# Automated PIN Cracking

**Justin Engler**

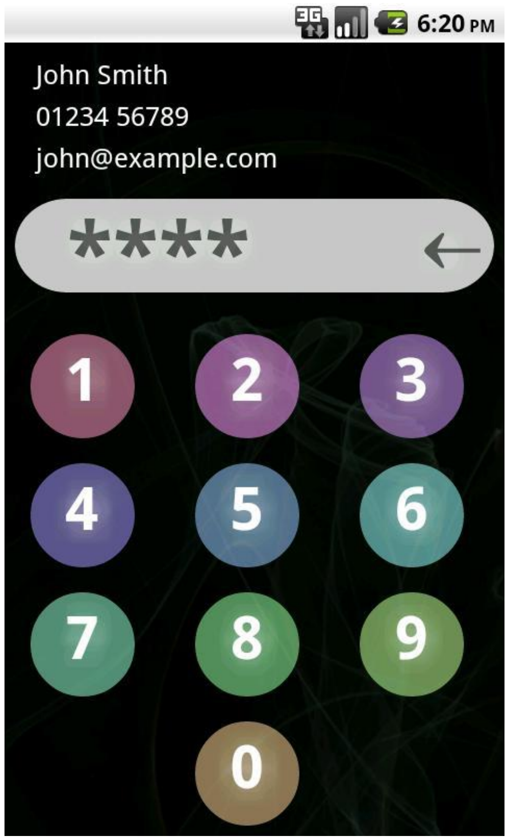
**Paul Vines**

## Agenda

- Current PIN Cracking Methods
- Cracking with Robots
- R2B2
- C3BO
- Defeating the Robots

## PINs

- One of the most popular ways to lock mobile devices
  - Commonly still only 4-digit despite ability to be longer
  - User chosen, so typically low-entropy



play.google.com

## PIN Cracking Now

- Jailbreak and Crack
- Keyboard Emulation
- Brute Force the UI

## PIN Cracking Now

- Jailbreak and Crack
- Keyboard Emulation
- Brute Force the UI
- Punish an Intern

## Punish an Intern

- Forcing your intern to try all 10,000 4-digit combinations will surely be more productive than anything else they could have been doing, except maybe getting coffee
- **Problem:** Interns are universally bad at their jobs, so they might miss some of the combinations

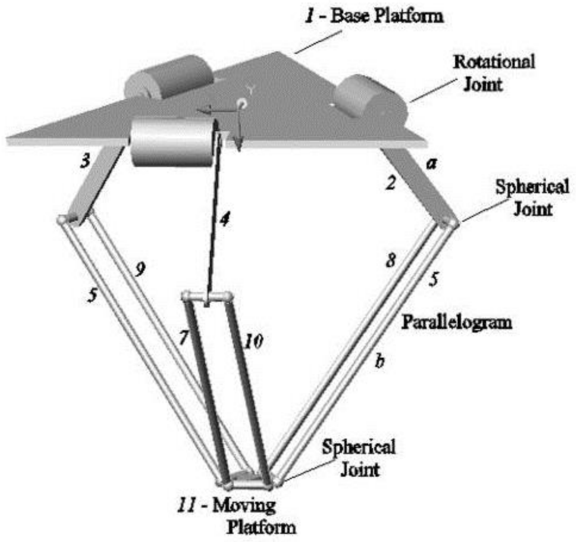
## PIN Cracking with Robots

- Required Abilities:
  - “Push” buttons in sequence
  - Remember what buttons were pushed
  - Recognize success
    - Not always necessary

## Delta Robot

- Designed for fast precision industrial work
- Simple combination of 3 single-motor arms gives precision 3D movement with somewhat small range of motion
- Fairly simple motion control

*



* **Lopez, Castillo, Garcia, and Bashir.** *Delta robot: inverse, direct, and intermediate Jacobians.* Proc. IMechE Vol.220(2006)

