# CS915/435 Advanced Computer Security - API Security

API Security

# Outline

- Introduction
  - What’s Security API?
  - Origins of Security APIs?
- APIs attacks on Hardware Security Module
  - PIN offset attack
  - Decimalisation table attack
  - XOR-To-Null attack
  - Meet-in-the-Middle attack

# What’s Security API

- A security API is an application programming interface that uses cryptography to enforce a security policy on the interactions between two entities.
- For example, APIs for a tamper resistant device provide crypto functions that are enforced by some security policy
  - Generate a key on certain conditions   genKey() → i
  - Encrypt with the key on certain conditions   Encrypt(m,i) → C
  - Decrypt with the key on certain conditions   Decrypt(C,i) → m

## Origins of Security APIs

- Born in an age when dedicated hardware was necessary in order to do cryptography
- In the 1970s, computers needed a simple command set to govern communication with the hardware.
- As digital cryptographic equipment became smaller and more portable, secure hardware increasingly adopted by the military, e.g., to secure battlefield communication.
- What if the secure hardware is captured in the battlefield?
- Long-term safety of the secret inside not guaranteed.
- Tamper resistance only provided a partial solution.

# The Killer App

- The ATM turned out a killer app to introduce security APIs to the commercial world
- ATMs are now a common feature of everyday life, but the technology took several decades to develop.
- ATM first introduced in 1967 by Barclays Bank in the UK
- It started out as a simple cash-dispensing device
- As the international ATM network grew, the environment becomes more dynamic
- Insider threats from the maintenance staff who needed full access to the system to maintain and upgrade the software.
- Hardware Security Module arose to address that problem.

## Aims of HSM

- HSM has to fulfill two aims:
  1. To **isolate** the defined **code** which is security relevant and to physically separate it into a device not under control of the system administrator
  2. To **minimise** the amount of security **relevant code** and to keep the rules governing its use as simple as possible, so that it would be easier to avoid bugs.

## Origins of Security API Attacks

- Ross Anderson was the first to introduce API attacks to the academic community.
- In 1992, he became involved in as expert witness in a number of lawsuits in the UK, pertaining to so-called ‘phantom withdrawals’
- A classic paper “**why cryptosystems fail**” that focused on the known failure modes in the use of HSM

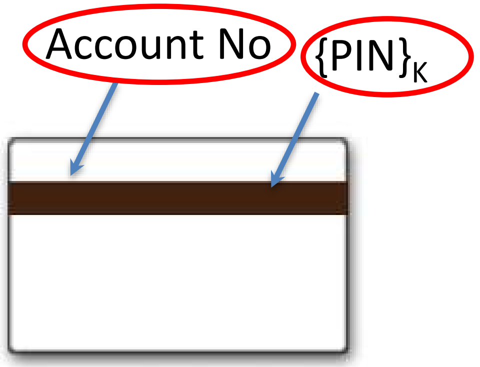
## Why cryptosystems fail?

- *"One large UK bank even wrote the encrypted PIN to the card stripe. It took the criminal fraternity fifteen years to figure out that you could change the account number on your own card's magnetic stripe to that of your target, and then use it with your own PIN to loot his account."*

-

Ross Anderson

# What happened?



- The PIN is encrypted by a secret key only known to the ATM.
- User 1: Account No 1, $\{PIN1\}_k$
- User 2: Account No 2, $\{PIN2\}_k$
- What if User 1 copies their own $\{PIN1\}_k$ to user 2's card?

## More findings on API attacks

- Many banks calculated customer PINs by encrypting the customer’s Primary Account Number (PAN) with a secret key
- And converted the resulting ciphertext into a four-digit number, which is the default PIN
- If customers wished to change PIN, the bank stored an offset

# An example with IBM 3624-Offset PIN Generation Method

| | |
|---|---|
| Account Number | 4556 2385 7753 2239 |
| Encrypted Accno | 3F7C 2201 00CA 8AB3 |
| Shortened Enc Accno | 3F7C |
| Decimalization table | 0123456789ABCDEF<br>0123456789012345 |
| Decimalised PIN | 3572 |
| Public Offset | 4344 |
| Final PIN | 7816 |

