A Secure Mobile OTP Token

by: Fred Cheng

FPC Consultancy

International Technological University

fredtcheng@yahoo.com

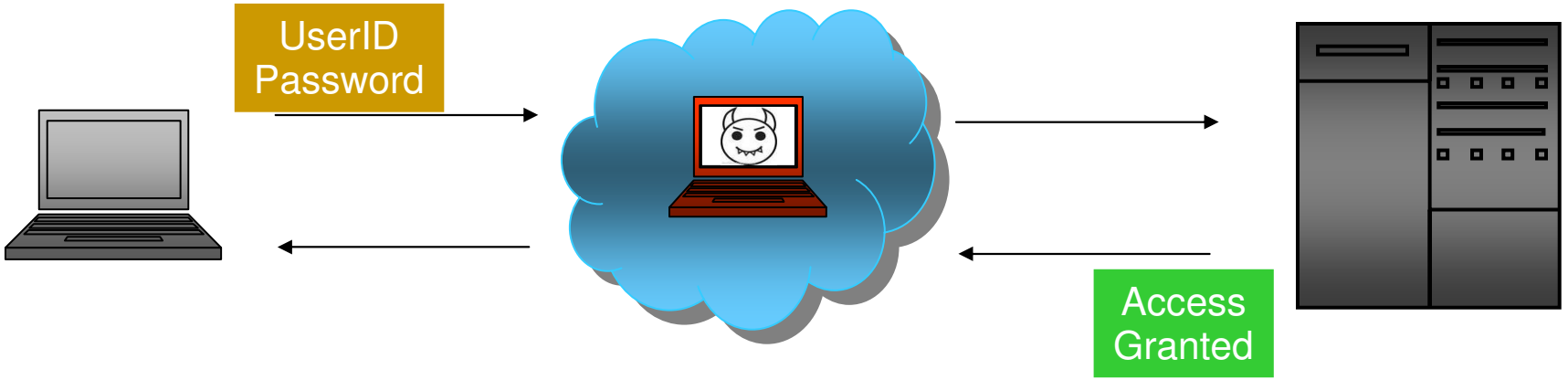
Topics

- Introduction
- Background & Selected Related Works
- Base Cipher
- Implementing a Secure Mobile OTP Token
- Security Analysis
- Conclusion

Introduction

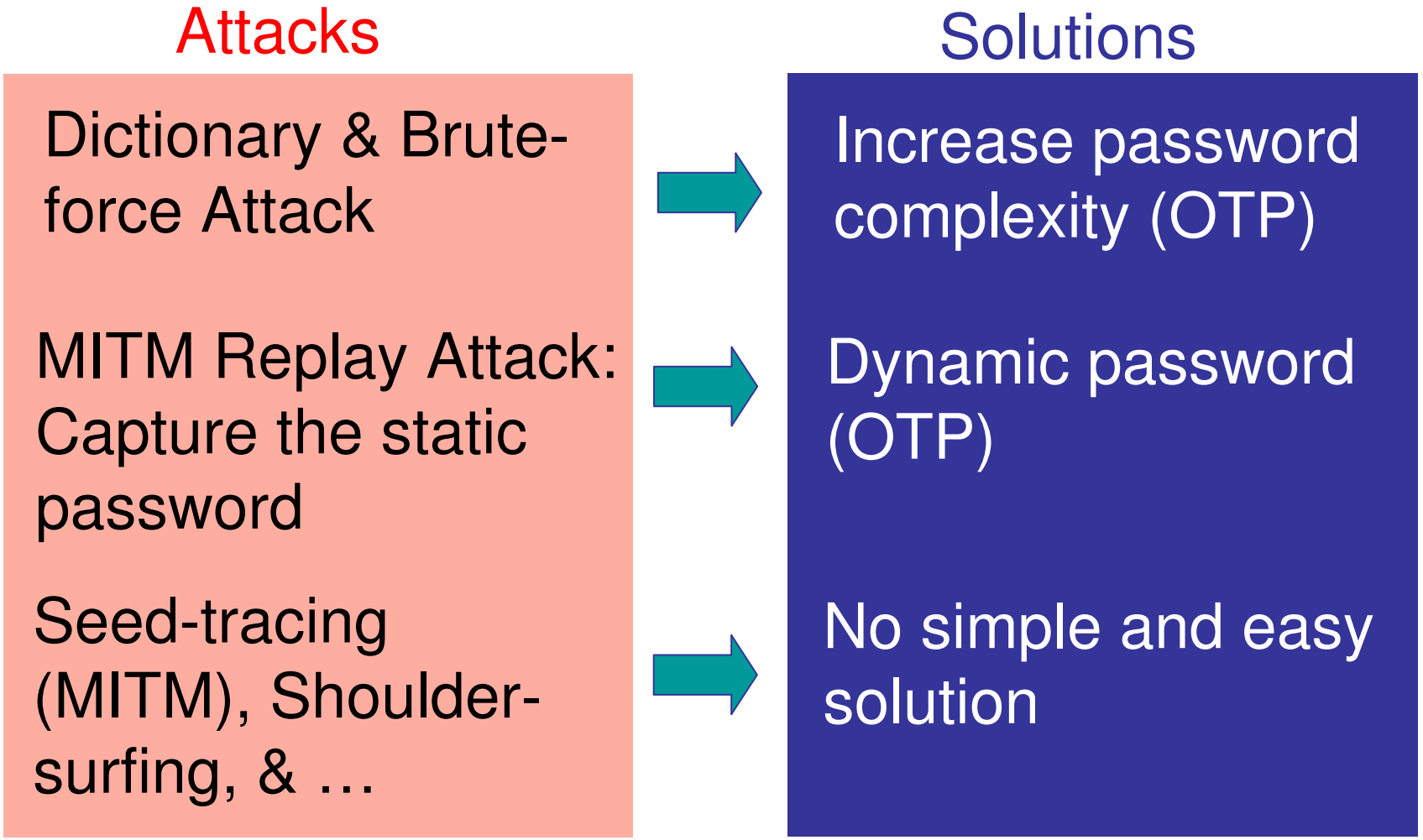
- Using One-time Password (OTP) for remote authentication becomes popular.
- It is natural to have mobile phone as an OTP token.
- This paper proposes an encryption cipher to build a secure Mobile OTP token that can resist certain security attacks.
- The token also preserves full compliance and interoperability with existing infrastructure.

