T2Pair: Secure and Usable Pairing for Heterogeneous IoT Devices

Xiaopeng Li, **Qiang Zeng**, Lannan Luo, Tongbo Luo

UNIVERSITY OF SOUTH CAROLINA

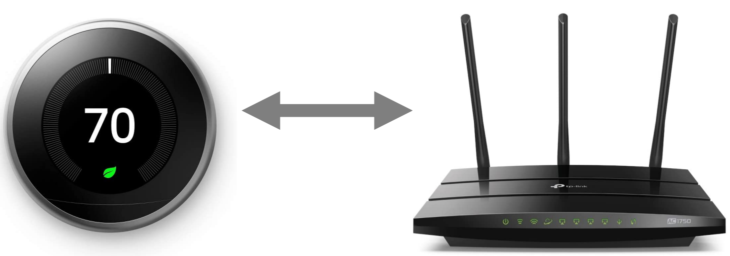
JD.COM

CCS 2020

IoT Pairing

- Pairing is supposed to establish a secure communication channel

- IoT pairing is important for
  - adding a new IoT device to a network
  - data transmission between two devices (e.g., a blood-pressure meter in Walmart and your phone)



Design Requirements

- **Secure**: resilient to co-located malicious devices

- **Usable** for heterogeneous IoT devices
  - No conventional UIs like keyboards
  - Not special sensors (e.g., inertial)

Existing Approaches

- Proximity-based
  - Move2Auth [InfoCom'17]: wireless signal features
  - Perceptio [S&P'19]: ambient context

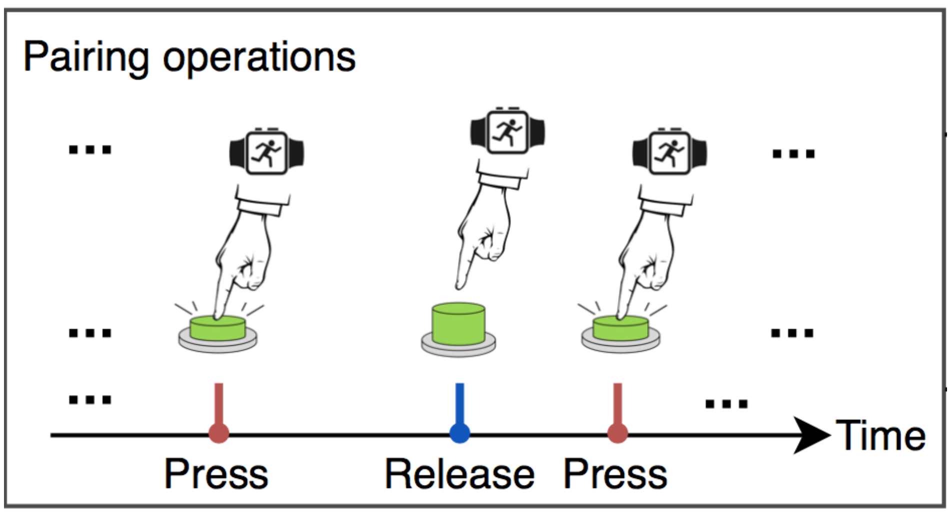
Insecure: exploited by co-located attackers

- Physical contact-based
  - ShaVe/ShaCK [TMC'09]: shake two devices together
  - H2H [CCS'13]: measure heartbeat data