# Outline

- Introduction
  - What’s Security API?
  - Origins of Security APIs?
- APIs attacks on Hardware Security Module
  - **PIN offset attack**
  - Decimalisation table attack
  - XOR-To-Null attack
  - Meet-in-the-Middle attack

## A new feature - and a new bug

- One bank wished to restructure the customer PANs.
- Unfortunately, changing the PAN would change the original PINs
- Customers couldn’t accept new PINs
- So the solution was to use an HSM to adjust the offsets

# How did it work?

- As one bank's request, the HSM manufacturer added a transaction API:
  - Host -> HSM: old_PAN, new_PAN, old_offset
  - HSM -> Host: new_offset
- The **manufacturer** added warning that the above API was dangerous and should be removed after use, but the warning was forgotten.
- The API was never removed.

# Attack

- A year later, a programmer found how to abuse this API by feeding his own PAN (say attacker_PAN)

Host -> HSM: attacker_PAN, victim_PAN, attacker_offset
HSM -> Host: new_offset

The aim is to find out victim's PIN (= decimalised_PIN + offset)

Since the final PIN must remain unchanged, we have

attacker_PIN = new_offset + decimalised_PIN

Hence: victim_PIN = decimalised_PIN + offset
= attacker_PIN – new_offset + offset

## Essence of the API attacks

- In 2000 at Cambridge Security Protocols workshop, Ross Anderson asked: “So how can you be sure that there isn’t some chain of 17 transactions which will leak a clear key?”
- Basic idea: an unexpected sequence of transactions which would trick a security module into revealing a secret in a way the designers couldn’t possibly have intended.

# Outline

- Introduction
  - What's Security API?
  - Origins of Security APIs?
- APIs attacks on Hardware Security Module
  - Redundant API attack
  - **Decimalisation table attack**
  - XOR-To-Null attack
  - Meet-in-the-Middle attack

# Recall how the PIN is generated

| | |
|---|---|
| Account Number | 4556 2385 7753 2239 |
| Encrypted Accno | 3F7C 2201 00CA 8AB3 |
| Shortened Enc Accno | 3F7C |
| Decimalization table | 0123456789ABCDEF |
| | 0123456789012345 |
| Decimalised PIN | 3572 |
| Public Offset | 4344 |
| Final PIN | 7816 |

# Big flaw in the API

- The decimalization table is not fixed.
- An insider attacker can exploit this flaw to find out any bank customer's PIN in 15 attempts rather than 10,000.
- For example, use

  0123456789ABCDEF

  0100000000000000

- A trial PIN of 0000 will confirm the default PIN doesn't contain the number 1, so one trial eliminates $10^4 - 9^4 = 3439$
- It's possible to optimize the search so that it takes about 15 guesses instead of 10,000 to find out the PIN (see *"Decimalisation table attacks for PIN cracking", TR-560, Cambridge University, 2003*).

## Discovery of the attack

- The decimalisation table attack first discovered by Mike bond when he was investigating the “phantom” withdrawals in 2002.
- In 2003, Citibank asked the High Court to forbid the disclosure of the bank API vulnerabilities.
- But information already in the public domain was unaffected.

## Outline

- Introduction
  - What’s Security API?
  - Origins of Security APIs?
- APIs attacks on Hardware Security Module
  - PIN offset attack
  - Decimalisation table attack
  - **XOR-To-Null attack**
  - Meet-in-the-Middle attack