Why OTP & OTP Token?



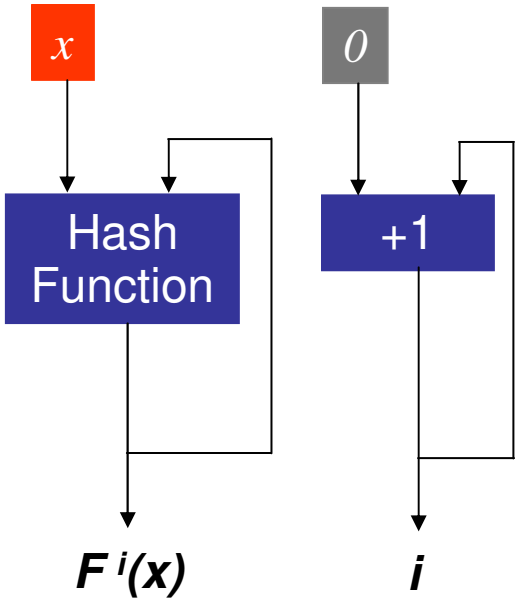
- OTP: One-time Password
  - For network remote authentication
- Security weakness with basic authentication
  - Publicly known UserID
    - John Dole at ACE Corp. ⟶ j.dole@ace.com
  - Static Password

Password Security Attack



Dynamic Password

- Proposed by Leslie Lamport
  - In his landmark 1981 ACM Paper
- The Algorithm
  - Using a One-way Function – F
    - An initial seed x & event counter i
  - Dynamic session password
    - $F(x), F(F(x)), \ldots, F^{i}(x)$
  - Each password is only used once in one session.

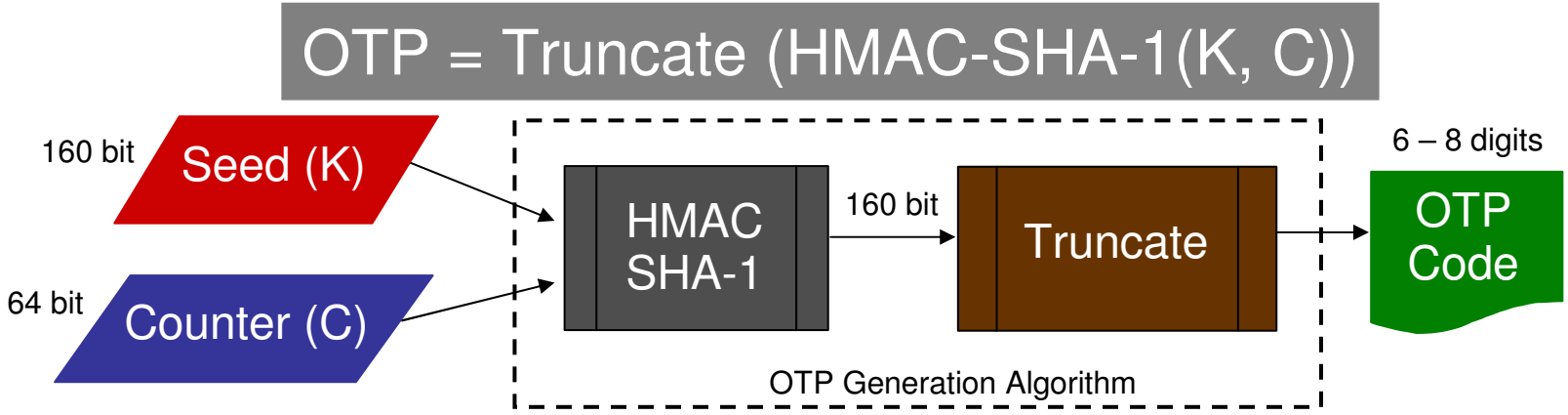


- The beginning of a One-time Password (OTP) development.

Ref [22]: Lamport, L., "Password Authentication with Insecure Communication". Communications of the ACM, 24(11): 770-772, Nov. 1981.

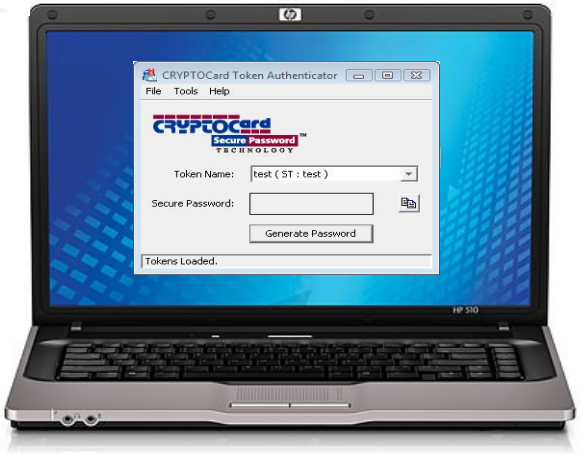
OTP Token

- Various OTP algorithms and implementations were introduced and marketed
  - Expensive, in-compatible & non-interoperable
- OATH: Initiative for Open AuTHentication
  - Free Standard, Compatibility, Interoperability & Low Cost
  - OTP Algorithm – RFC4226