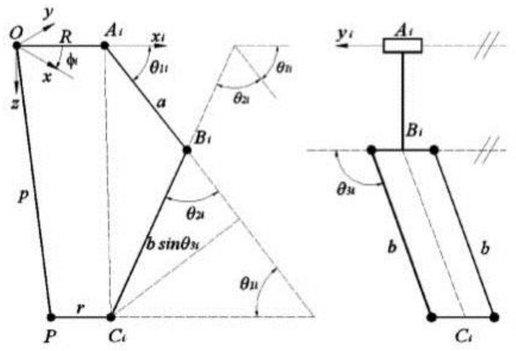
# Still a lot of maths…

$$p_x = \frac{f_1 - e_1 - e_3[e_2 f_2 - e_2 e_4 - e_5 f_1 + e_1 e_5 / e_2 e_6 - e_3 e_5]}{e_2},$$

$$p_x = \frac{e_2 f_2 - e_2 e_4 - e_5 f_1 + e_1 e_5}{e_2 e_6 - e_3 e_5},$$

$$p_z = [e_8 - p_x^2 - p_y^2 + 2k_3 p_x - 2s_3 p_y]^{1/2} \qquad (29)$$

$$\begin{aligned}\hat{b}_i \cdot \vec{v} = {} & [\sin\theta_{3i}\cos(\theta_{2i}+\theta_{1i})][v_x\cos\phi_i - v_y\sin\phi_i] \\ & + \cos\theta_{3i}[v_x\sin\phi_i + v_y\cos\phi_i] \\ & + [\sin\theta_{3i}\sin(\theta_{2i}+\theta_{1i})]v_z = J_{ix}v_x \\ & + J_{iy}v_y + J_{iz}v_z\end{aligned} \qquad (9)$$



