BASTION-SGX: Bluetooth and Architectural Support for Trusted I/O on SGX

Travis Peters[1], Reshma Lal[2], Srikanth Varadarajan[2], Pradeep Pappachan[2], David Kotz[1]
Dartmouth[1], Intel[2]

Hardware and Architectural Support for Security and Privacy (HASP)
@ ISCA 2018

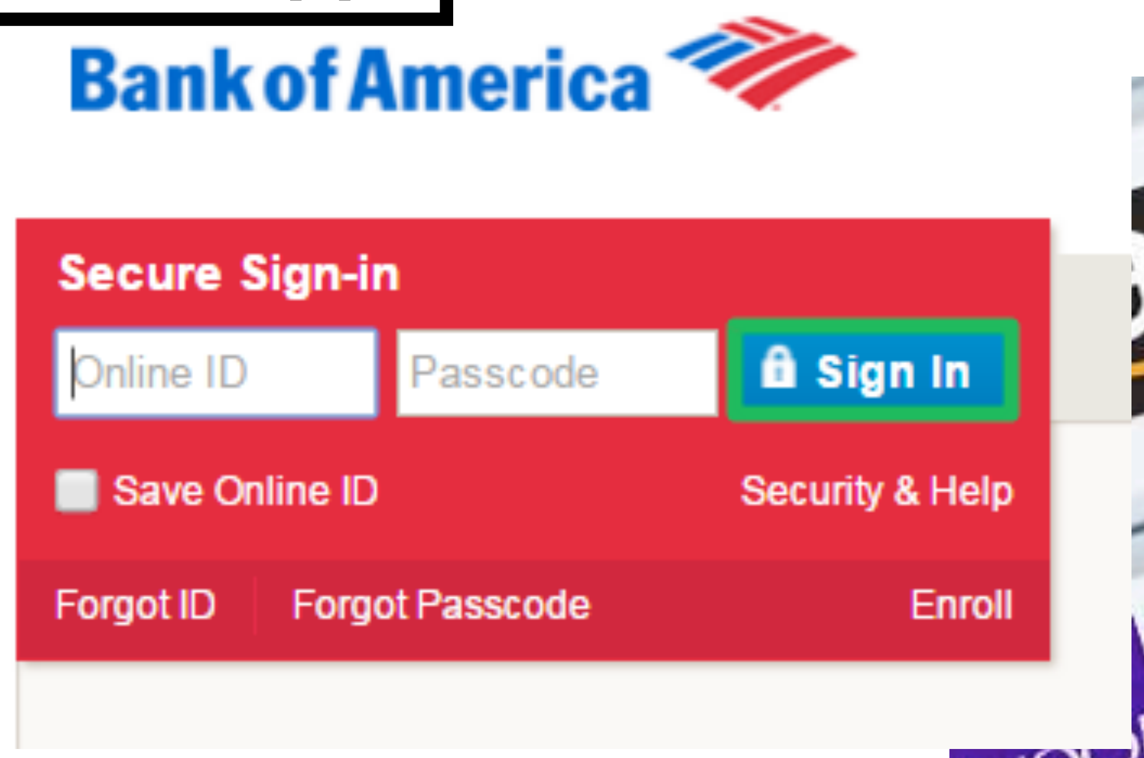
June 2nd, 2018
Los Angeles, CA, USA

Outline

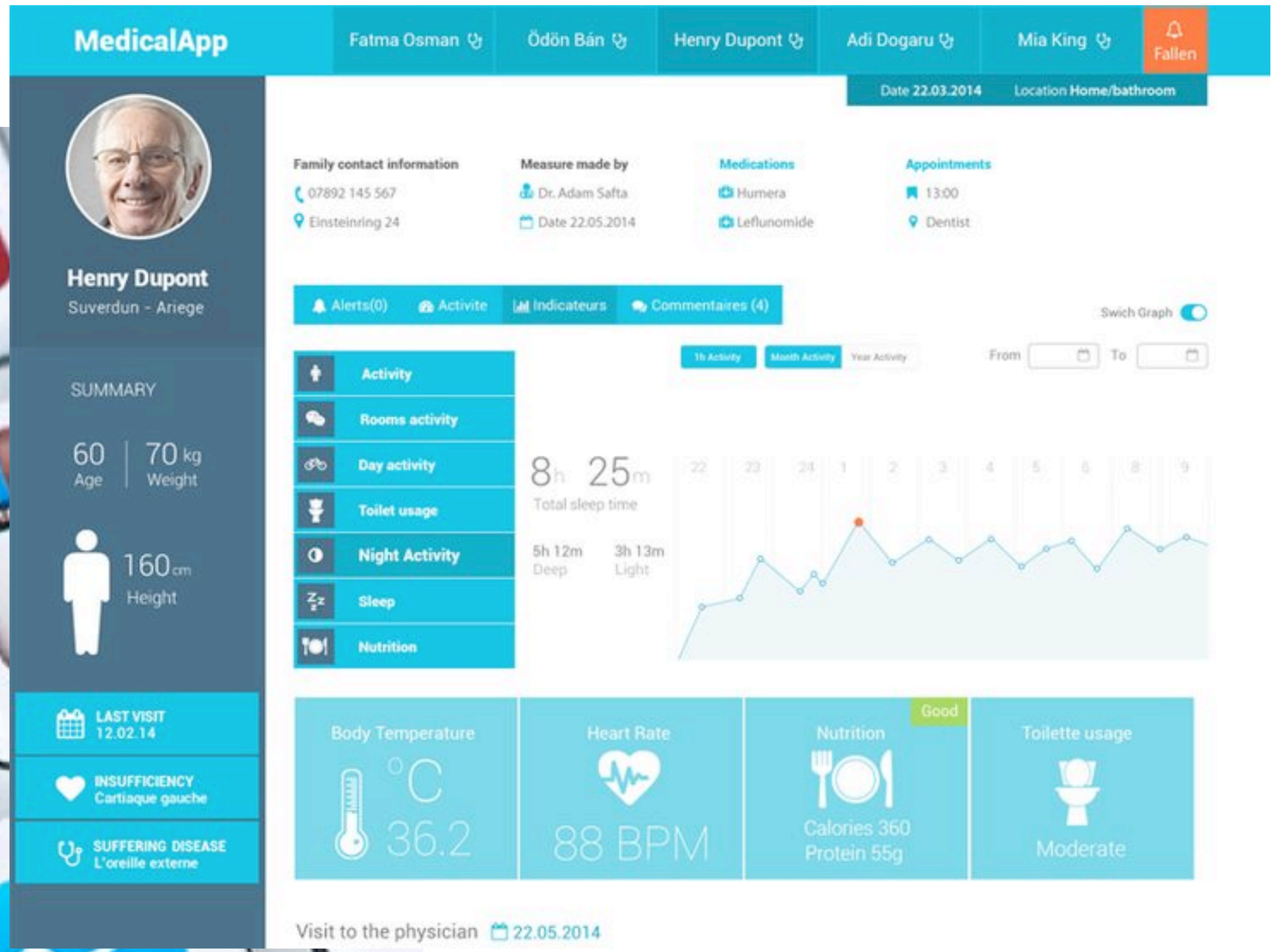
- Motivation
  App security & the insecurity of I/O — we need app security + I/O security!
- BASTION-SGX
  A novel Bluetooth Trusted I/O architecture
- Challenges
  Fine-grained channel selection & security policy enforcement
- Proof-of-Concept
  Delivering secure input from Bluetooth keyboards to SGX apps
- Conclusion
  Take-aways and future work