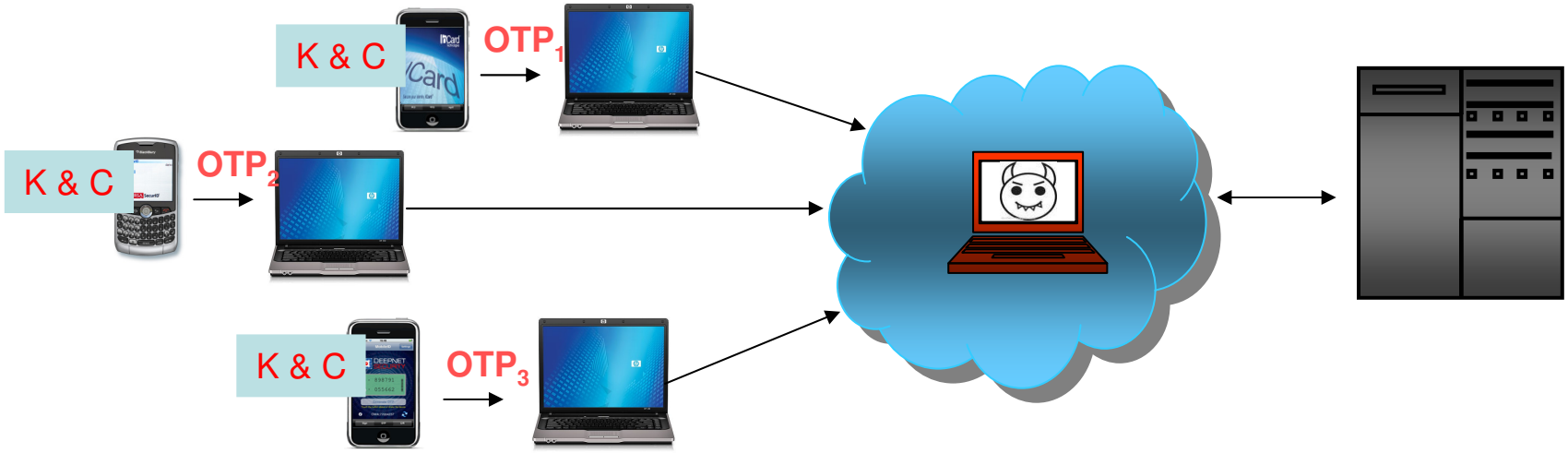
Variety of OTP Tokens

- Hardware
- Software
- Mobile OTP Token
  - Embeds OTP function in cellular phone



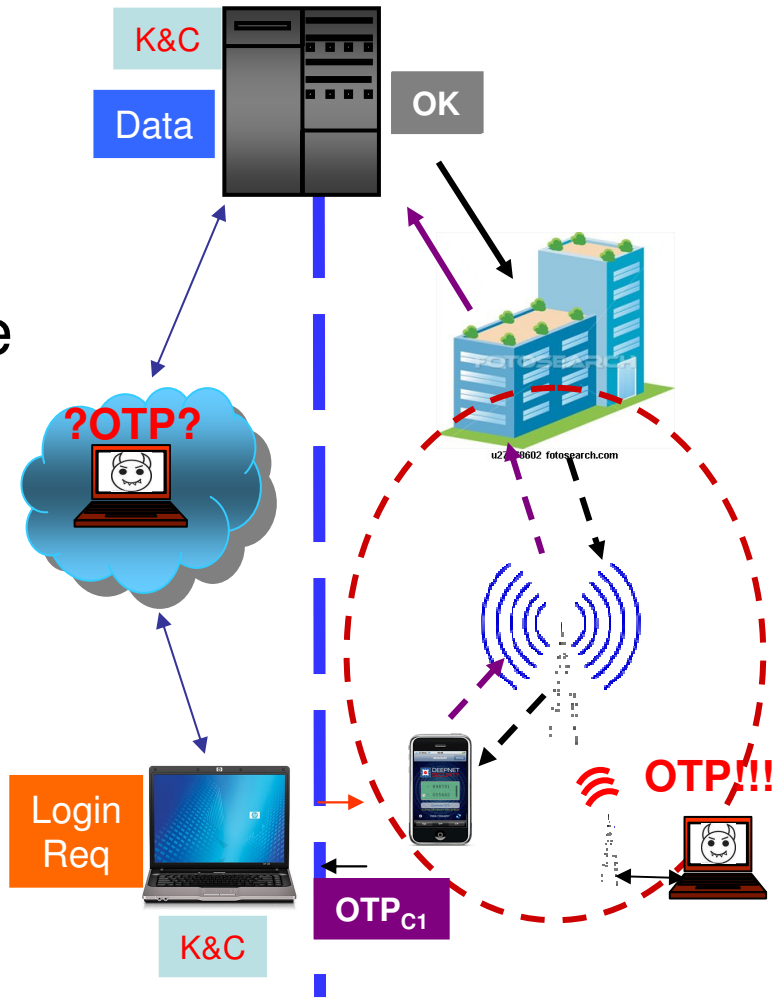
Stand-alone Mobile OTP Token

- Cellular phone is an OTP Token
  - Generating OTP code
  - Replacing the dedicated H/W or S/W OTP token
- Software based token
  - Seed (K) and Counter (C) are stored inside phone
- Ref: [1] [7] [8]