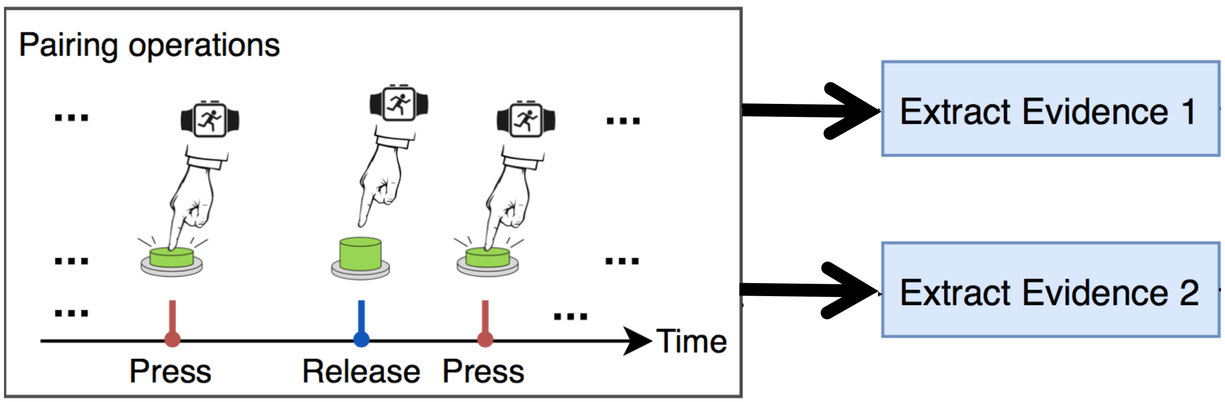
More secure but needs special **hardware/sensors**

Our Insights

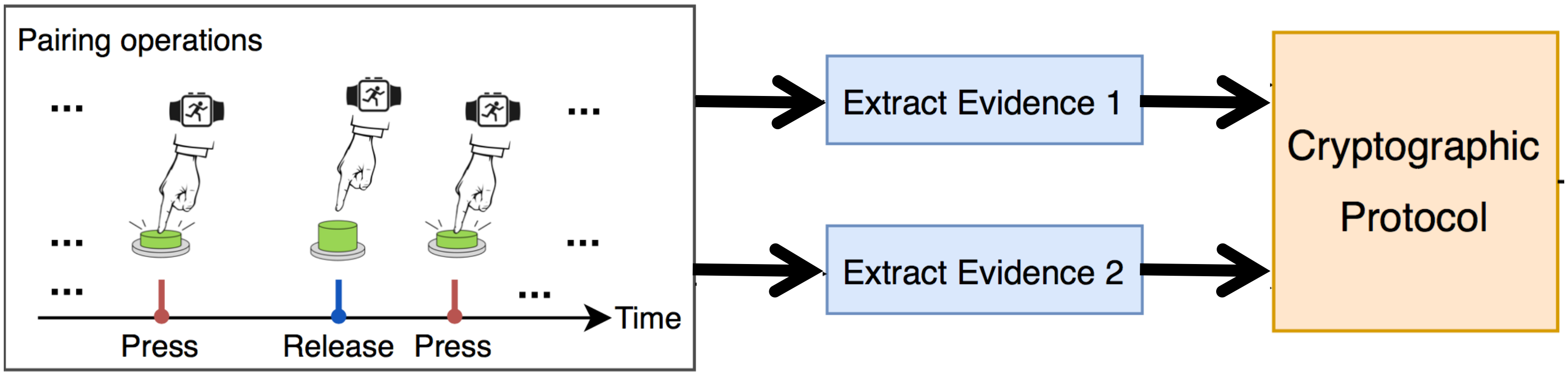
- Most IoT devices (>92%) have a button, knob, and/or small touchscreen
- Given a user wearing a smartwatch, when she presses a button of an IoT device, both the IoT device and the smartwatch can sense the operation
- Both sides have clocks: **timestamps as evidence**