App Security is Imperative…

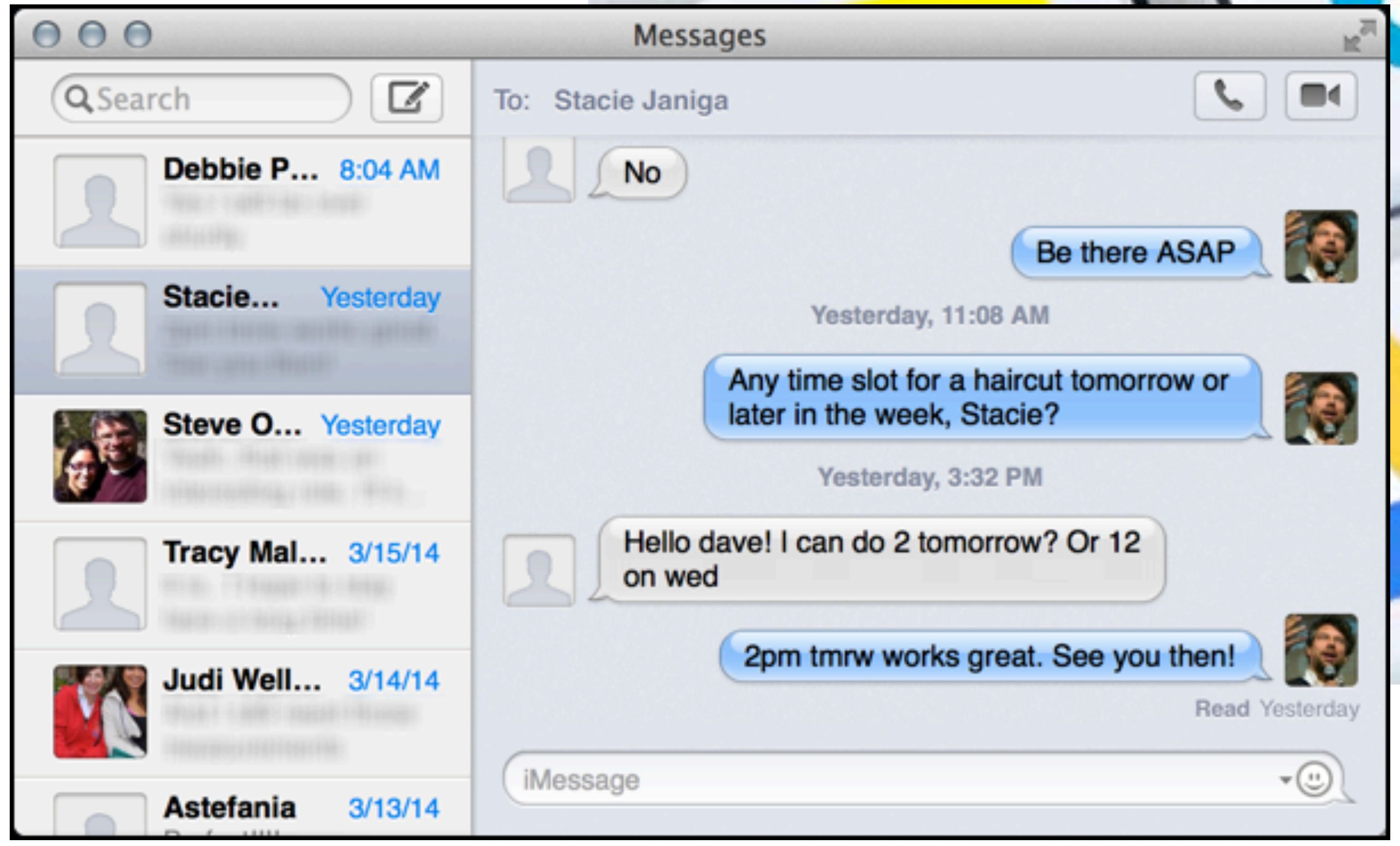
Financial Apps



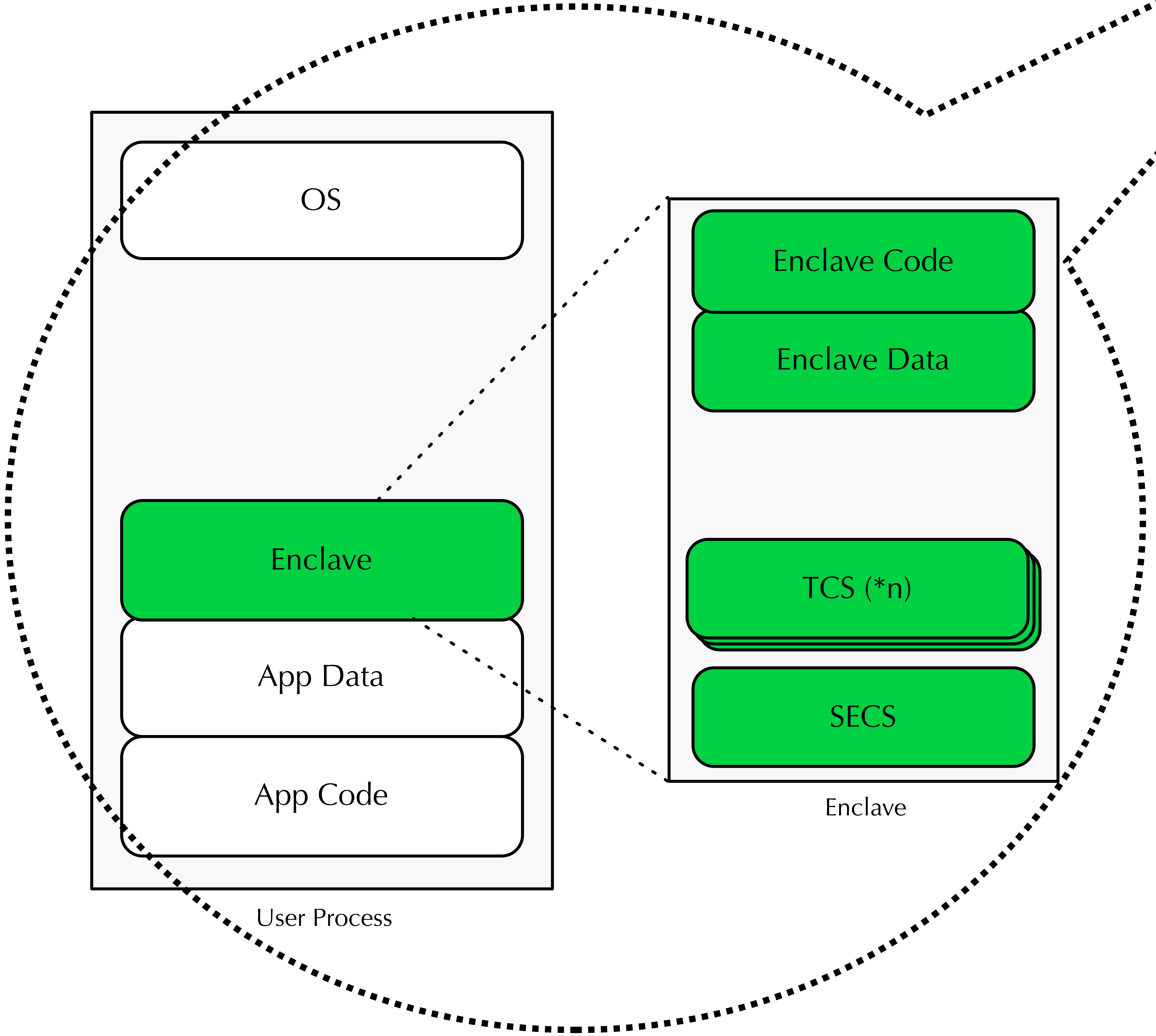