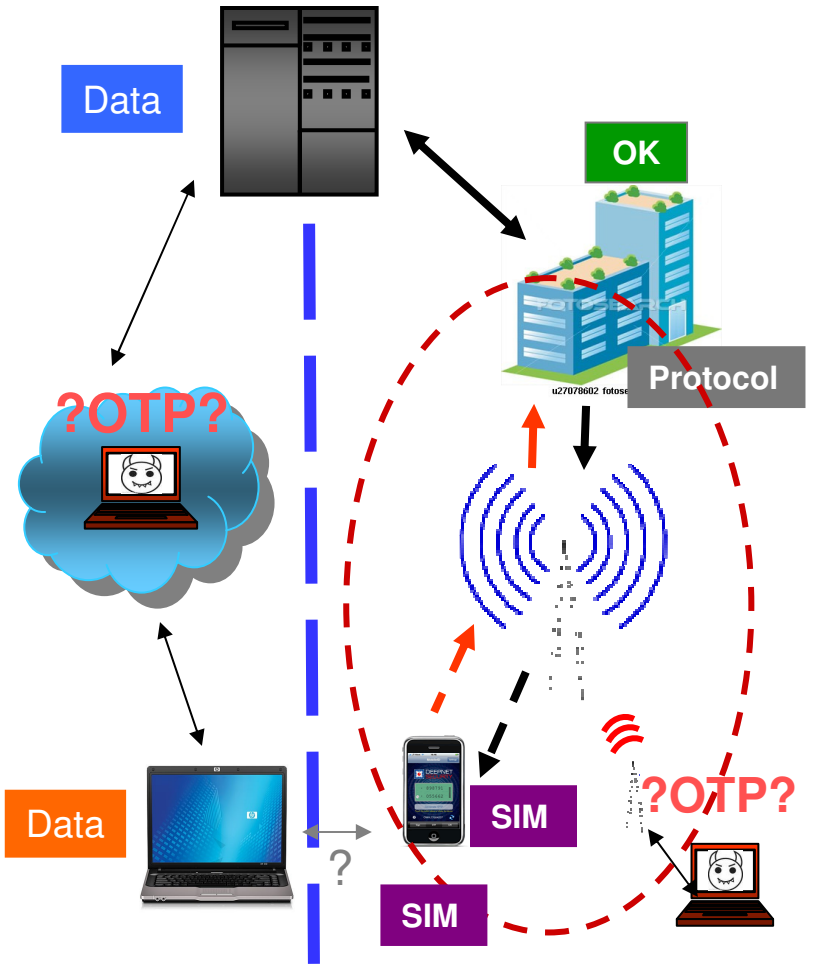
Out of Band Transceiver

- Cellular Network
  - Secure out of band channel
- Phone (using SMS)
  - Transceiver of the OTP code
- Seed (K) and C storage
  - At server or computer
- Limitation
  - Unreliable & untimely SMS
  - Cellular service coverage
- Ref: [2] [9] [10]



Mobile Authenticator

- Authentication
  - Provided by cellular system
  - New Protocols [9][10]
  - SIM (user credentials)
- Phone
  - Contains SIM
- Limitation
  - Cellular service coverage
- Ref: [3][13][14][15][16][17][18]



Mobile OTP Solutions

Items / Category	Stand Alone Token	Out of Band TXR	Mobile Authenticator
Role of phone	Computational platform	Transceiver of OTP code	Part of the authenticator
OTP generation	Phone	Phone or Server	Phone and Server
OTP submission	Through PC	SIM & SMS	SIM & Protocol
Type of phone	2.5G & up	2.5G & up	3G & up
Simple usability	Yes	No	No
Low phone $/m	Yes (zero)	No (SMS Plan)	No (3G Data Plan)
No cellular limitation	Yes	No	No
Compatibility & Interoperability	Yes	No	No
No system change	Yes	No (additional H/W,S/W)	No (complex system)
Protect secrets	No	Yes	Yes/ No
MITM attack safe	No	No	Yes
Shoulder-surfing attack safe	No	No	?

Low Cost Mobile OTP Token

- Stand-alone Mobile Token has its merits
  - Works with existing authentication infrastructure
  - Low cost – deployment & supporting
  - No cell coverage limitation
- Need to solve
  - Protecting the secrecies
    - Seed (K) and counter value (C)
  - Protecting security attacks
    - MITM seed tracing
    - Shoulder-surfing

Requirements & Solutions

- An Event-based OATH Mobile OTP Token
- Security Protection

Requirements		Solutions
Preserve phone power	⇨	Less computation & local code
Compatibility & Interoperability	⇨	Using same OTP algorithm
Protect local confidential data	⇨	Rubbing Encryption Algorithm
Resist security attack	⇨	Rubbing Encryption Algorithm
OTP	→	OATH OTP
Seed Tracing	→	New solution
Shoulder-surfing	→	New solution

Rubbing Encryption Algorithm

- A secure scramble algorithm that uses complex key

Features	Benefits
Key embedded in H/W token	No need to memorize key
Decrypting w/o Entering Key	Using long & complex key
Long & Complex Key	High security with short plaintext
Secure Scramble Algorithm	No complex computation

REAL Cipher - Math

Given a numeric image X containing T characters selected from Y numerals,

$$x_1 \quad x_2 \quad x_3 \quad \ldots\ldots\ldots\ldots\ x_i \quad \ldots\ldots\ldots\ldots\ x_t$$

The occurrence possibility (P_i) of a numeral Y_i (assumed appears N_{yi} times) is,

$$P_i = N_{yi} / T \qquad (1)$$

Following Shannon Entropy Theory, Image X's uncertainty H(X) is as follows.

$$H(X) = - \sum_{i=1}^{T} P_i \, (\mathrm{Log}_2 \, P_i) \qquad (2)$$

When each numeral has equal chance to be displayed and N_{yi} are all equal (N), image X reaches a **Equiprobable** state and has the highest uncertainty. [20]

$$T = NY \, , \qquad (3)$$

$$P_i = N_{yi} / T = N/ NY = 1/Y = P \, . \qquad (4)$$

REAL Cipher - Math

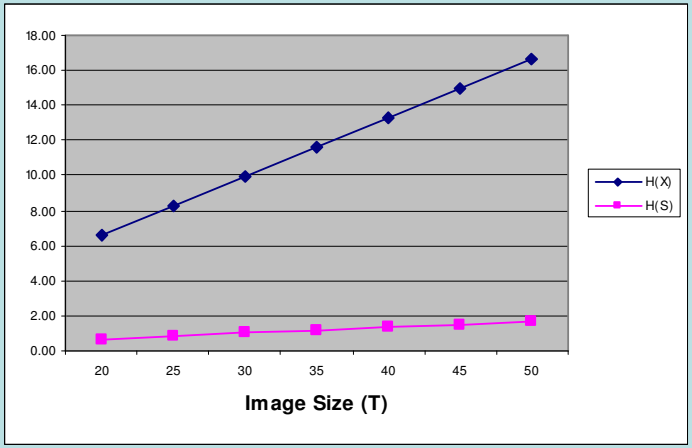
Substituting (4) into (2), Image X's uncertainty H(X) becomes

$$H(X) = T\,(\mathrm{Log}_2\, Y)/\, Y\ . \qquad (5)$$

Following similar procedure, each symbol's uncertainty H(S) can be found as follows

$$H(S) = T\,(\mathrm{Log}_2\, Y)/\, Y^2\ . \qquad (6)$$

An Equiprobable Numeric Image X



Variety of symbol (Y) is held at constant (= 10)

