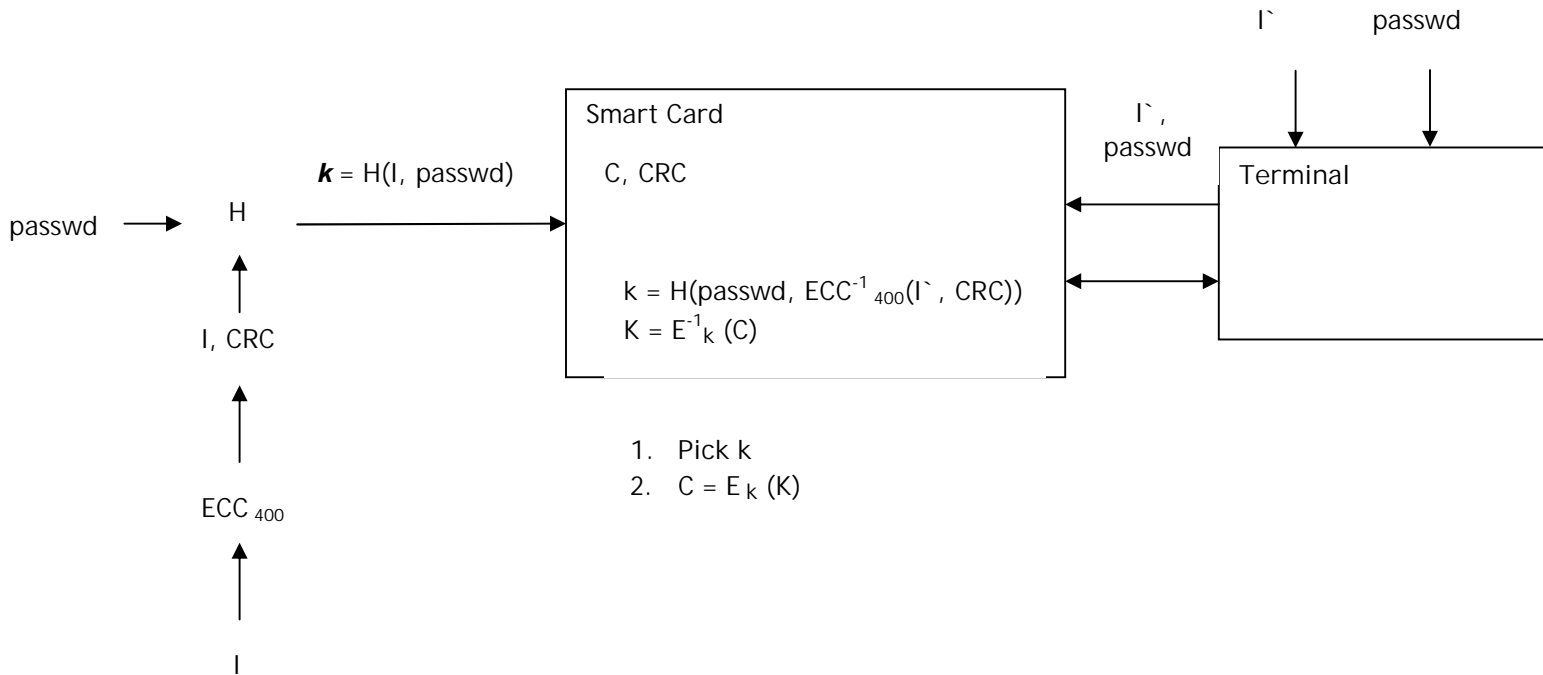


Idea 2



Can we use the password now?

Idea 3



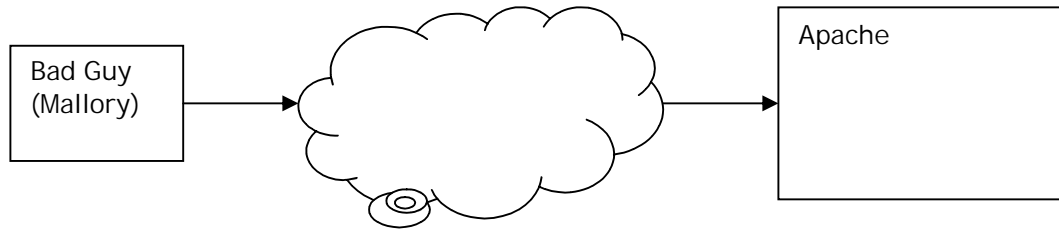
User Carries

- Password
- Iris
- Smart Card $[C, CRC]$

Software Security

- Buffer Overflows
- Format string bugs
- Race conditions
- SQL Injection bugs
- Tractor beaming
- XSS bugs
- Chroot/chdir bug
- File descriptor attacks
- RPC implementation bug
- Bad Software design
- Usability flaws

Except for the last two all are mistakes made by the programmer.



“AAAA
 AAAA
 AAAA

 AAAA”

- Apache ends up running attacker provided code
- Apache runs as root on unix

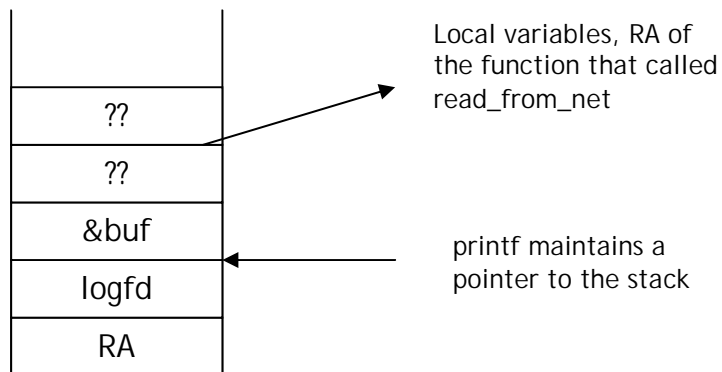
Un-trusted Input Bugs

- format_string.c

```
void read_from_net()
{
    char buf[1000];
    net_read(nuf, 1000);
    fprintf(logfd, buf);
}
```

say buf = "%s%s%s"

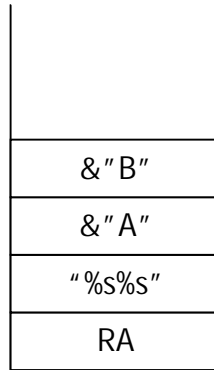
Stack



How printf works?

```
Printf("%S%S", "A", "B");
```

Stack



Problem!

```
Printf("Hello%n", &bytes);
```

Writes number of bytes written so far to *&bytes

```
Void login_user()
```

```
{  
    int authenticated = 0;  
    char uname[100];  
    getusername();
```

```
}
```

```
void getusername(char* uname)
```

```
{  
    net_read(uname);  
    fprintf(logfd, uname);  
}
```