Health & Wellness Apps

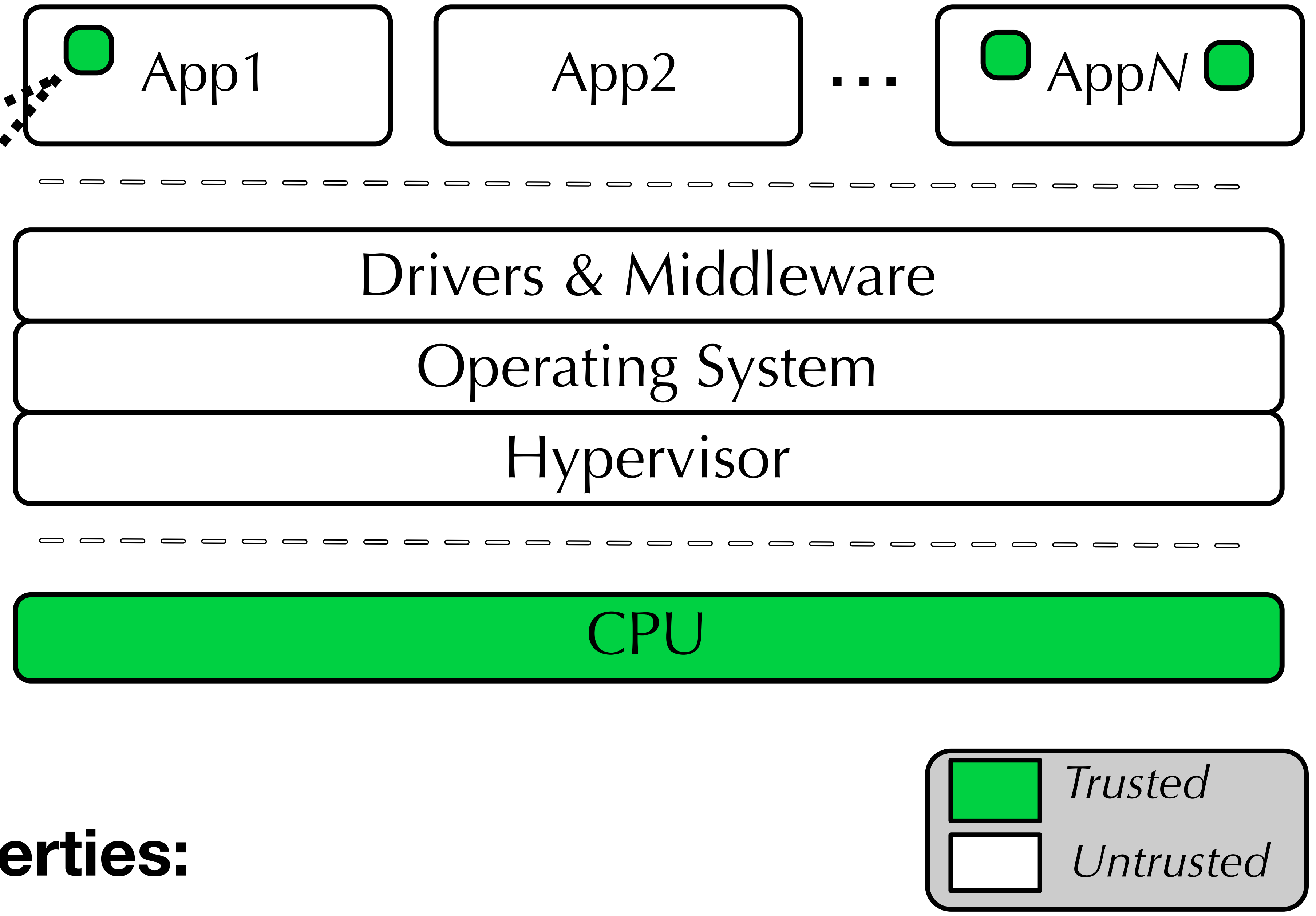


Messaging Apps



Intel's SGX Can Help!





Properties:

- Has its own code and data