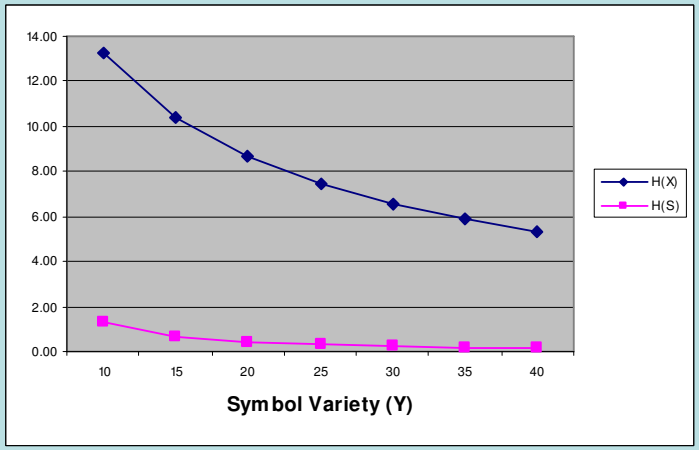
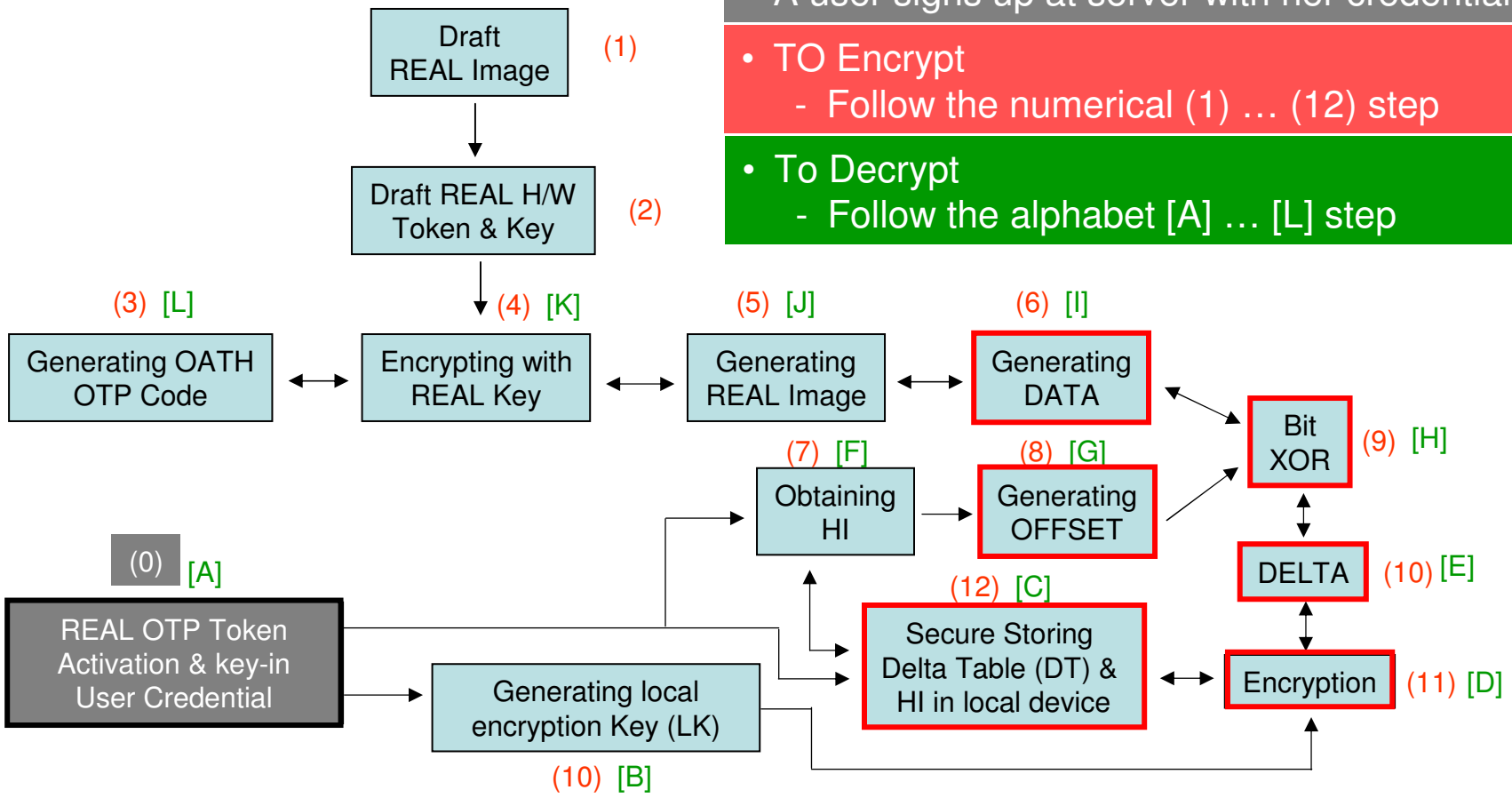


Image size (T) is held at constant (= 40)

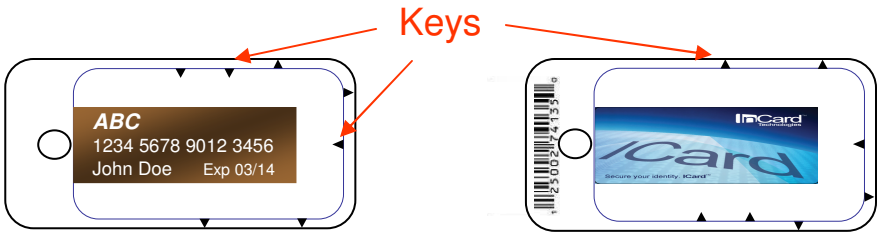
REAL Operating Procedure

- A user signs up at server with her credential
- TO Encrypt
  - Follow the numerical (1) … (12) step
- To Decrypt
  - Follow the alphabet [A] … [L] step



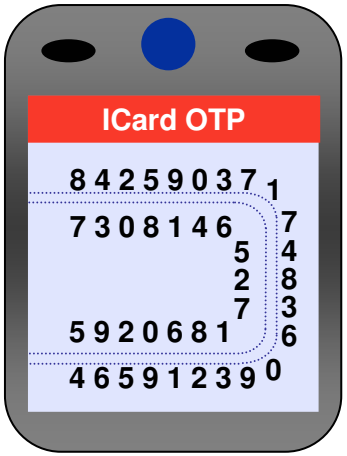
REAL Image-Token-Key (Step 1&2)

- REAL Image Generation
  - Security level setting
  - Screen size, font size, token size, usability and other factors
- REAL Hardware Token
  - Low cost, easy to carry and use
- REAL Key
  - Code pointer as key
- Token can be of multi-dimensions
  - REAL key can be in multi-dimensional form



