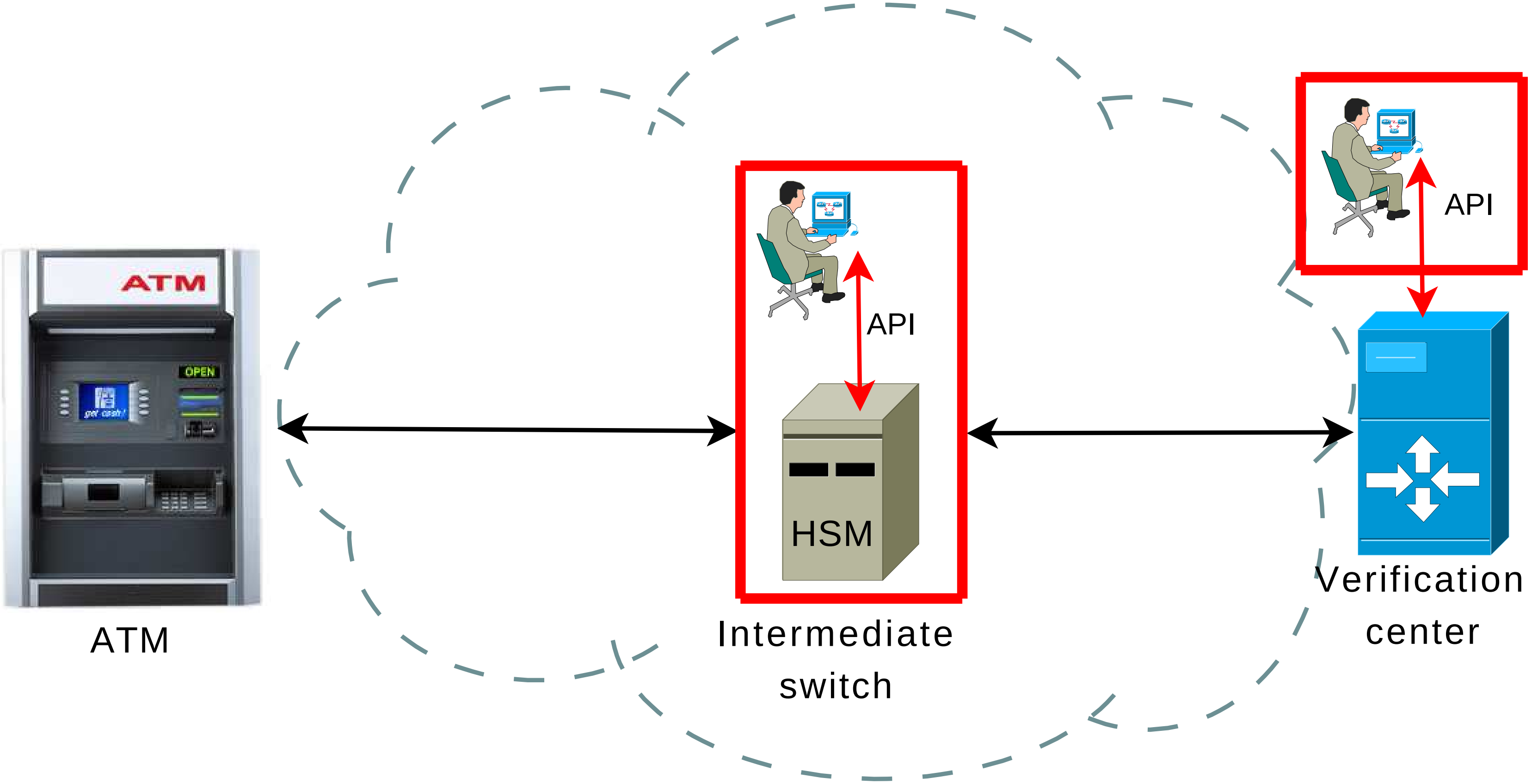
Financial Cryptography - Jan 30, 2008

Weighing Down
“The Unbearable Lightness of PIN Cracking”

Mohammad Mannan and P.C. van Oorschot

Carleton University

PIN processing network



HSM = Hardware Security Module
EPB = Encrypted PIN Block

PIN cracking attacks

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

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

Why is any particular solution interesting?

