Automated PIN Cracking

Justin Engler

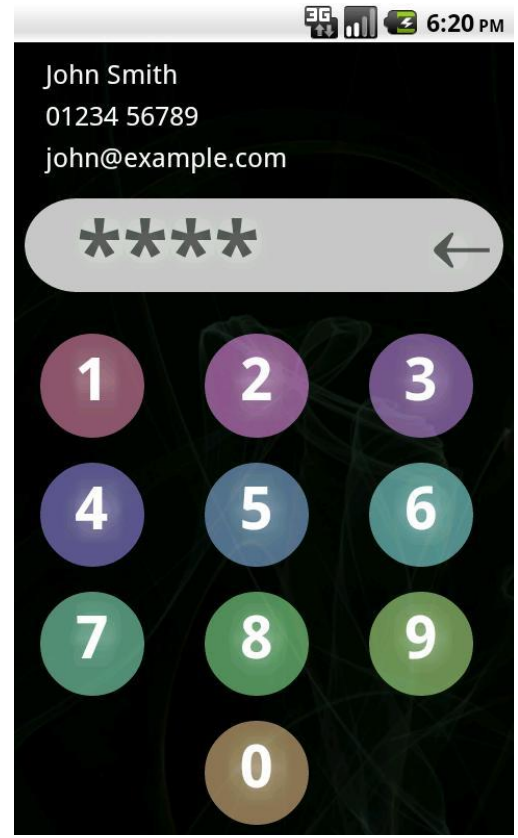
Paul Vines

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
- Punish an Intern

Jailbreak and Crack

- Use jailbreaking/rooting exploits on the device
- Bypass the lock screen with these new user capabilities
- **Problem:** not all devices have known exploits for gaining root (and without wiping the device)

Keyboard Emulation

- If the device supports a keyboard attachment
  - Make a device that emulates a keyboard and tries all the different PIN combinations automatically
- **Problem:** not all devices support an external keyboard being added

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

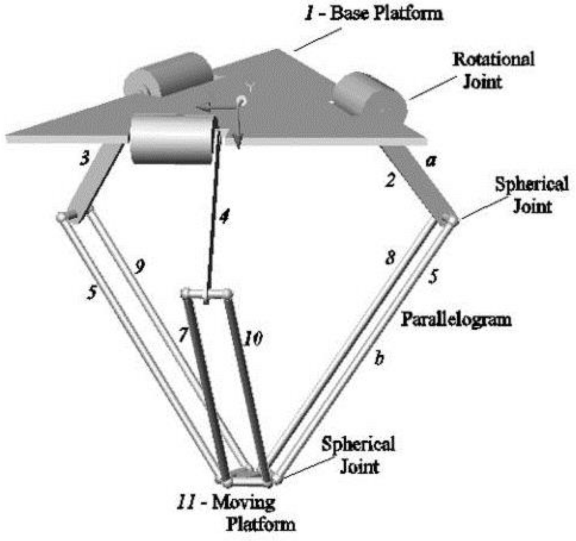
- Required Abilities:
  - “Push” buttons in sequence
  - Remember what buttons were pushed
  - Recognize success
    - Not always necessary

Robotic Reconfigurable Button Basher (R2B2)

- Homemade Delta Robot body
- Arduino Uno brain
- Total cost: < $200

Delta Robot

*

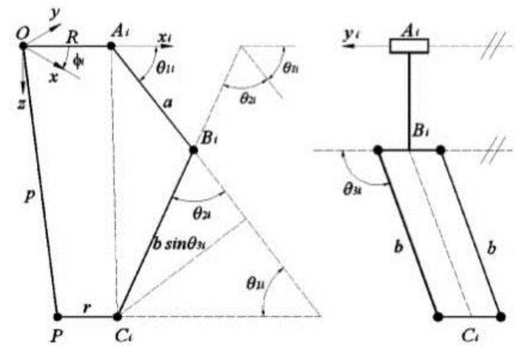


- Designed for fast precision industrial work
- Simple combination of 3 single-motor arms gives precision 3D movement with somewhat small range of motion
- Fairly simple motion control