REAL token front side

REAL token back side



REAL Image
T = 40
Y = 0 ~ 9 numeral

Encryption with REAL Key (step 3~6)

- REAL key position → $W_6 \sim W_0$
- Pre-generated OTP Code (= 807235)
  - $D_5 = 8$, $D_4 = 0$, $D_3 = 7$, $D_2 = 2$, $D_1 = 3$ and $D_0 = 5$
  - Program Digit: $W_6 = D_6 = 3$ (odd → front key, even → backside key)
- Randomly place other numerals to make X an Equiprobable Image

Placement of OTP Code

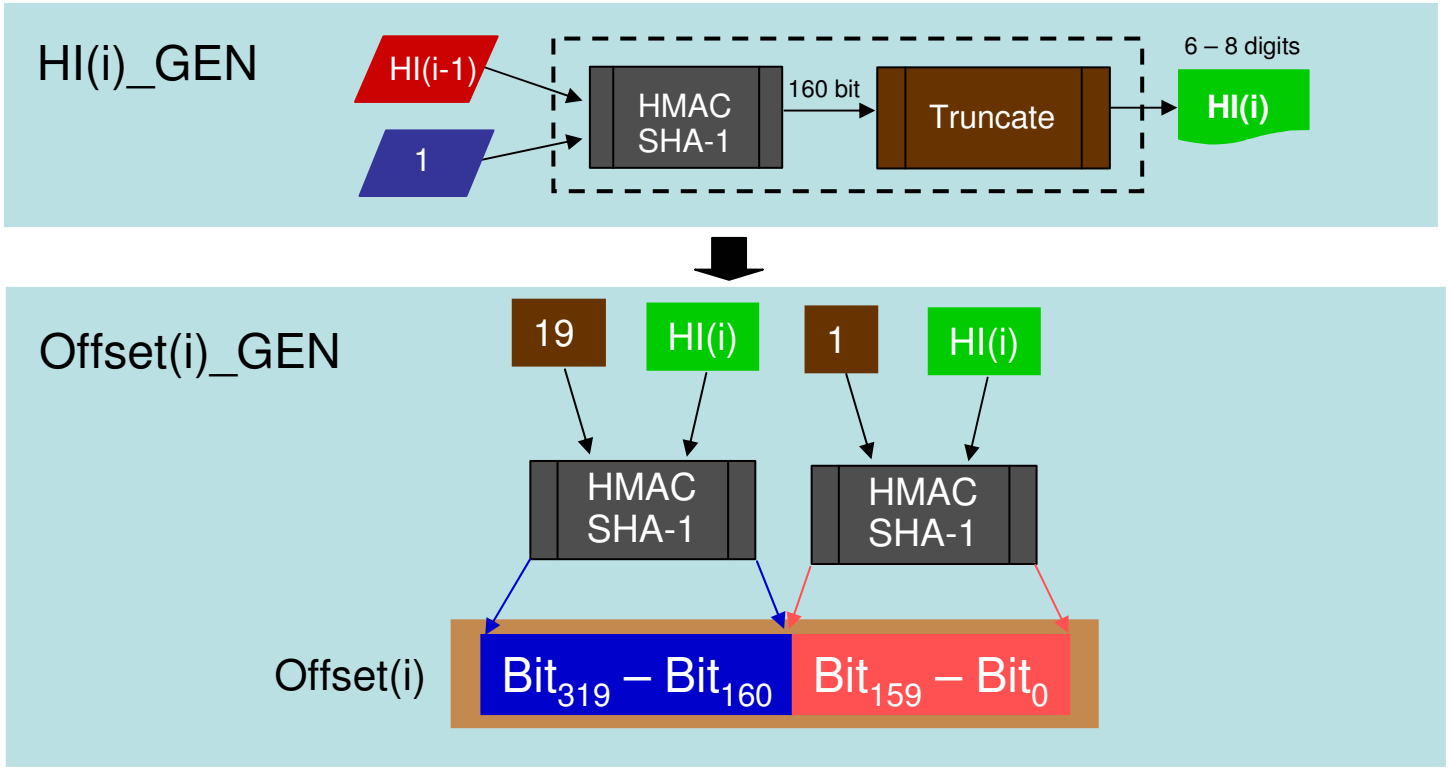
Key Locations	W_6	W_5	W_4	W_3	W_2	W_1	W_0
I	D_6	D_5	D_4	D_3	D_2	D_1	D_0
II	3	**8**	**0**	**7**	**2**	**3**	**5**

Final REAL Image X = DATA(i) = Concatenate ($X_{40} \sim W_i \sim X_1$)

$X_{40}\ X_{39}$ .. 3 .. X_k .. 8 .. X_j .. 0 .. X_i .. 7 .. X_h .. 2 .. X_g .. 3 .. X_f .. 5 .. $X_2\ X_1$

Offset & HI Generation (step 7~8)

- Offset is generated from the last Hashed Index (HI)
  - Reuse OATH OTP generation algorithm

