Analysis of Security APIs (ASA-2) – June 26, 2008

Minimizing Threats from Flawed Security APIs: A Banking PIN Example

Mohammad Mannan

Carleton University

Observations

1. Designing 'perfectly secure' APIs seems difficult
2. With increased efforts we may improve API security
3. Formal proofs may help
   - but do not guarantee real-world security
4. Flaws will be found – tomorrow if not today
   - history suggests so
   - PIN cracking attacks (FC 2007, CHES 2001)

What should we do with flawed APIs?

1. Can we design APIs to minimize damage resulting from a flaw?
   - can damage estimation be included in API design?
2. What would be the criteria for such a design?

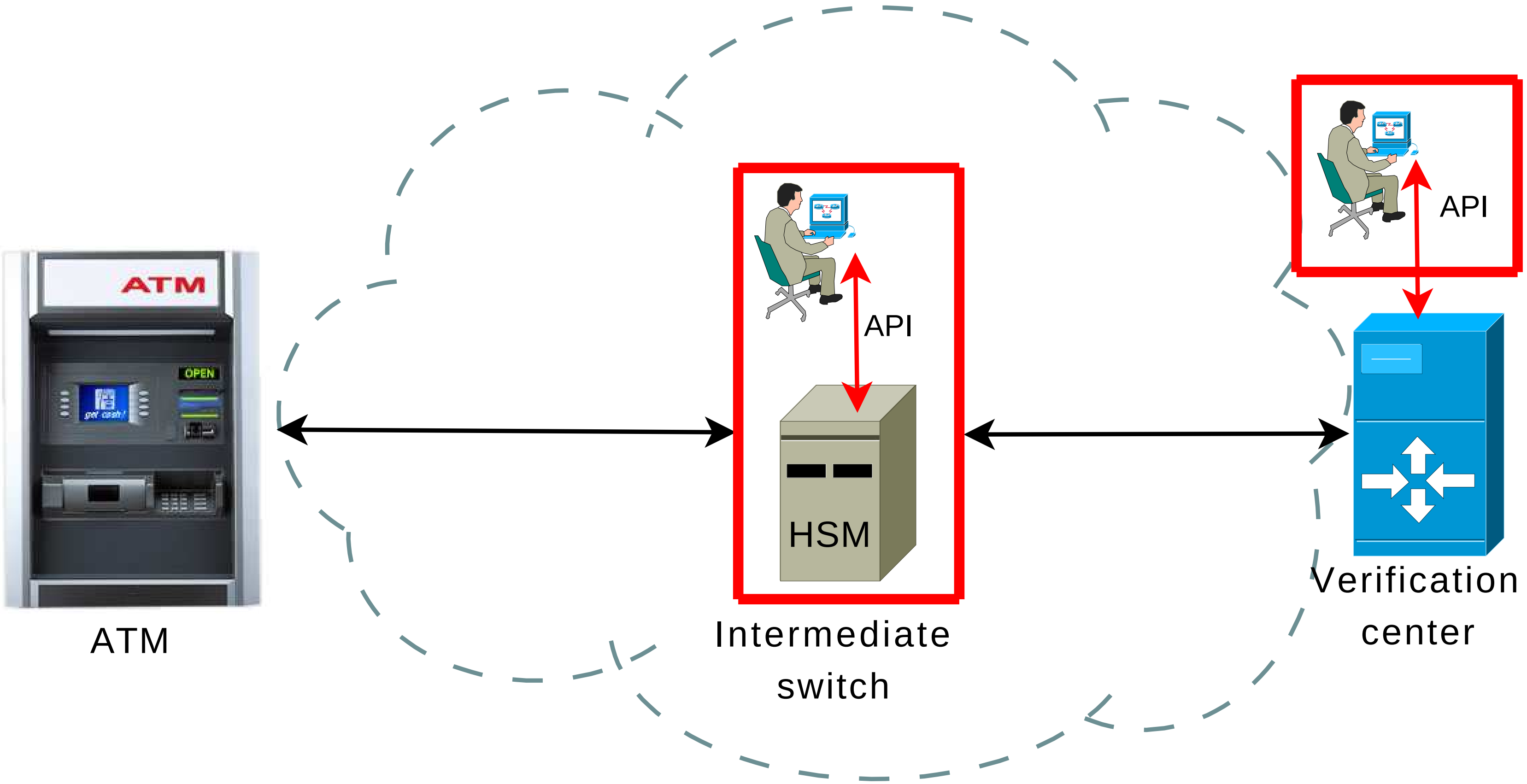
A specific case to consider

Weighing Down "The Unbearable Lightness of PIN Cracking"
(Financial Cryptography 2008)