- A challenging problem since banking network is protected with symmetric crypto
  - – HSMs at intermediate nodes can ‘see’ everything
  - – intermediate nodes are untrustworthy

Salted-PIN: motivation

1. Lesson from history: API flaws will persist and attacks will continue

   – we focus on minimizing information disclosure

   (here customer PIN)

2. Current Encapsulated PIN Block (EPB) contains customer PIN

   – we propose to use secret ‘salt’ with the PIN

Salted-PIN: requirements

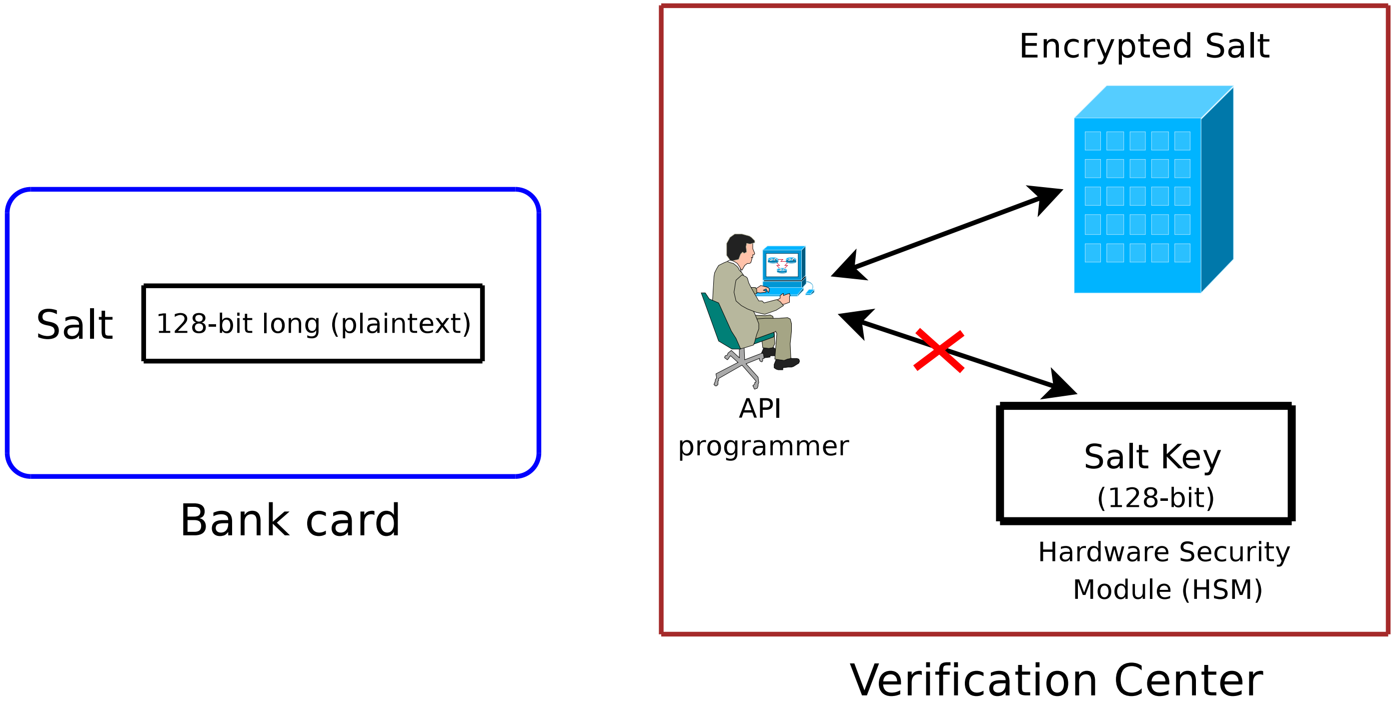
1. We require updating bank cards (data), ATMs and issuer/verification HSMs