$$k_i = (R-r)\cos\phi_i,\quad s_i = (R-r)\sin\phi_i,\quad i = 1,2,3$$

$$e_1 = k_3^2 - k_1^2 + s_3^2 - s_1^2,\quad e_2 = 2k_1 - 2k_3$$

$$e_3 = 2s_3 - 2s_1,\quad e_4 = k_3^2 - k_2^2 + s_3^2 - s_2^2$$

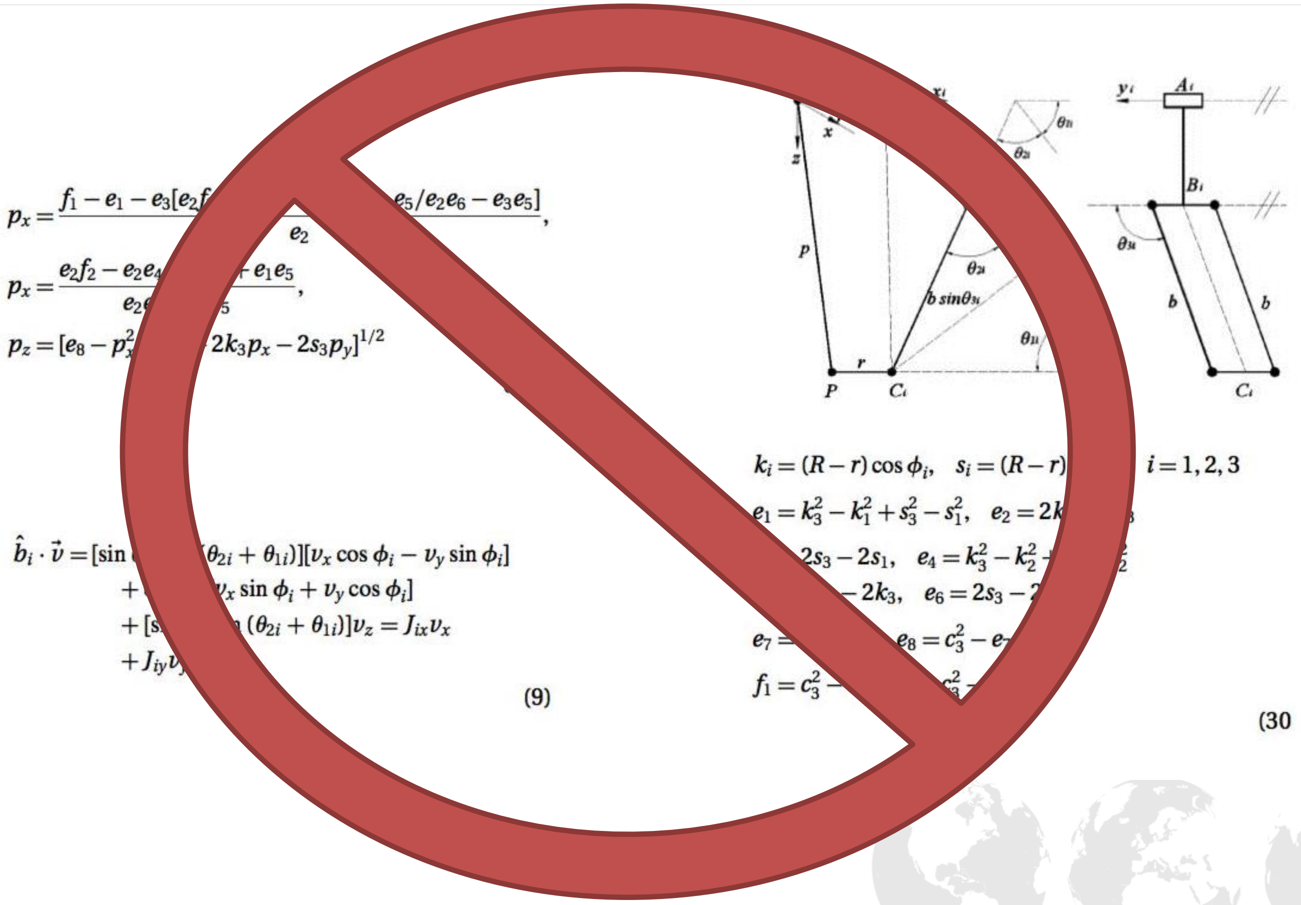
$$e_5 = 2k_2 - 2k_3,\quad e_6 = 2s_3 - 2s_2$$

$$e_7 = k_3^2 + s_3^2,\quad e_8 = c_3^2 - e_7$$

$$f_1 = c_3^2 - c_1^2,\quad f_2 = c_3^2 - c_2^2$$

(30

# Still a lot of maths…



$p_x = \frac{f_1 - e_1 - e_3[e_2 f}{}$ … $e_5/e_2e_6 - e_3e_5]$ … $e_2$,
$p_x = \frac{e_2 f_2 - e_2 e_4}{e_2 e}$ … $+ e_1 e_5$ … $5$,
$p_z = [e_8 - p_x^2$ … $2k_3 p_x - 2s_3 p_y]^{1/2}$
$\hat{b}_i \cdot \vec{v} = [\sin$ … $(\theta_{2i} + \theta_{1i})][v_x \cos\phi_i - v_y \sin\phi_i]$
$+$ … $v_x \sin\phi_i + v_y \cos\phi_i]$
$+ [s$ … $(\theta_{2i} + \theta_{1i})]v_z = J_{ix} v_x$
$+ J_{iy} v$
(9)
$k_i = (R - r)\cos\phi_i, \quad s_i = (R - r)$ … $i = 1, 2, 3$
$e_1 = k_3^2 - k_1^2 + s_3^2 - s_1^2, \quad e_2 = 2k$
$2s_3 - 2s_1, \quad e_4 = k_3^2 - k_2^2 +$
$-2k_3, \quad e_6 = 2s_3 - 2$
$e_7 =$ … $e_8 = c_3^2 - e$
$f_1 = c_3^2 -$ … $c_3^2$
(30
$x_i$
$x$
$z$
$\theta_{1i}$
$\theta_{2i}$
$p$
$\theta_{2i}$
$b \sin\theta_{3i}$
$\theta_{1i}$
$r$
$P$
$C_i$
$y_i$
$A_i$
$B_i$
$\theta_{3i}$
$b$
$b$
$C_i$