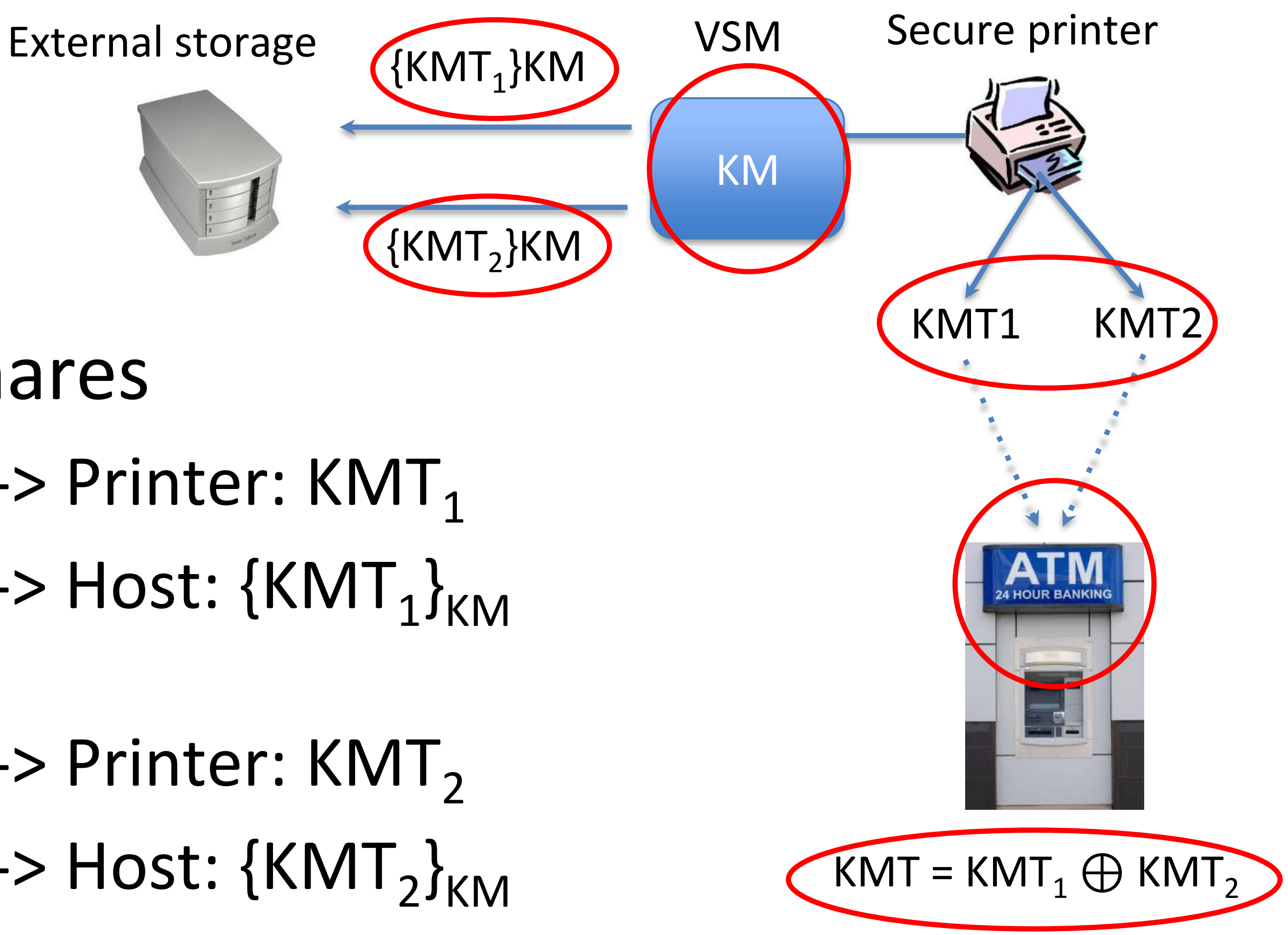
## Visa Security Module

- The VSM was introduced in the 1980s and widely adopted
- Members banks form a network of VSMs.
- Visa’s goal was to have banks hook up their ATMs with the VSM network, so customers of one bank could draw cash from another bank’s ATM.
- But, why should HSBC trust that Barclay to protect HSBC customers’ PINs?
- The trust is rooted to the tamper resistance of the VSM (and correctness of the APIs).

## Shared Control

- One problem was how to set up shared keys among ATM.
- No single employee of any bank in the network should learn the value of any customer’s PIN.
- The solution was to enforce a policy of **shared control**.
- Also called dual control

# Shares of Terminal Master Key



Why not store KMT in VSM?