- Provides confidentiality & integrity
- Full access to app memory

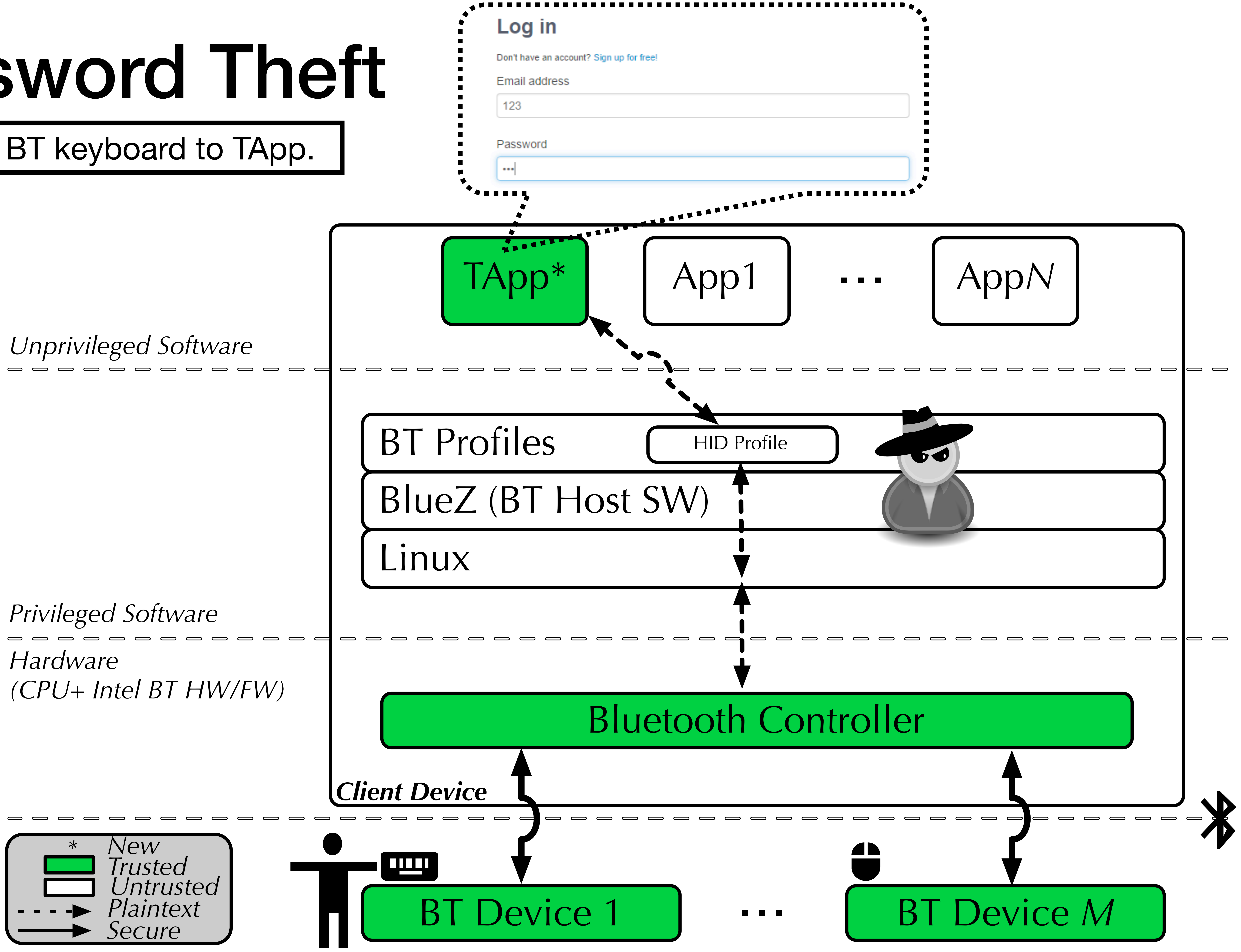
Highlights:

- Small attack surface (app + processor)
- Prevents even privileged SW from stealing or tampering w/ app secrets

I/O Security is ***Also*** Imperative!

Example: Password Theft

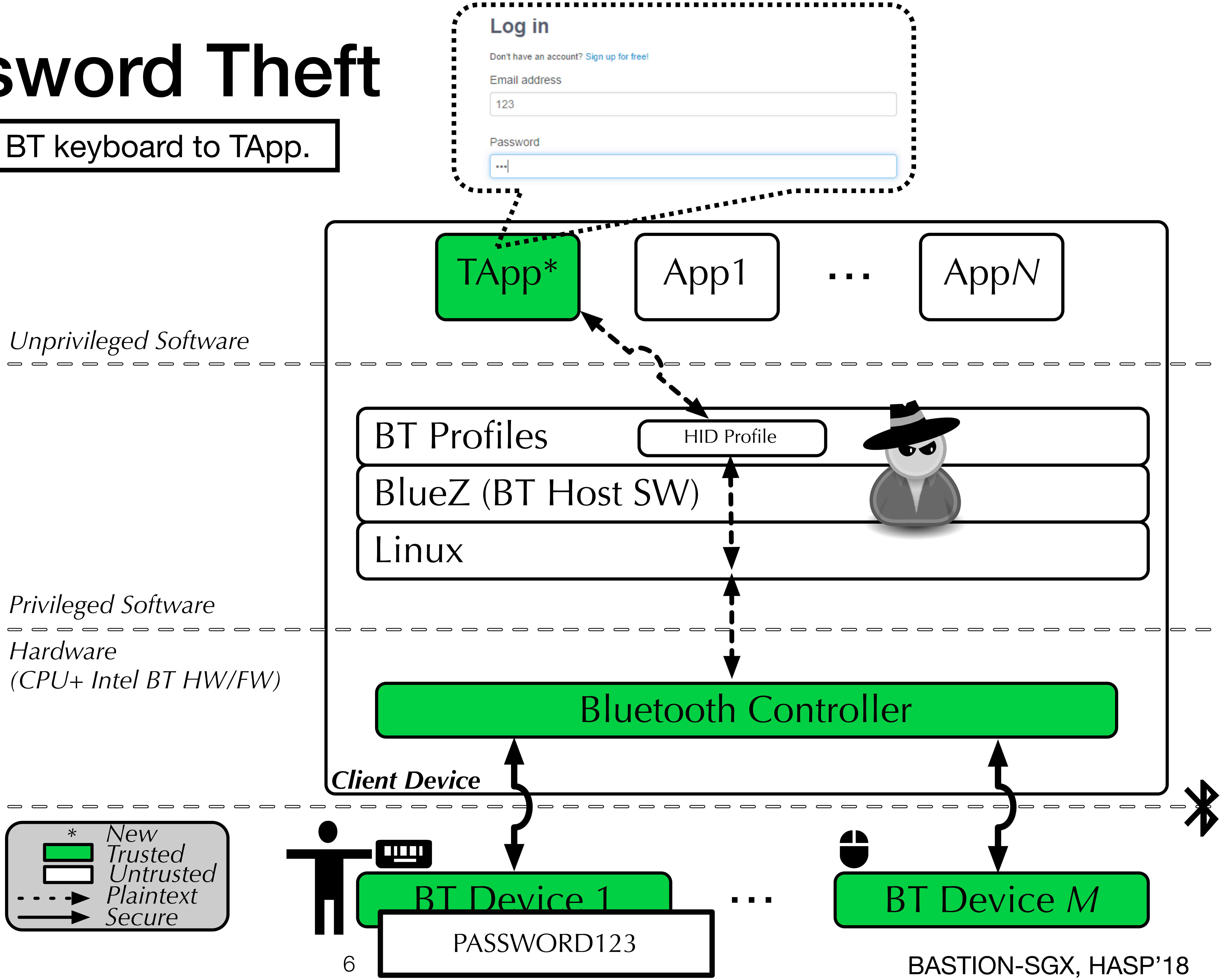
Objective: Secure input from BT keyboard to TApp.



Example: Password Theft

Objective: Secure input from BT keyboard to TApp.