## Robotic Reconfigurable Button Basher

- Arduino microcontroller
- 3-d printed parts
- Open source code and design for a delta robot by Dan Royer (marginallyclever.com)
  - Uses serial port communication to control the movement of the robot
- Available as a kit, or DIY

## Modifications

- The original delta robot kit was modified to have its tool be a touch-screen stylus tip for pressing buttons
  - Important: Stylus tip needs to be grounded
- A camera was added to allow easier user interface with the robot to set up the PIN cracking task
  - And recognize when the device is unlocked!
- The motion control software was modified to speed up movement, up to 5 presses/second

## Wrap Everything in Python

- Controls the robot movement through the serial port
- Performs image analysis of the camera feed
- Provides a simple interface for the user to set the robot up for PIN cracking
- Detects success of PIN cracking to stop robot and alert user

## Capacitive Cartesian Coordinate Bruteforcing Overlay (C3BO)

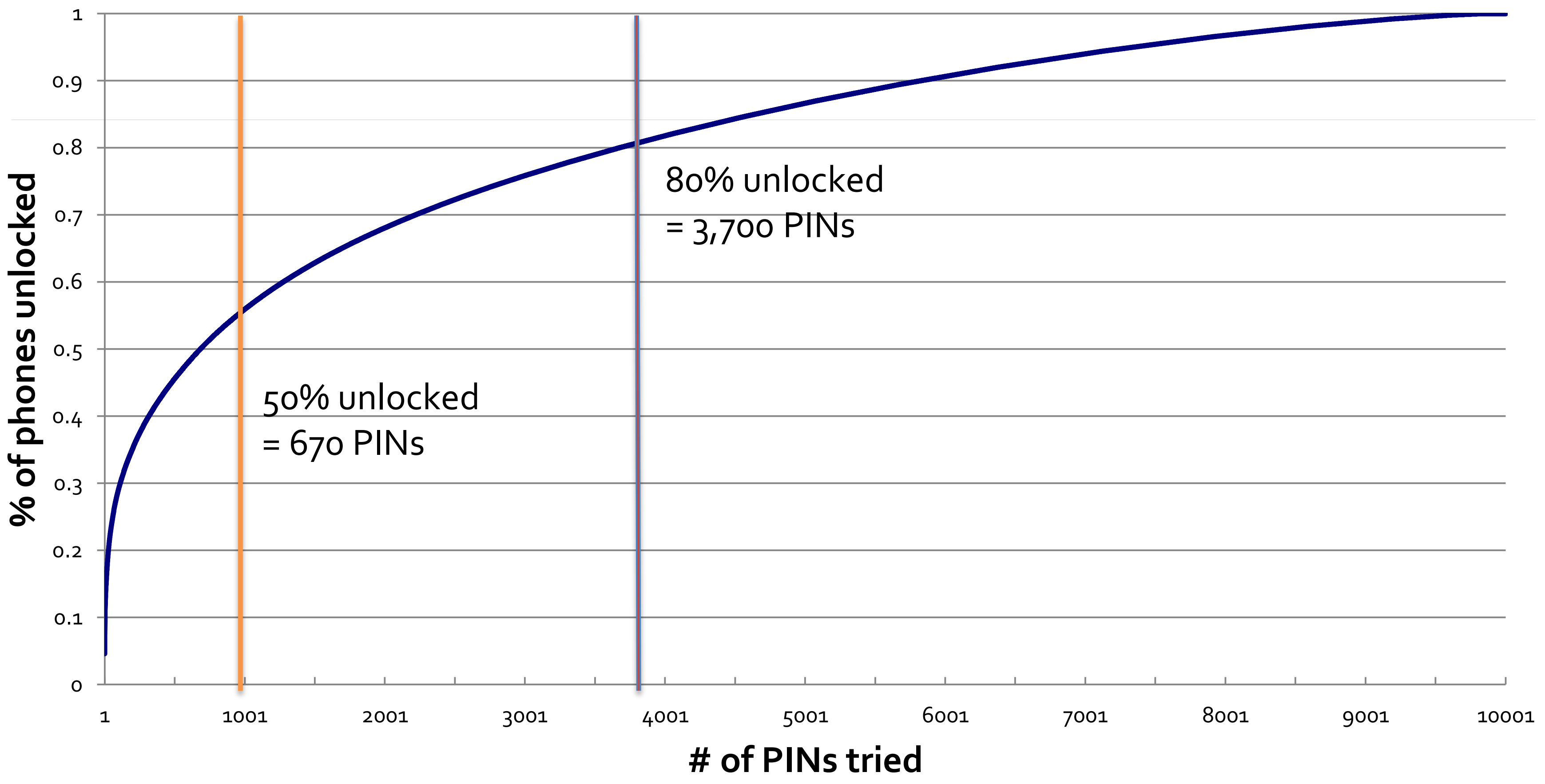
- Attach a grid of electrodes to the device’s virtual keyboard
- Trigger electrodes via an Arduino to trick the device into thinking the screen was touched at that point
- Faster, Better, Cheaper (~$50)