- Print shares
  - HSM -> Printer: $KMT_1$
  - HSM -> Host: $\{KMT_1\}_{KM}$
  - HSM -> Printer: $KMT_2$
  - HSM -> Host: $\{KMT_2\}_{KM}$

# Attack on the Terminal Master Key (1)

- Combine key share

  Host -> VSM: $\{KMT_1\}_{KM}$, $\{KMT_2\}_{KM}$

  VSM -> Host: $\{KMT_1 \oplus KMT_2\}_{KM}$

- Attack

  Host -> VSM: $\{KMT_1\}_{KM}$, $\{KMT_1\}_{KM}$

  VSM -> Host: $\{KMT_1 \oplus KMT_1\}_{KM} = \{0\}_{KM}$

# Attack on the Terminal Master Key (2)

KPIN is the bank's PIN verification key

- Key wrapping API
  - Host -> VSM: $\{KMT\}_{KM}$, $\{KPIN\}_{KM}$
  - VSM -> Host: $\{KPIN\}_{KMT}$

- Attack
  - Host -> VSM: $\{0\}_{KM}$, $\{KPIN\}_{KM}$
  - VSM -> Host: $\{KPIN\}_{0}$

## Outline

- Introduction
  - What's Security API?
  - Origins of Security APIs?
- APIs attacks on Hardware Security Module
  - PIN offset attack
  - Decimalisation table attack
  - XOR-To-Null attack
  - **Meet-in-the-Middle attack**

# Meet-in-the-Middle attack

- VSM had an API to generate a check value for a terminal master key; this allows checking the correct entry of the master key at ATM
  - Host -> HSM: $\{KMT_1\}_{KM}$
  - HSM -> Host: $\{0000000000000000\}_{KMT1}$

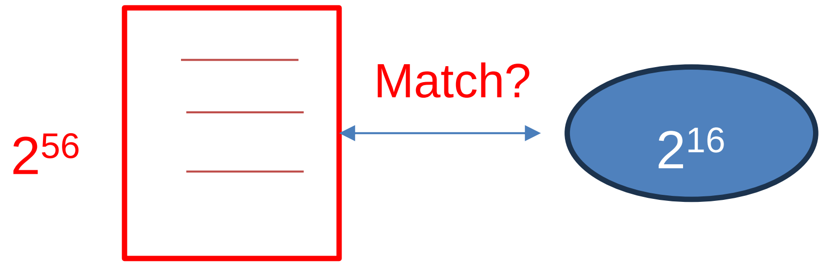
- With $\{0000000000000000\}_{KMT1,}$ you may try brute force, but it will take $2^{55}$ operations to break DES

## Threat Model

- Insider attacker who has physical access to HSM
- The goal is break one Terminal Master Key
- He needs just one key to break other keys
  - Simply call the HSM API to wrap other keys under the cracked key
- The effect of breaking one key will be catastrophic!

# How it works?

- At lunch time, he uses HSM to generate $2^{16}$=65536 keys and obtain 65536 check values
- His goal is to crack any one of the keys
- He goes home and does the exhaustive search
  - Put all 65536 check values in a hash table
  - Perform a brute force search, by guessing a key and computing its check value
  - Compare the check value against the hash table
- Repeat above for $2^{40}$ time to find a hit: one week on a normal PC!
- The bank’s estimate was $2^{16}$ weeks = 1000 years.



## Why it worked?

- DES key was weak, only 56 bit
  - A modern DES search machine can break the key in 56 hours.
  - Meet-in-the-middle attack reduces it to 40 bits.
- Speed and memory trade-off
  - A common attack technique
  - The attack doesn’t target a specific key, but target a large group of keys

## Summary

- Historic development of API attacks
- Four attacks that were unexpected by the banks
  - PIN offset attack
  - Decimalization table attack
  - XOR to Null key attack
  - Meet in the middle attack
- Trusted computing is on the rise
- Every PC will have a tamper resistant chip
- Are there more API attacks to be found?

# References

- Ross Anderson, “Why cryptosystems fail”, CCS, 2003.
- Mike Bond, “understanding security APIs”, PhD Thesis, Cambridge University, 2002
- Mike Bond, Piotr Zielinski, “Decimalisation table attacks for PIN cracking”, TR-560, Cambridge University, 2003.