Extended version available at:

`http://www.scs.carleton.ca/%7Emmannan/publications/saltedpin-tr.pdf`

PIN processing network



HSM = Hardware Security Module
EPB = Encrypted PIN Block

PIN cracking attacks

1. PIN processing APIs are decades old
   - several flaws have been uncovered allowing PIN extraction
2. "The Unbearable Lightness of PIN Cracking" (FC 2007) enumerates some very efficient attacks
   - we focus on the attacks outlined in this paper

An example attack: using translate-only APIs (FC 2007)

1. ISO-1 PIN format is not bound to any account number
   - other PIN formats can be translated to the ISO-1 format
2. Attack cost
   - setup: 10,000 EPBs with known PINs + 10,000 API calls
   - per-account: 2 API calls + search in a 10,000 items table
   - a more efficient attack requires only 100 special EPBs with known PINs

A recent attack

WIRED
BLOG NETWORK

Citibank Replaces Some ATM Cards After Online PIN Heist -- Update

By Kevin Poulsen June 20, 2008 | 9:05:00 PM Categories: Crime

Following up on my story Wednesday about the purported hacking of a Citibank ATM server, and the subsequent arrest of two cash-rich Brooklyn men, a New York Citibank customer says he received two notices this month from Citibank warning about breaches of a "third party" ATM processing system.

"These security breaches could have resulted in unauthorized access to your Citibank Banking Card number and associated Personal Identification Number (PIN)," the first notice, e-mailed on June 3, warned.

Result of a compromised third-party PIN processor?

Current (partial) ‘solutions’

1. Inter-banking agreements
2. Restricted APIs, i.e., unnecessary APIs in an HSM are disabled
3. Minor fixes for specific flaws
   - new flaws emerge often
   - applying fixes to intermediate nodes is difficult

Salted-PIN: motivation

1. Current Encapsulated PIN Block (EPB) contains customer PIN
   - we proposed to use secret 'salt' with the PIN
   - API flaws now may reveal the 'salted' (e.g. hashed) PIN, but getting the final user PIN still should be difficult (or 'computationally' infeasible)

Threat model

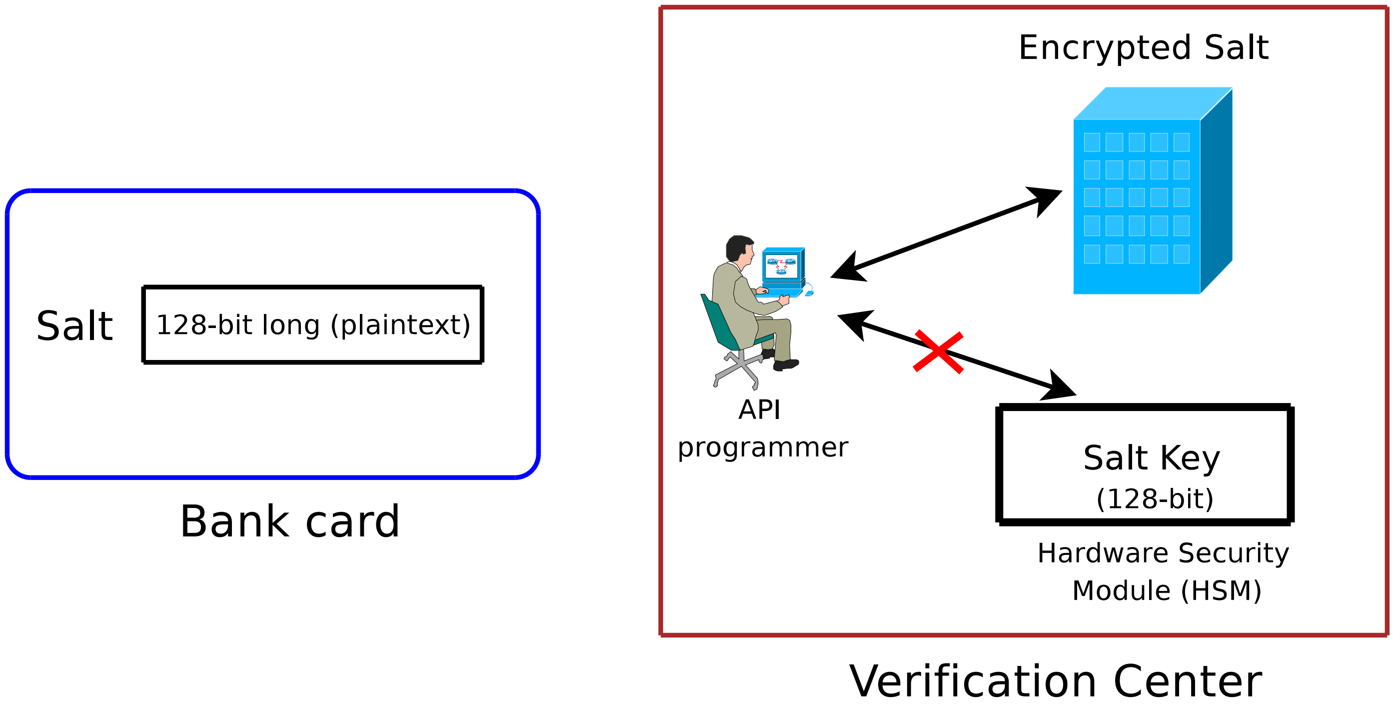
1. Attackers have access to
   - PIN processing APIs
   - transaction data (EPBs, account number)
2. No access to keys inside an HSM
3. Card skimming attacks are not considered