## C3BO :(

- Doesn’t work.

## (a bit) Better than brute-forcing

- Harvested 4-digit sequences from online password lists
  - (eharmony, myspace, etc.)
  - Presumably what Nick Berry did for his blog but wouldn't share…
- Combined with Daniel Amitay's (danielamitay.com) phone app PIN list


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
% of phones unlocked
1
1001
2001
3001
4001
5001
6001
7001
8001
9001
10001
# of PINs tried
80% unlocked
= 3,700 PINs
50% unlocked
= 670 PINs

## Real Buttons Too!

- R2B2 can of course also be used for brute-force PIN cracking of physical buttons as well
- Electronic keypads or completely mechanical keys, provided it can detect when it has succeeded

## Defeating the Robots

- Forced delay timer after X attempts
  - On Android this is 30 seconds regardless of previous attempts
    - R2B2 would succeed in a worst case of ~20 hours
    - Likely success much sooner (80 mins =50%, 7 hrs =80%)
- User Lockout after X attempts
  - On iOS, 1 minute lockout after 5 guesses
    - Lockout time quickly scales up for continued bad guesses (1 minute, 5 minutes, 15 minutes, 60 minutes)
    - Roughly 20% success rate on a 20 hour run

## Robots > Apps

- Lots of apps to replace lock screen or provide additional “protection” to elements of the phone (media storage etc.)
- Tried 13:
  - 4 had lockouts of >= 5 minutes/5 attempts
  - 9 had no lockout at all

## Defeating the Robots For Users

| **PIN character set and length** | 1 PIN per second | 1 PIN per second, plus 30 seconds every 5 guesses |
|---|---|---|
| 3 Digits | 16 Minutes | 117 Minutes |
| 4 Digits | 167 Minutes | 19.4 Hours |
| 5 Digits | 27 Hours | 8.1 Days |
| 6 Digits | 11.8 Days | 81 Days |
| 4 Lowercase + Digits | 19.4 Days | 136 Days |
| 7 Lowercase + Digits | 2484 Years | 7.83e10 Centuries |
| 4 Printable ASCII (94) | 2.48 Years | 7.81e7 Centuries |
| 7 Printable ASCII (94) | 20563 Centuries | 6.48e13 Centuries |

## Acknowledgments

- Thanks to iSEC Partners and the NCC Group for supporting this research
- Thanks to Dan Royer for providing the motion control code and robot build plans
- Thanks to Daniel Amitay for parts of our PIN data
- Thanks to David Nichols for analyzing the PIN using apps

## Contact Information

- Justin Engler
  - @justinengler
- Paul Vines
  - plvines@uw.edu