1. User enters a password field and types her password…



Example: Password Theft

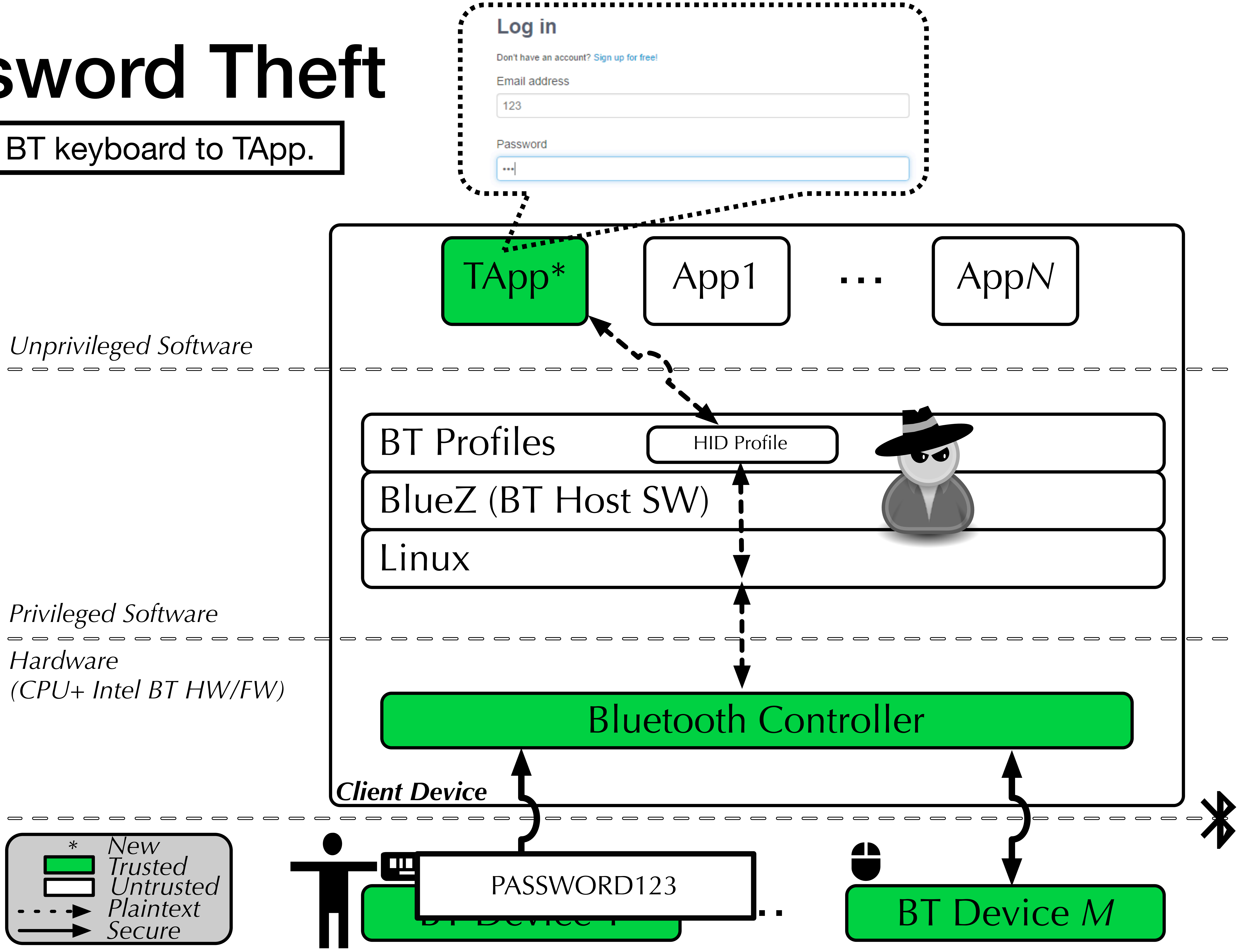
Objective: Secure input from BT keyboard to TApp.

1. User enters a password field and types her password…

2. The password is encapsulated within various BT protocol layers for transport and routing.

BT security protects the password during OTA transport.

The OTA packet is decrypted as soon as it arrives in the client's BT controller.