We focus on large-scale attacks that can extract e.g., millions of PINs per hour

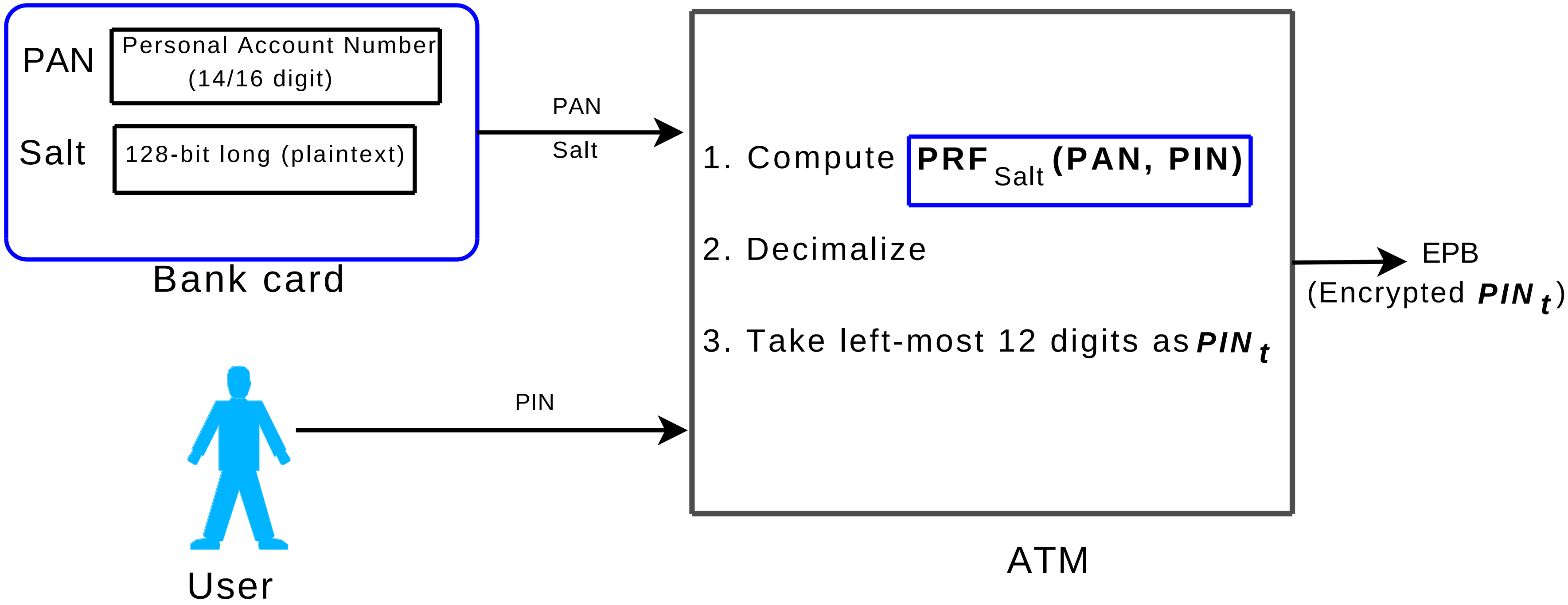
Salted-PIN: requirements

1. We require updating bank cards (data), ATMs and issuer/verification HSMs
2. We do not require any changes to
   - intermediate nodes
   - user behaviour