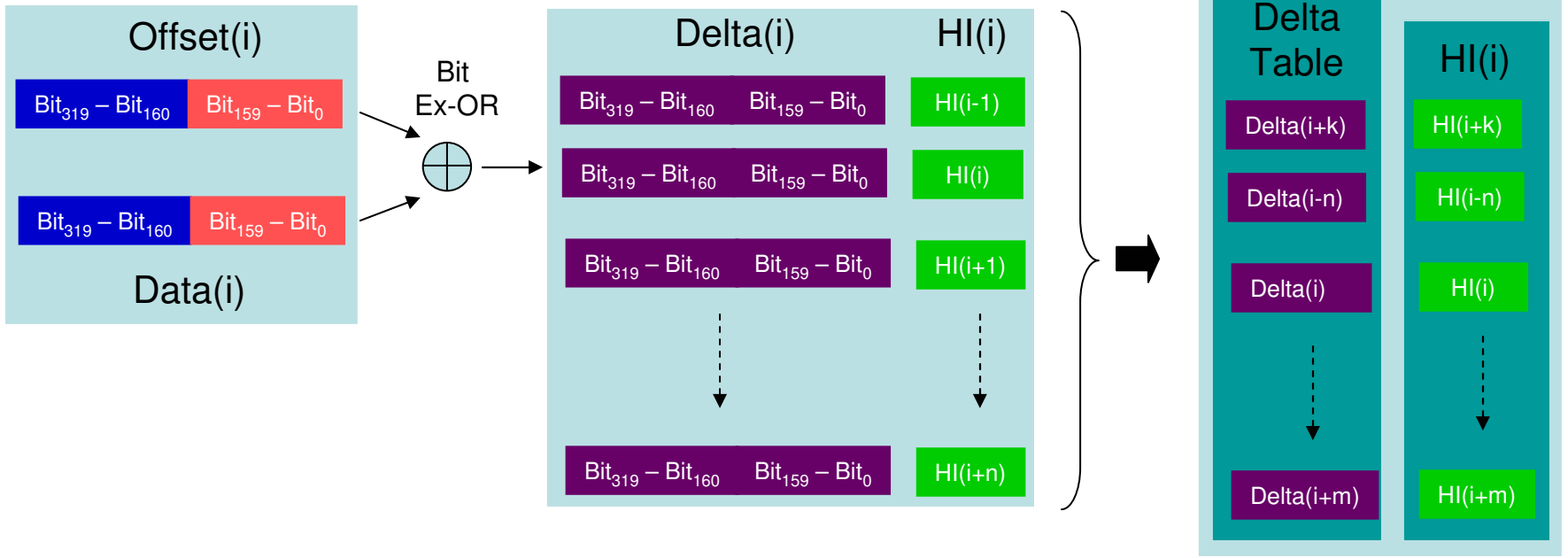


Delta Table Generation (Step 9~12)

- Delta(i) = Bit Ex-OR (Offset(i), Data(i))
- Delta Table (DT)
  - Compilation of Delta(i) with HI(i)
  - Rearranging Delta(i) order according to the value of HI(i)
  - Ensure higher security with the local storage
- User_Key = HMAC-SHA-1(HMAC-SHA-1(UC, UC)), UC)

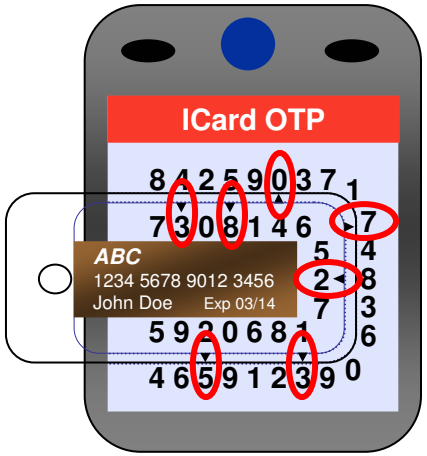
Stored in phone

Encryption with a User Key

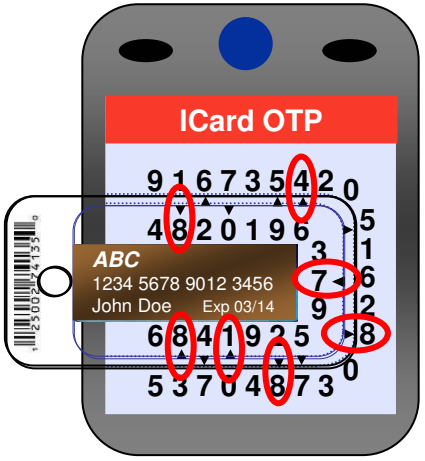


REAL Decryption (Step A~L)

- Reverse previous steps
- Place H/W token over the numerical image (beginning with token's front side 1st)
- Rubbing sequence
  - 1st Pointer indicates front (odd number) and backside (even number) key selection
  - Reading from 2nd pointer: Top/Down or Left/Right and clockwise



Code = **807235**



Code = **478818**

Security Attacks – Seed Tracing

- OTP code is generated by a known algorithm with a static SEED
- Collecting codes & comparing to code database
  - Finding Pseudo Random Sequence

$OTP_1, OTP_2, OTP_3, OTP_4, OTP_5, OTP_6, \ldots$ → **OTP Seed**

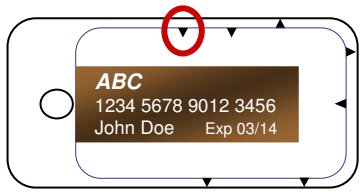
- Solution
  - Multi-Seeding OTP (Ms.OTP)
    - To break the Pseudo Random Sequence
    - Increase OTP code randomness

Seed A **Seed B**

Multiple Seeding OTP (Ms.OTP)

- A REAL Bi-Seeding Mobile OTP Token
  - One H/W token with two encryption keys
  - One REAL key to encrypt codes from one OTP Seed
  - Front 1st pointer to show which key to rub the OTP code
- Randomly mixing OTP codes from either Seed
  - Breaking pseudo random sequence from collected codes
  - Server records the mixing pattern during the provisioning

$OTP_{seed\text{-}A}$, $OTP_{seed\text{-}B}$, $OTP_{seed\text{-}A}$, $OTP_{seed\text{-}A}$, $OTP_{seed\text{-}B}$, $OTP_{seed\text{-}A}$, $OTP_{seed\text{-}B}$, …



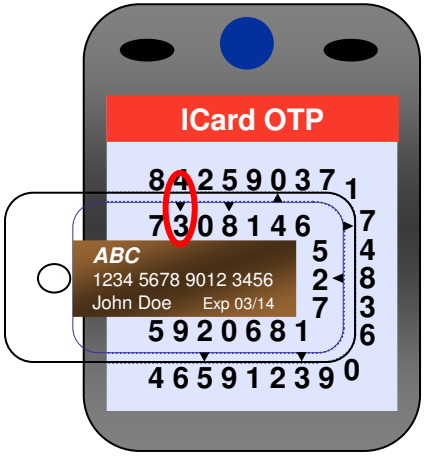
Front side to encrypt codes using Seed A

