Lecture 12.1: User-Enabled Device Authentication - I

CS 436/636/736
Spring 2012

Nitesh Saxena
The Problem: “Pairing”

Examples (single sensor setting):
- Communication between Alice’s and Bob’s devices when they have:
  - Pairing a Bluetooth cell phone with a headset
  - No prior context
  - Pairing a WiFi laptop with an access point
  - No common trusted CA or TTP
- Pairing two Bluetooth cell phones
PIN-based Bluetooth Pairing

### Initialization

- **User enters PIN**
  - $K_{init} = E_{ID_B}(PIN, L_{PIN}, IN_{RAND})$
- **User enters PIN**
  - $K_{init} = E_{ID_B}(PIN, L_{PIN}, IN_{RAND})$

### Link Key

**Alice**
- $L_K_A = E_{ID_A}(RAND_A)$
- $K_{sk} = (L_K_A \oplus L_K_B)$

**Bob**
- $L_K_B = E_{ID_B}(RAND_B)$
- $K_{sk} = (L_K_A \oplus L_K_B)$

**Generate**
- $RAND_A \oplus K_{init}
- RAND_B \oplus K_{init}$

**Generate**
- $K_{init} = E_{ID_A}(PIN, L_{PIN}, IN_{RAND})$
- $K_{init} = E_{ID_B}(PIN, L_{PIN}, IN_{RAND})$
Authentication

Alice

Bob

Authentication

$\text{SRES}_a \leftarrow E_k(\text{ID}_a, \text{RAND}_a)$

$\text{AU}_a \rightarrow \text{RAND}_a$

$\text{AU}_b \rightarrow \text{RAND}_b$

$\text{SRES}_b \leftarrow E_k(\text{ID}_b, \text{RAND}_b)$
PIN-based Bluetooth Pairing

Initialization

User enters PIN

\( K_{\text{init}} = E_{\text{PIN}}(\text{ID}_A, \text{PIN}, \text{IN}_{\text{RAND}}) \)

User enters PIN

\( K_{\text{init}} = E_{\text{PIN}}(\text{ID}_B, \text{PIN}, \text{IN}_{\text{RAND}}) \)

Link Key

\( LK_a = E_{\text{RAND}_a}(\text{ID}_A, \text{RAND}_a) \)

\( LK_b = E_{\text{RAND}_b}(\text{ID}_B, \text{RAND}_b) \)

Generate \( \text{RAND}_a \)

\( \text{RAND}_a \oplus K_{\text{init}} \)

Generate \( \text{RAND}_b \)

\( \text{RAND}_b \oplus K_{\text{init}} \)

\( LK_a = E_{\text{RAND}_a}(\text{ID}_A, \text{RAND}_a) \)

\( LK_b = E_{\text{RAND}_b}(\text{ID}_B, \text{RAND}_b) \)

\( K_{\text{sek}} = (LK_a \oplus LK_b) \)
Authentication

Alice

SRES = E(ID_A, K_{mv}, AU RAND_A)

SRES = E(ID_A, K_{mv}, AU RAND_B)

AU RAND_A

AU RAND_B

Bob
(In)Security of PIN-based Pairing

- Long believed to be insecure for short PINs
  - Why?
- First to demonstrate this insecurity; *Shaked and Wool* [Mobisys’05]
Attack Implementation

- Coded in C on linux platform
  - Given a piece of code for SAFER algorithm, implemented the encryption functions $E_{22}, E_{21}, E_1$
- Hardware for sniffing: bluetooth packet analyzer with windows software
- Log Parser (in perl): reads the sniffer log, and parses it to grab IN_RAND, RAND_A \(\text{xor} \) $K_{\text{init}}$, RAND_B \(\text{xor} \) $K_{\text{init}}$, AU_RAND_A, AU_RAND_B, SRES
Timing Measurements of Attack

- Theoretically: $O(10^L)$, with decimal digits
  - Assuming the PINs are chosen uniformly at random
- Empirically, on a PIII 700MHz machine:

<table>
<thead>
<tr>
<th>No. of digits in PIN (L)</th>
<th>CPU time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1.294</td>
</tr>
<tr>
<td>5</td>
<td>12.915</td>
</tr>
<tr>
<td>6</td>
<td>129.657</td>
</tr>
<tr>
<td>7</td>
<td>1315.332</td>
</tr>
</tbody>
</table>
Timing of Attack and User Issues

- **ASCII PINs**: $O(90^L)$, assuming there are 90 ascii characters that can be typed on a mobile phone
  - Assuming random PINs
- However, in practice the actual space will be quite small
  - Users choose weak PINs;
  - Users find it hard to type in ascii characters on mobile devices
- Another problem: shoulder surfing (manual or automated)
The Problem: “Pairing”

Idea
- make use of a physical channel between devices
  - Also know as out-of-band (OOB) channel
  - with least involvement from Alice and Bob

Authenticated: Audio, Visual, Tactile
**Seeing-is-Believing** (McCune et al. [Oakland'05])

- Protocol (Balfanz, et al. [NDSS'02])

- Insecure Channel
  - $pk_A$
  - $pk_B$
  - $H(pk_A)$
  - $H(pk_B)$

- Authenticated Channel

- Secure if
  - $H(.)$ weak CR
  - 80-bit for permanent
  - 48-bit for ephemeral

Rohs, Gfeller
[PervComp'04]
Challenges

- OOB channels are low-bandwidth!
- One of the device might not have a keypad/receiver!
- Neither has a receiver and only one has a good quality transmitter
  - (Non-)Universality!
- Usability Evaluation
- Multiple devices – scalability!
Challenges

- OOB channels are low-bandwidth!
Challenges

- One of the devices might not have a keypad/receiver
  - e.g., keyboard-desktop; AP-phone
Unidirectional SAS (Saxena et al. [S&P'06])

- Blinking-Lights

\[ p_k_a, H(R_A) \]
\[ p_k_b, R_B \]
\[ R_A \]
\[ hs(R_A, R_B; p_k_a, p_k_b) \]

Insecure Channel
Authenticated Channel
User I/O
Secure (with prob. 1-2^{-15}) if
- 15-bit AU hs()

Multiple Blinking LEDs (Saxena-Uddin [MWNS'06])
Challenges

- Neither has a receiver and only one has a display
  - e.g., AP-laptop/PDA
A Universal Pairing Method

- Prasad-Saxena [ACNS’08]
- Use existing SAS protocols
- The strings transmitted by both devices over physical channel should be
  - the same, if everything is fine
  - different, if there is an attack/fault
- Both devices encode these strings using a pattern of
  - Synchronized beeping/blink
  - The user acts as a reader and verifies if the two patterns are same or not
Is This Usable?

- Our test results are promising
  - Users can verify both good test cases and bad ones
- Blink-Blink the easiest
  - Very low errors (less than 5%)
  - Execution time ~22s
- Then, Beep-Blink
  - Very low errors with a learning instance (less than 5%)
  - Execution time ~15s
- Beep-Beep turns out error-prone
Further Improvement: Auxiliary Device

- Saxena et al. [SOUPS’08]

- Auxiliary device needs a camera and/or microphone – a smart phone
- Does not need to be trusted with cryptographic data
- Does not need to communicate with the devices
Further Improvement: Auxiliary Device

- **Blink-Blink**
  - ~14s (compared to 22s of manual scheme)

- **Beep-Blink**
  - Approximately takes as long as the same as manual scheme
  - No learning needed

- In both cases,
  - Fatal errors (non-matching instances decided as matching) are eliminated
  - Safe errors (matching instances decided as non-matching) are reduced

- It was preferred by most users
Many Other Methods Exist

- We will look at them next time
- And, analyze and compare them from the usability and security standpoint
References

- Cracking Bluetooth PINs: 
  http://www.eng.tau.ac.il/~yash/shaked-wool-mobisys05/

- OOB channel related references:
  - Many on my publications page