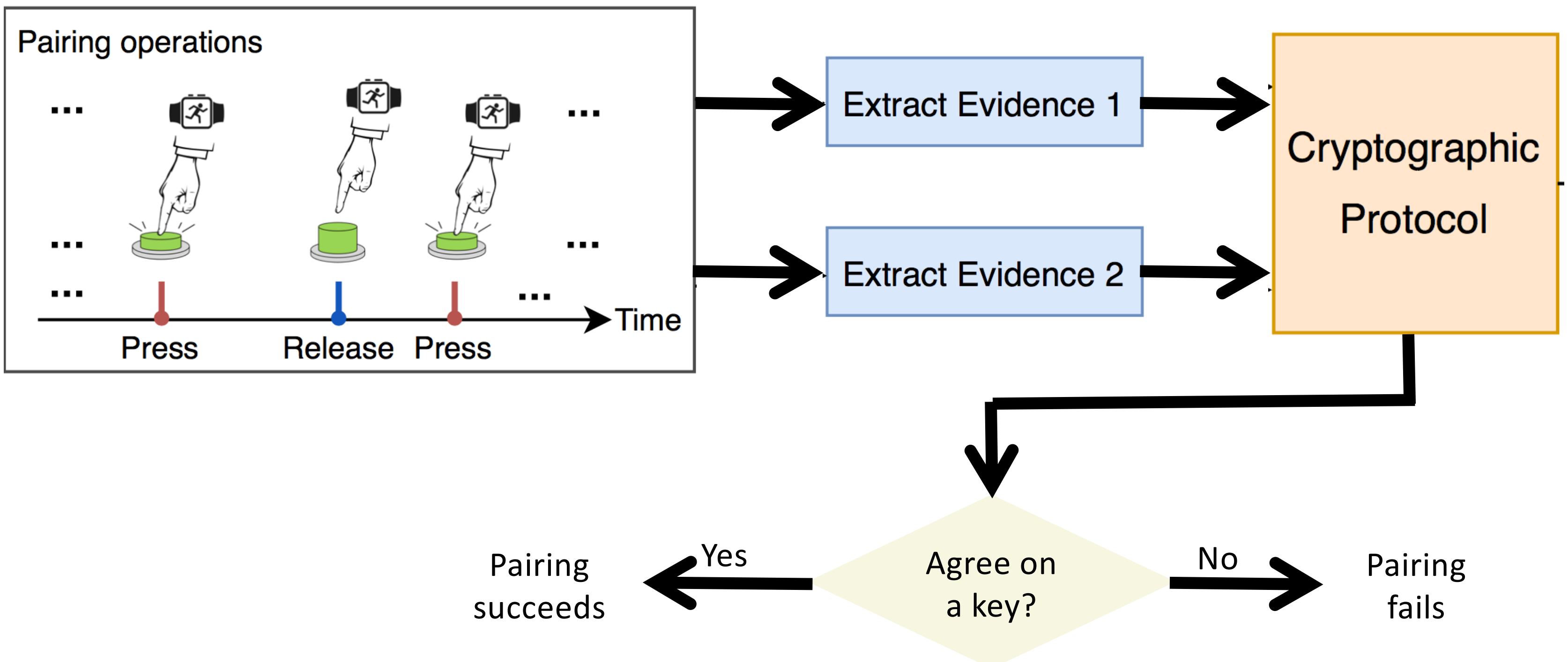
T2Pair: System Architecture



T2Pair: System Architecture

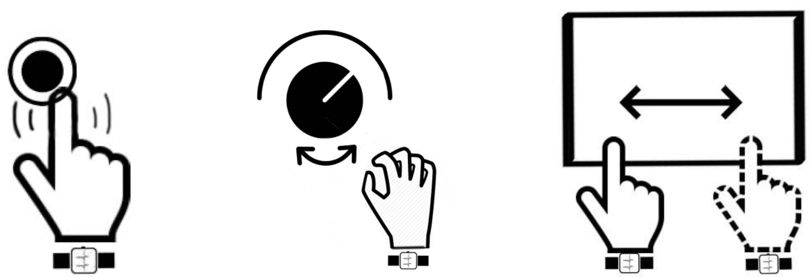


T2Pair: System Architecture



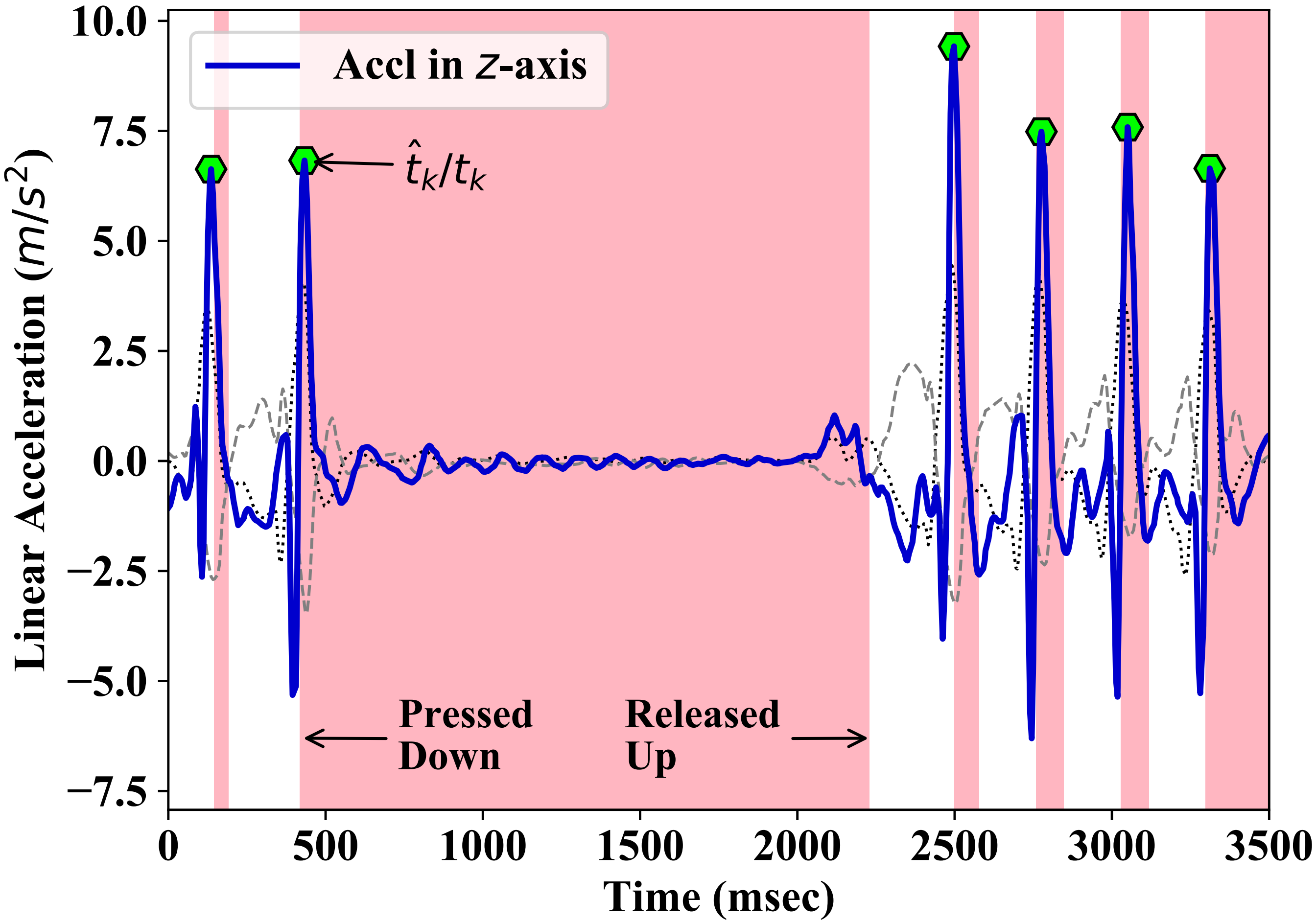
Pairing Operations

- Pressing the button a few times
- Twisting the knob back and forth
- Zig-zag swiping on the touchscreen



Sensing Physical Operations

- Correlation between button events and IMU data



Threat Model and Countermeasures

- Mimicry attacks: an adversary mimics a user to press a device to pair it with the user's smartwatch
  - Countermeasure: random pauses (enforced automatically)
- Man-in-the-Middle attacks
  - Countermeasure: **faithful** fuzzy commitment
  - Why fuzzy commitment?
    - two pieces of evidence are similar but not identical
- Online brute-force attacks
  - Countermeasure: Zero-knowledge password proof
- Offline brute-force attacks
  - Countermeasure: Diffie-Hellman Encrypted Key Exchange