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

“Fairly Simple” Motion Control

Still a lot of math…



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

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

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

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

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

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

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

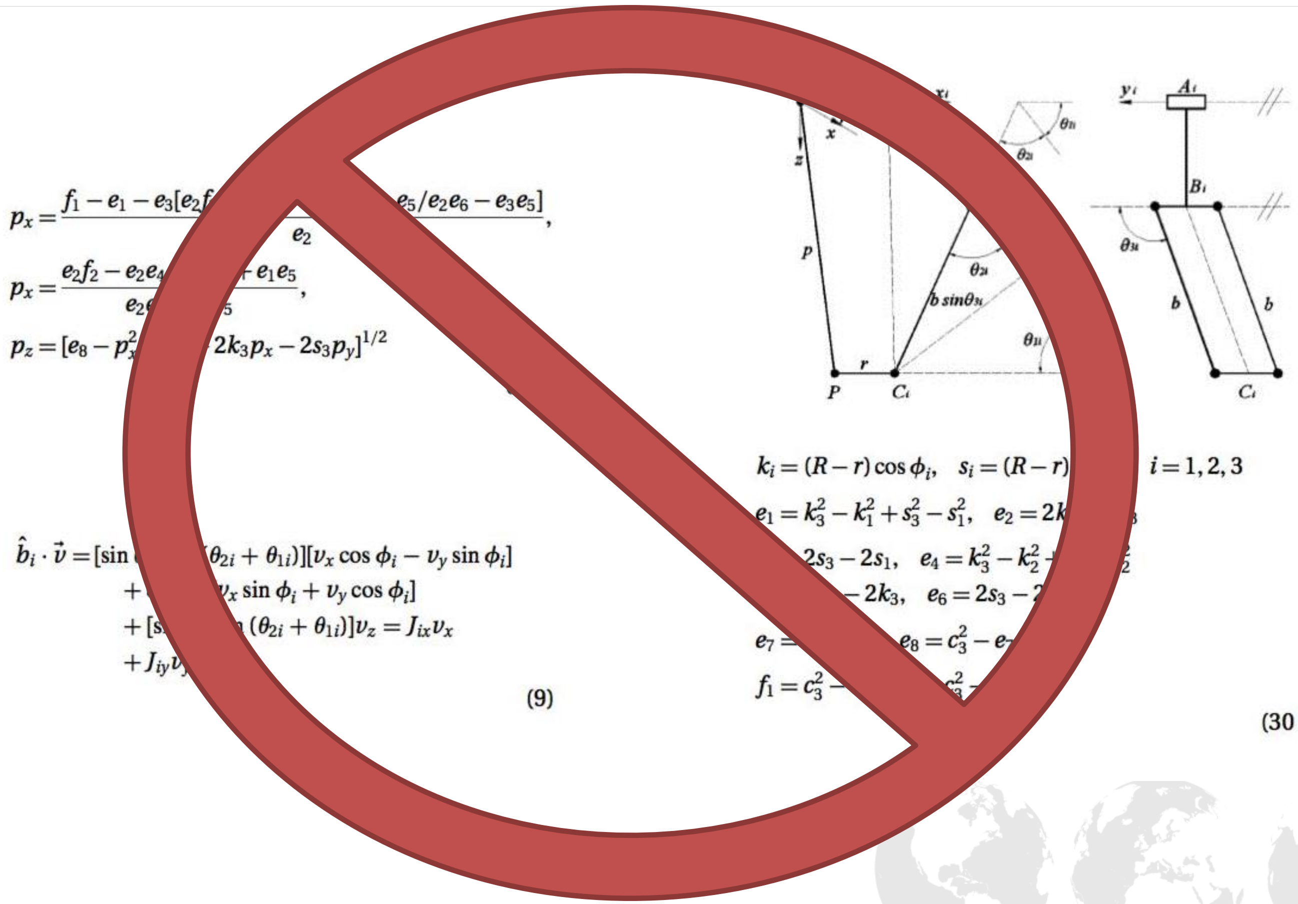
$$e_7 = k_3^2 + s_3^2, \quad e_8 = c_3^2 - e_7$$

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

(30

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

So we found someone else's code to do it



Arduino Uno

- Standard robotic hobby microcontroller board
- Open source code for controlling a delta robot by Dan Royer (marginallyclever.com)
  - Uses serial port communication to control the movement of the robot
- Easy to tweak functionality for pressing buttons instead of manufacturing
- Easy to control with a Python program

Modifications

- The original delta robot kit was modified to have its tool be a touch-screen stylus tip for pressing buttons
- 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)