2. We do not require any changes to

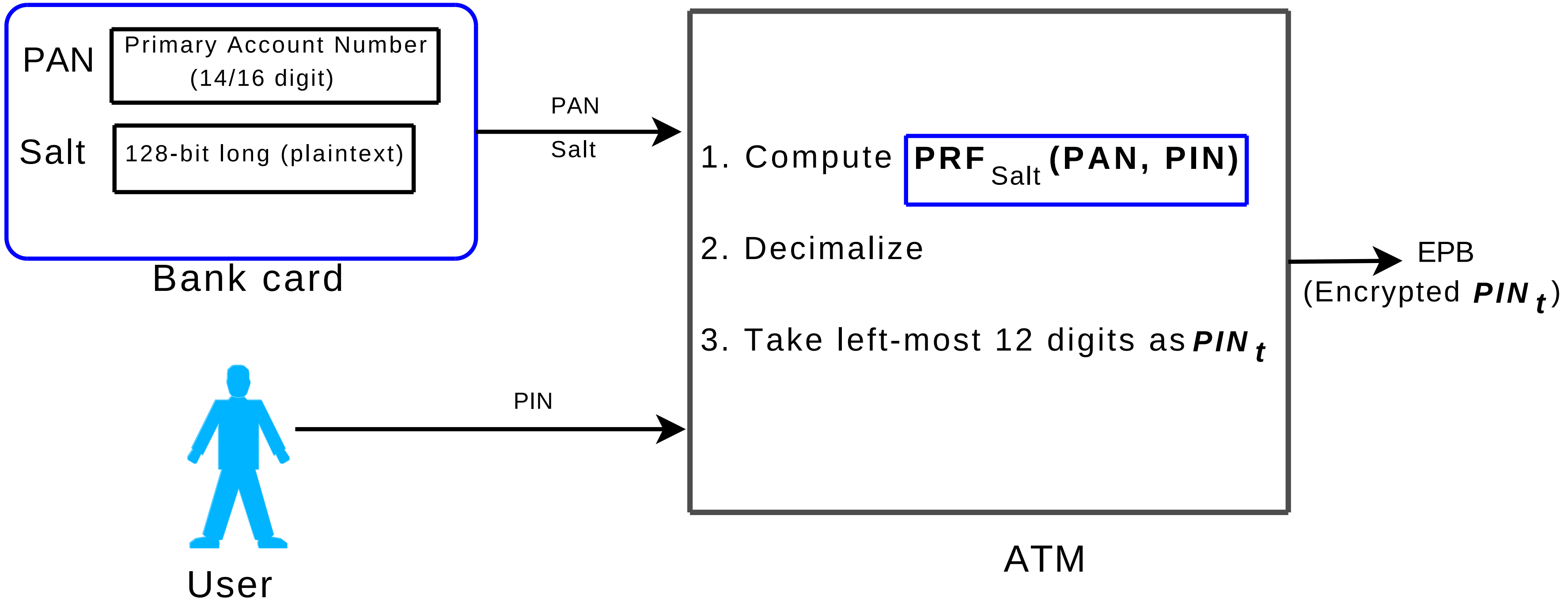
   – intermediate nodes

   – user behaviour

Salted-PIN: setup

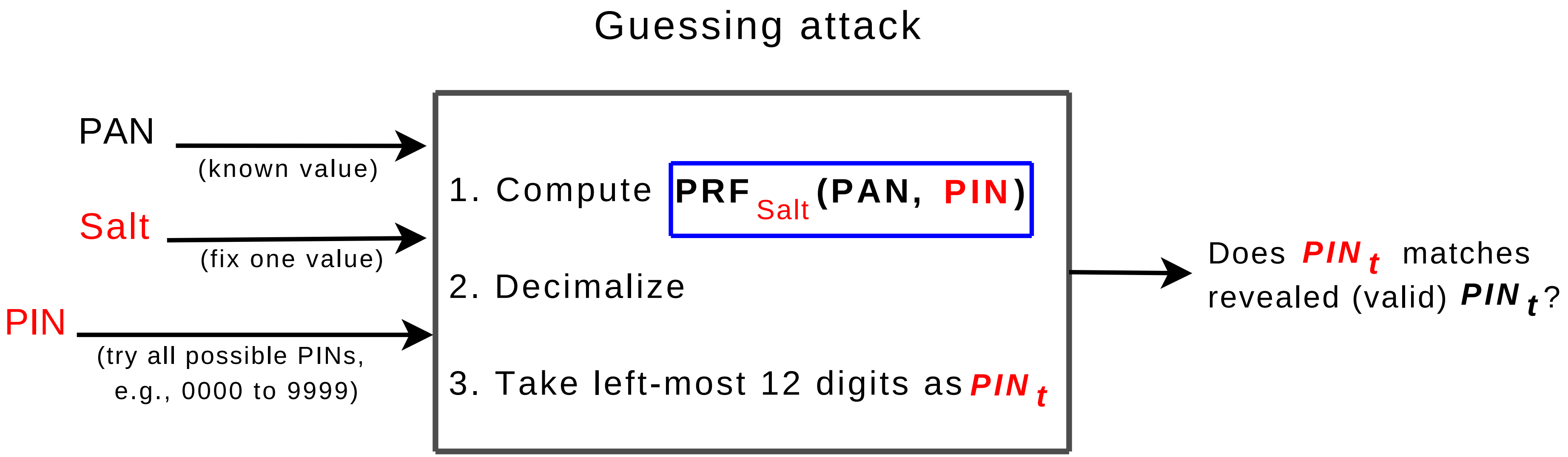


Salted-PIN: processing


PAN
Primary Account Number (14/16 digit)
Salt
128-bit long (plaintext)
Bank card
PAN
Salt
PIN
User
1. Compute PRF$_{\text{Salt}}$(PAN, PIN)
2. Decimalize
3. Take left-most 12 digits as PIN_t
ATM
EPB
(Encrypted PIN_t)

- previous attacks now reveal only PIN_t

PIN_t length limitations


Guessing attack
PAN
(known value)
Salt
(fix one value)
PIN
(try all possible PINs, e.g., 0000 to 9999)
1. Compute PRF$_{\text{Salt}}$(PAN, PIN)
2. Decimalize
3. Take left-most 12 digits as PIN_t
Does PIN_t matches revealed (valid) PIN_t?

- may have to try $O(2^{40})$ salt values

One variant of salted-PIN

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 $O(2^{80})$

Concluding remarks

1. PIN processing APIs should be designed assuming malicious switches

2. Deployment barriers to salted-PIN need more study