Pairing Protocol

Device d_1		Device d_2
	Phase 1: Initialization	
	Initiates the pairing	
	Phase 2: Extracting Evidence	
$E_{d_1} = Time\_Int\_Seq(d_1)$		$E_{d_2} = Time\_Int\_Seq(d_2)$
if self-checking fails, aborts		if self-checking fails, aborts and reminds the user
	Phase 3: Fuzzy Commitment	
① picks a random value $P \in \mathbb{F}_{2^k}^m$		
② $\lambda \in \mathbb{F}_{2^k}^n \xleftarrow{\text{encode}} \mathrm{RS}(2^k, m, n, P)$		
③ commits: $\delta = e(E_{d_1}) \oplus \lambda$	$\xrightarrow{\delta}$	④ decommits: $\lambda' = e(E_{d_2}) \oplus \delta$
		⑤ $P' \xleftarrow{\text{decode}} \overline{\mathrm{RS}}(2^k, m, n, \lambda')$
	Phase 4: PAKE	
⑥ picks a; $A = g^a \bmod p$; $w = \mathrm{h}(P)$	$\xrightarrow{\mathrm{E}(w, A)}$	⑦ picks b; $B = g^b \bmod p$; $w' = \mathrm{h}(P')$
⑨ $K = B^a \bmod p$	$\xleftarrow{\mathrm{E}(w', B \Vert C_1)}$	⑧ $K' = A^b \bmod p$; picks a challenge C_1
⑩ picks a challenge C_2	$\xrightarrow{\mathrm{E}(K, C_1 \Vert C_2)}$	⑪ if C_1 is not received, aborts
⑫ if C_2 is not received, aborts	$\xleftarrow{\mathrm{E}(K', C_2)}$	

Traditional Encoding Does Not Work Well

"121": **0111 1001**
"57": **0011 1001**

Ham(121, 57) = 1

"128": **1000 0000**
"127": **0111 1111**

Ham(127, 128) = 8

Traditional Encoding Does Not Work Well

"121": **0111 1001**
"57": **0011 1001**

Ham(121, 57) = 1

"128": **1000 0000**
"127": **0111 1111**

Ham(127, 128) = 8

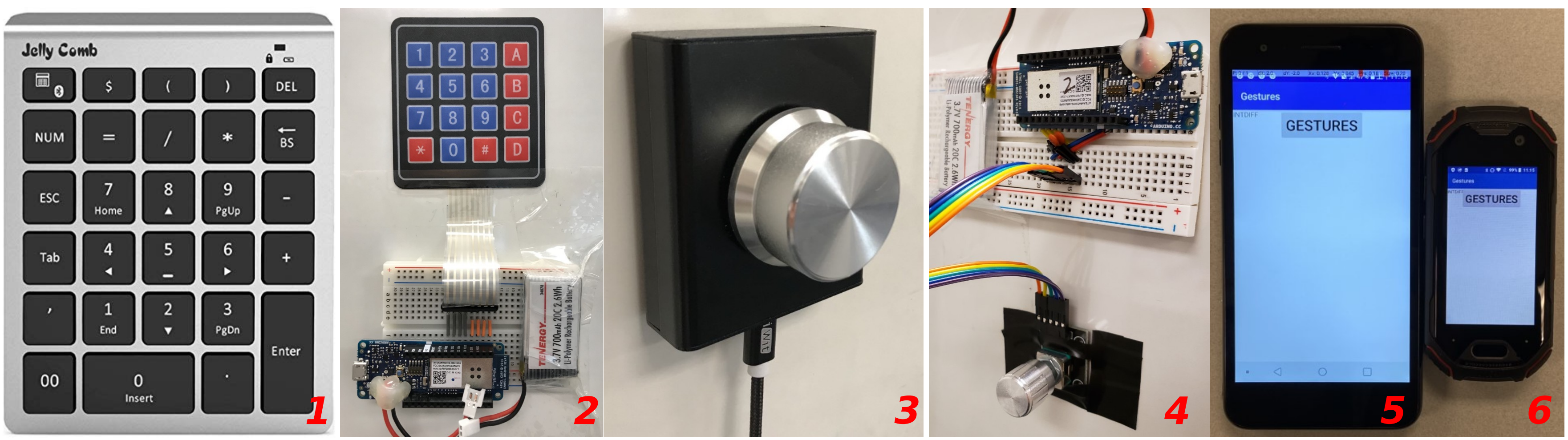
- **Our solution**: reduce an interval value by dividing a base value and represent it by counting "1".