Example: Password Theft

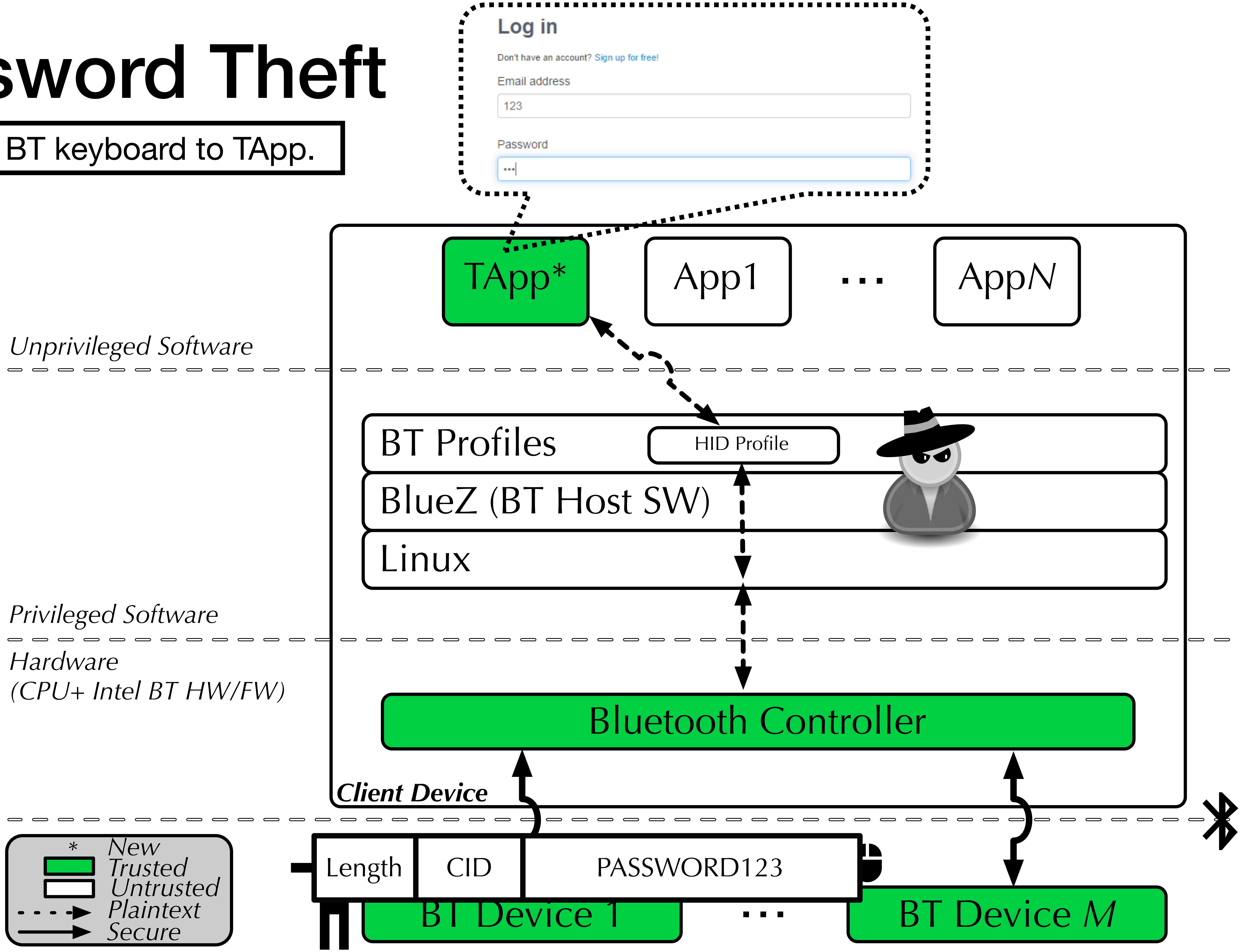
Objective: Secure input from BT keyboard to TApp.

1. User enters a password field and types her password…

2. The password is encapsulated within various BT protocol layers for transport and routing.

BT security protects the password during OTA transport.

The OTA packet is decrypted as soon as it arrives in the client's BT controller.



Example: Password Theft

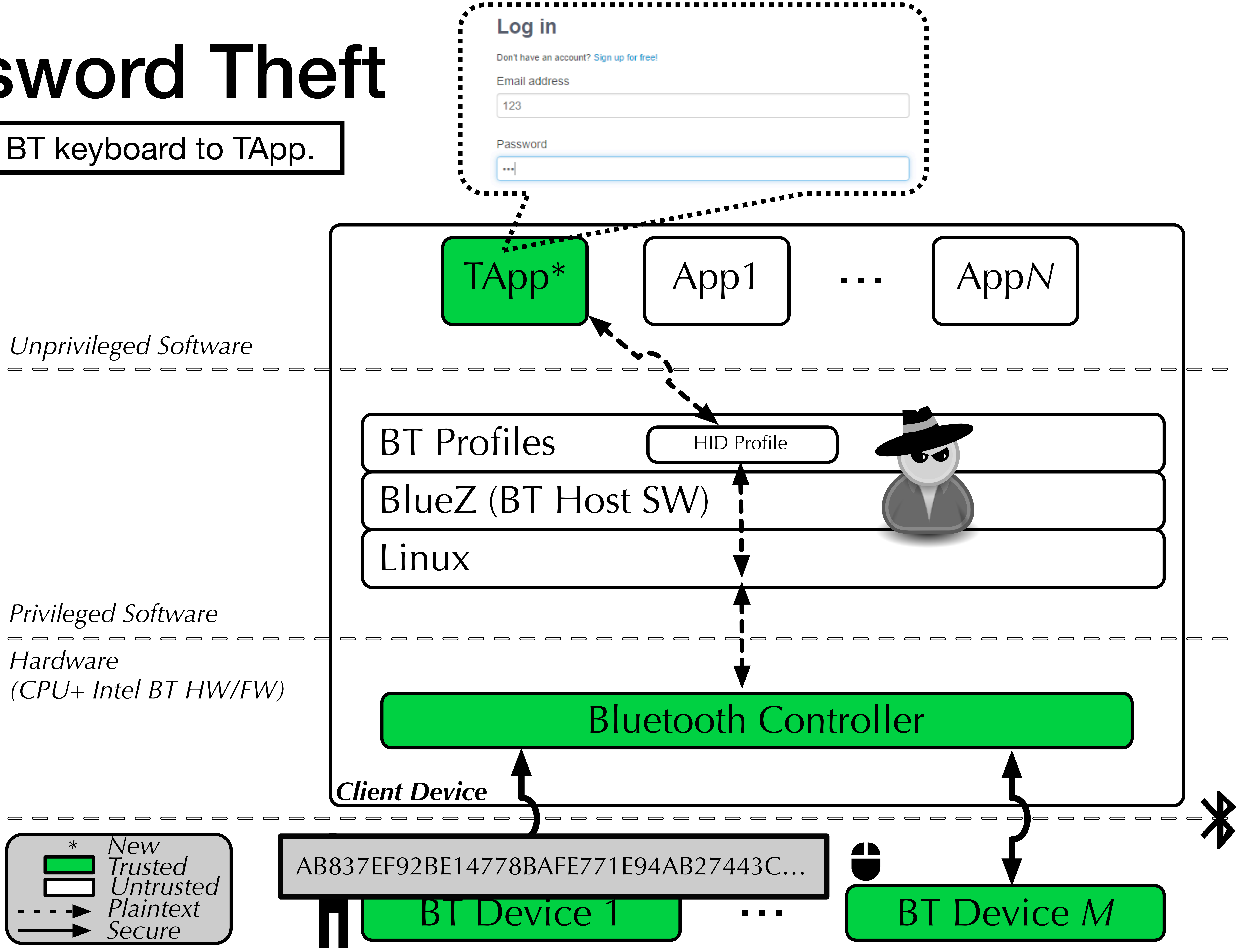
Objective: Secure input from BT keyboard to TApp.