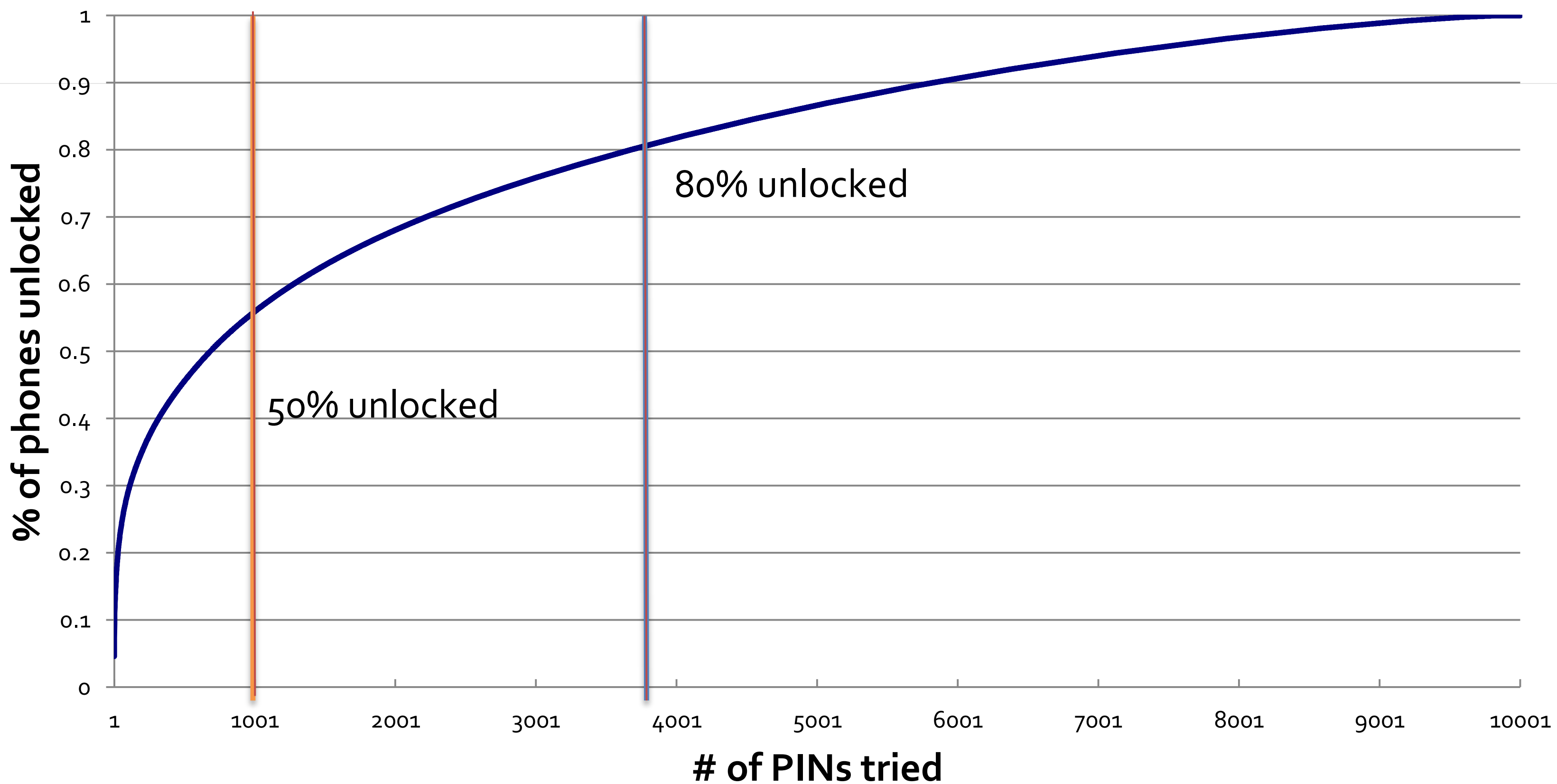
- Attach a grid of electrodes to the device's virtual keyboard
- Trigger electrodes via an Arduino to trick the device into thinking the screen was touched at that point
- No mechanical motion = faster button pressing
- More user configuration required to manually place the electrodes

C3BO continued

- Cheaper than R2B2 (~ $50)
- Nearly the same software for controlling/detecting device state changes with camera

(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
- And we get...



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

Challenges

- Detecting button values:
  - Too tough to reliably do on all devices
  - User set up time is negligible for a 10-digit keypad
- Recognizing delays:
  - Some devices have more easily recognized delay messages than others
  - If necessary, the user can manually input the delay pattern of a device (i.e. 30 seconds every 5 tries)

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

Are these robots useful, then?

- Compared to R2B2:
  - Jailbreak + Bypass: Best if available
  - Keyboard Emulator: The fastest brute-forcing
  - C3BO: Usable on any capacitive touch keyboard, a bit slower and more setup required than a keyboard emulator
  - R2B2: Flexible and usable on basically any PIN protected device but slower and more cumbersome

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