Salted-PIN: setup



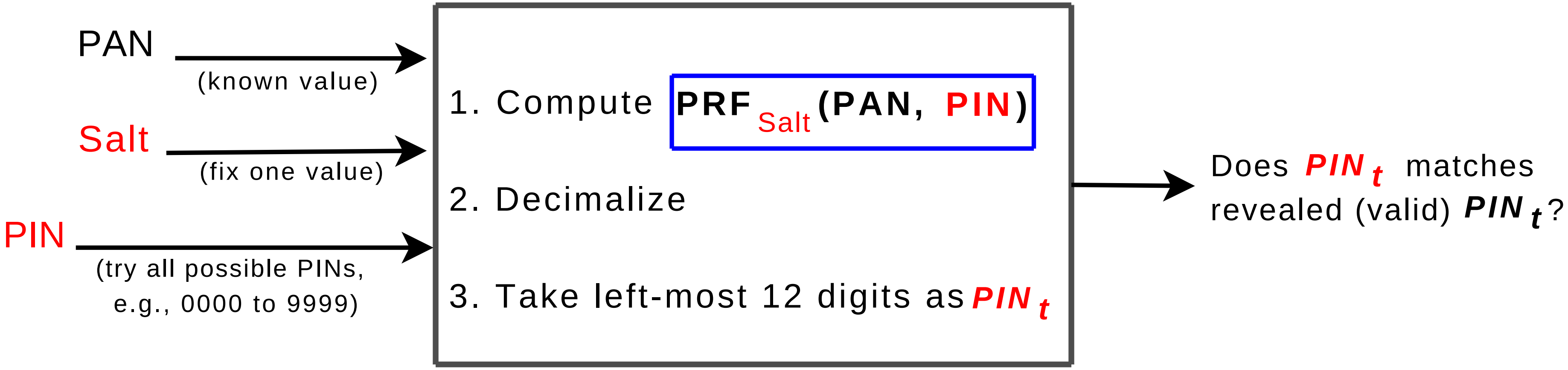
Salted-PIN: processing



previous attacks now reveal only PIN_t

PIN_t length limitations

Guessing attack



this search requires $O(2^{40})$ steps, but
setup cost is significant (10^{12} vs. 10,000 API calls)

A more efficient translate-only attack on salted-PIN

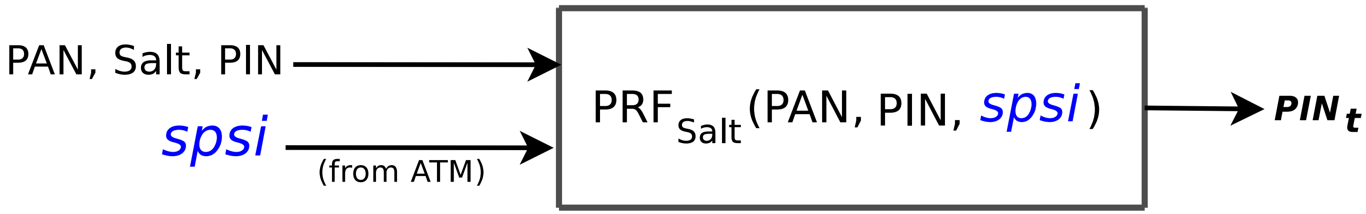
1. Trade-off between setup cost (EPB table size) and per-account attack cost can be exploited
   - for table size 10^n ($n \in \{2, 3, \ldots, 12\}$), the required number of per-account API calls is 10^{12-n}

Variant: double EPBs

1. Using 24 digits from PRF output, create two PIN_t values
2. Now two EPBs are required for PIN verification
3. Intermediate switches do not need to be aware of this
4. The cost of finding an appropriate salt value is now $O(2^{80})$

Variant: service-point specific

1. Use service-point specific information ($spsi$) for PIN processing



2. $spsi$ may include (see ISO 8583 Data Elements fields)
   - card acceptor identification code
   - card acceptor name/location

generates a localized PIN_t for each PIN verification

restricts a fake card to be used only from a particular location

Lessons learned

1. Minimize disclosure of sensitive info (e.g. customer PIN)
   - use long-term secrets to generate one-time passcodes
2. Make reuse of disclosed info "difficult"
   - currently attackers can compromise once and exploit any number of times from anywhere
   - 'localization' of exploits may reduce incentives for an attack

Attacks are still possible but "unattractive"

Concluding remarks

1. Assume flaws will persist even if we try our best
2. Design for damage control

Thank you ☺

Question/Comments?

`mmannan@scs.carleton.ca`

`http://www.scs.carleton.ca/~mmannan`