$$n = \lfloor i/B \rfloor$$

$$e(i) = \underbrace{\overbrace{1, 1, \cdots, 1}^{n}, 0, 0, \cdots, 0}_{L}$$

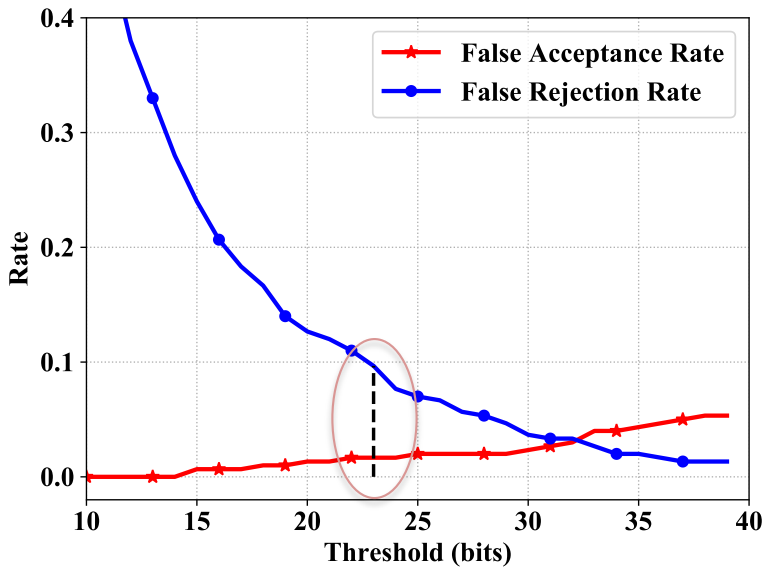
Evaluation

- Accuracy
- Resilience to mimicry attacks
- Randomness and entropy
- Parameter studies
  - Operation number, IMU sampling rate, postures, …
- Usability

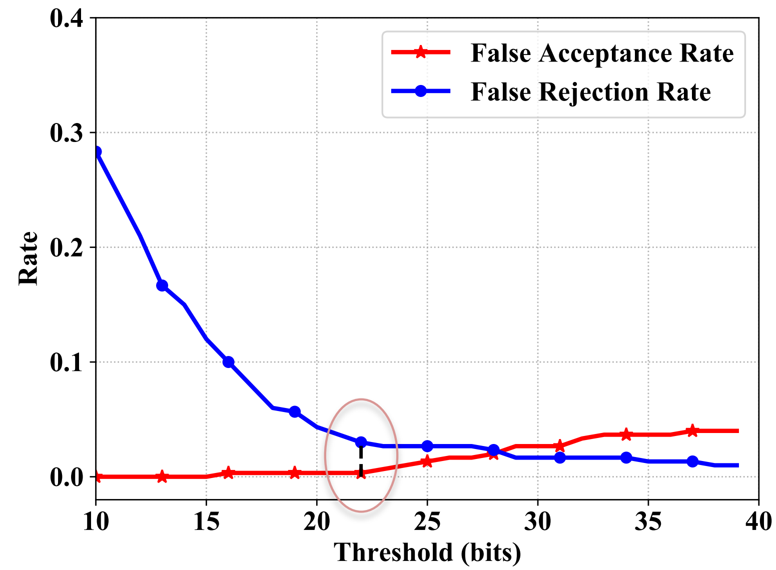


Accuracy

- Both FRR and FAR can be improved by adding random pauses.
- Pauses: 0.00 FAR and low FRR for button, knob and screen.