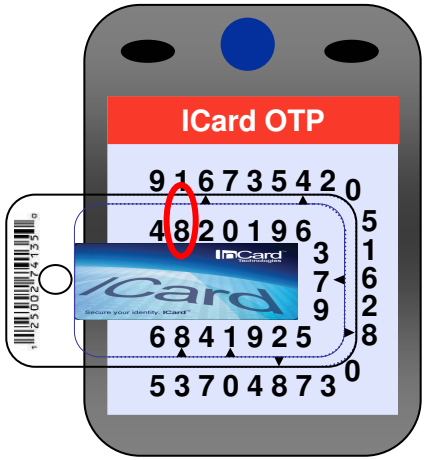
Back side to encrypt codes using Seed B

Multiple Seeding OTP (Ms.OTP)

	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Notes
A1	3	8	0	7	2	3	5	Seed A, Token front side
B1	6	4	7	8	8	1	8	Seed B, Token back side

OTP Code
= **807235**

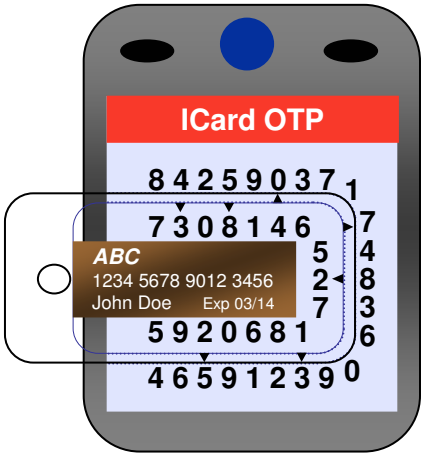




OTP Code
= **478818**

Security Attacks – Shoulder-surfing

- Secretly observing and collecting either OTP codes or token pointer locations
  - To trace OTP Seed or
  - To trace REAL encryption keys



OPT code = 807235

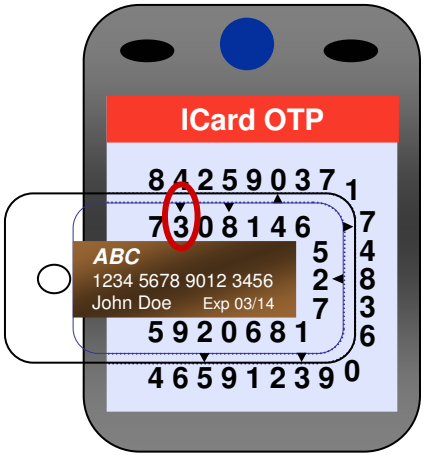
- Solution
  - Ms.OTP → Preventing OTP Seed tracing
  - Multi-Random OTP (Mr.OTP)
    - Preventing REAL key tracing
    - Breaking code to the pointer's physical locations

Multiple Random OTP (Mr.OTP)

- 1st pointer's code (N_{1f} or N_{1b}) value provides the seed for creating the extra randomness
- Each code value adds the 1st pointer's code value and drops the 10s digit if the sum is greater than 10

$$D_{if} = (\text{Value of } N_{1f} + \text{Value of } N_{(7-i)\,f}) \bmod 10$$

	D_6	D_5	D_4	D_3	D_2	D_1	D_0
I	3	8	0	7	2	3	5
II	3	11	3	10	5	6	8
III	3	1	3	0	5	6	8



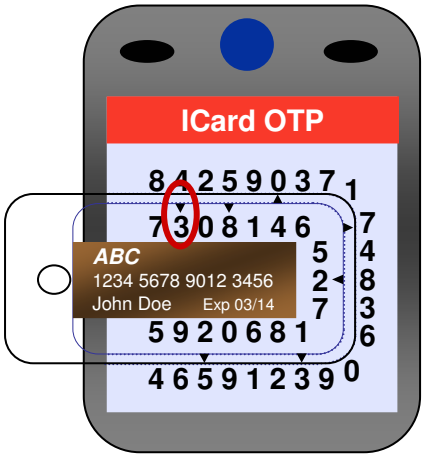
OTP Code = **130568**

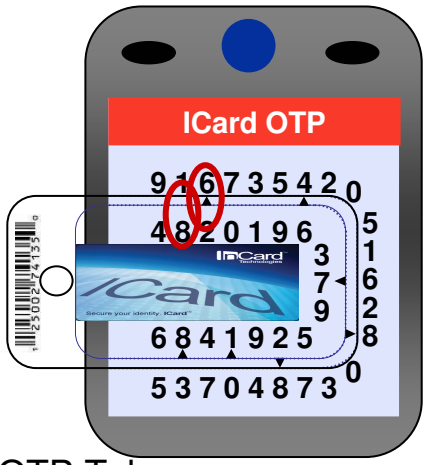
REAL Secure Mobile OTP Token

- Resists OTP Seed-tracing and Shoulder-surfing attacks
- REAL Secure Mobile OTP Token = Ms.OTP + Mr.OTP

	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Notes
A1	3	8	0	7	2	3	5	Seed A, front side, Ms.OTP
A2	3	1	3	0	5	6	8	OTP = Ms.OTP + Mr.OTP
B1	6	4	7	8	8	1	8	Seed B, back side, Ms.OTP
B2	6	0	3	4	4	7	4	OTP = Ms.OTP + Mr.OTP

OTP Code
= **130568**





OTP Code
= **034474**

Other Security Analysis

- Phone is lost or stolen
  - Have REAL Image w/o H/W token (if breaks the U_K 1st)

    Possibility (P_1) = 1/ C(40, 6) = 2.6 x 10^{-7}
  - Decrypt codes from Delta Table directly

    Needs user credential (U_K) & HI values
- H/W token is lost, stolen or secretly copied
  - No phone (REAL Image)

    Possibility (P_2) = 1 x 10^{-7}
- H/W token & DT are secretly copied

  Protected by user credential (U_K)
- Brute-force guess possibility = 1 x 10^{-6}

Conclusion

- Rubbing Encryption Algorithm (REAL)
  - A multi-dimensional secure cipher with long and complex keys
  - Provides high security level encryption for short length plaintext
- A REAL Secure Mobile OTP Token
  - Securely protects local stored confidential data
  - Resists MITM Seed-tracing and Shoulder-surfing attacks
  - Low cost, compatible & interoperable with existing authentication infrastructures
- Further Work
  - To explore more apps. on REAL multi-dimension, multi-key features and improve the usability against desired security level
  - Example:
    - "A Novel Rubbing Encryption Algorithm and The Implementation of a Web-based One-time Password Token", COMPSAC 2010. [23]

THANKS!

Q & A