Button without pause (FRR: 0.10, FAR: 0.02)



Button with pause (FRR: 0.03, FAR: 0.00)

Resilience to Trained Mimicry Attacks

- The attacker practices well (i.e., training), stands close to the target user, and has a clear view

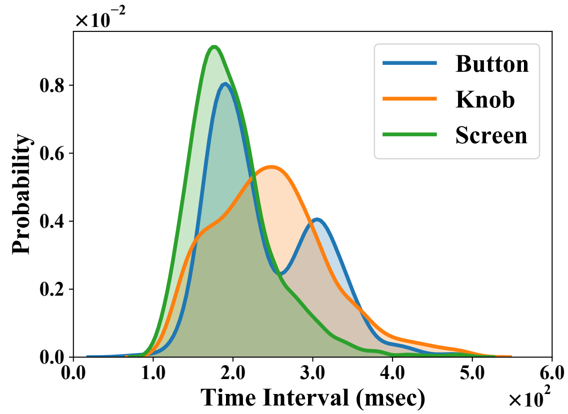
Pauses?	Dev.	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	Avg.
No	button	0.20	0.27	0.27	0.40	0.20	0.20	0.33	0.27	0.33	0.27	**0.274**
	knob	0.27	0.20	0.27	0.33	0.20	0.13	0.27	0.20	0.40	0.13	**0.240**
	screen	0.20	0.07	0.13	0.27	0.33	0.20	0.13	0.20	0.20	0.07	**0.180**
Yes	button	0.0	0.07	0.0	0.07	0.07	0.07	0.07	0.0	0.07	0.0	**0.040**
	knob	0.0	0.0	0.07	0.07	0.0	0.07	0.07	0.0	0.13	0.0	**0.040**
	screen	0.0	0.0	0.0	0.0	0.07	0.07	0.0	0.0	0.13	0.0	**0.027**

Randomness and Entropy

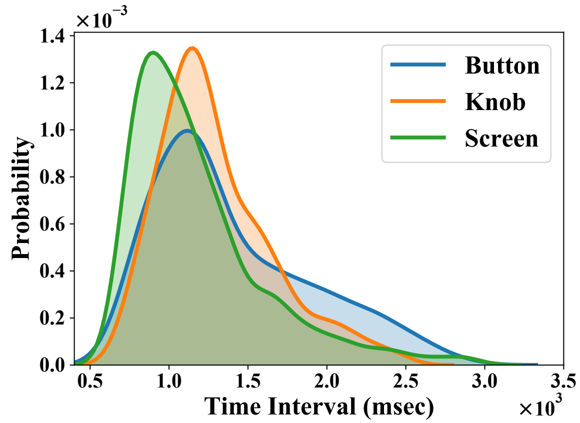
- Randomness
  - NIST statistical test ($p > 0.01$) confirms randomness.
  - Interval data is abstracted into **normal distributions**.
- Entropy

$$E_i = \frac{1}{2}\log_2(2\pi e\sigma^2) \qquad l_E = n_1 * E_1 + n_2 * E_2 + \log_2\binom{n_1+n_2}{n_2}$$

Device	Entropy (bits)	Bit Rate (bit/s)
button	34.3 – 38.5	10.3 – 13.2
knob	34.3 – 37.9	10.6 – 13.6
screen	32.3 – 36.6	11.6 – 14.8



Short Interval



Long Interval

Limitations

- If an attacker uses a camera that points at the user performing authentication, T2Pair is vulnerable online attacks
  - Offline attacks cannot succeed due to DH
- Still a low chance for trained mimicry attacks
  - More random pauses
- Not usable to hold a large phone and twist a small knob

Takeaways

- Prior IoT pairing approaches are insecure or inapplicable to constrained IoT devices
  - We propose the **first** secure and usable approach
- Simple operations (e.g., pressing a button, twisting a knob) are used for pairing
- Faithful fuzzy commitment: better accuracy
- Zero-knowledge password proof: turn a low-entropy "password" to a high-entropy key

Thank you !

Qiang Zeng

(zeng1@cse.sc.edu)