1. User enters a password field and types her password…

2. The password is encapsulated within various BT protocol layers for transport and routing.

BT security protects the password during OTA transport.

The OTA packet is decrypted as soon as it arrives in the client's BT controller.



Example: Password Theft

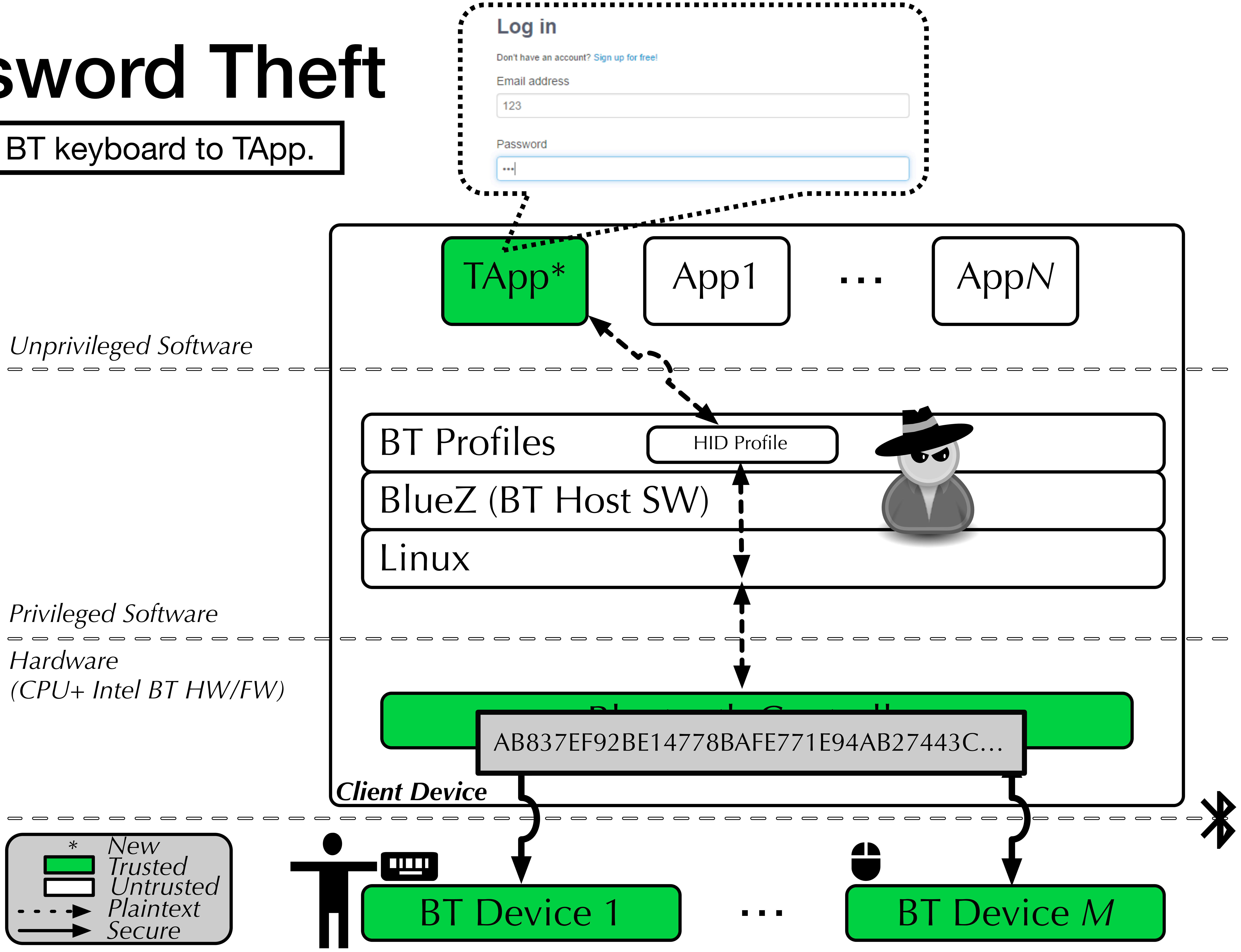
Objective: Secure input from BT keyboard to TApp.

1. User enters a password field and types her password…

2. The password is encapsulated within various BT protocol layers for transport and routing.

BT security protects the password during OTA transport.

The OTA packet is decrypted as soon as it arrives in the client's BT controller.



Example: Password Theft

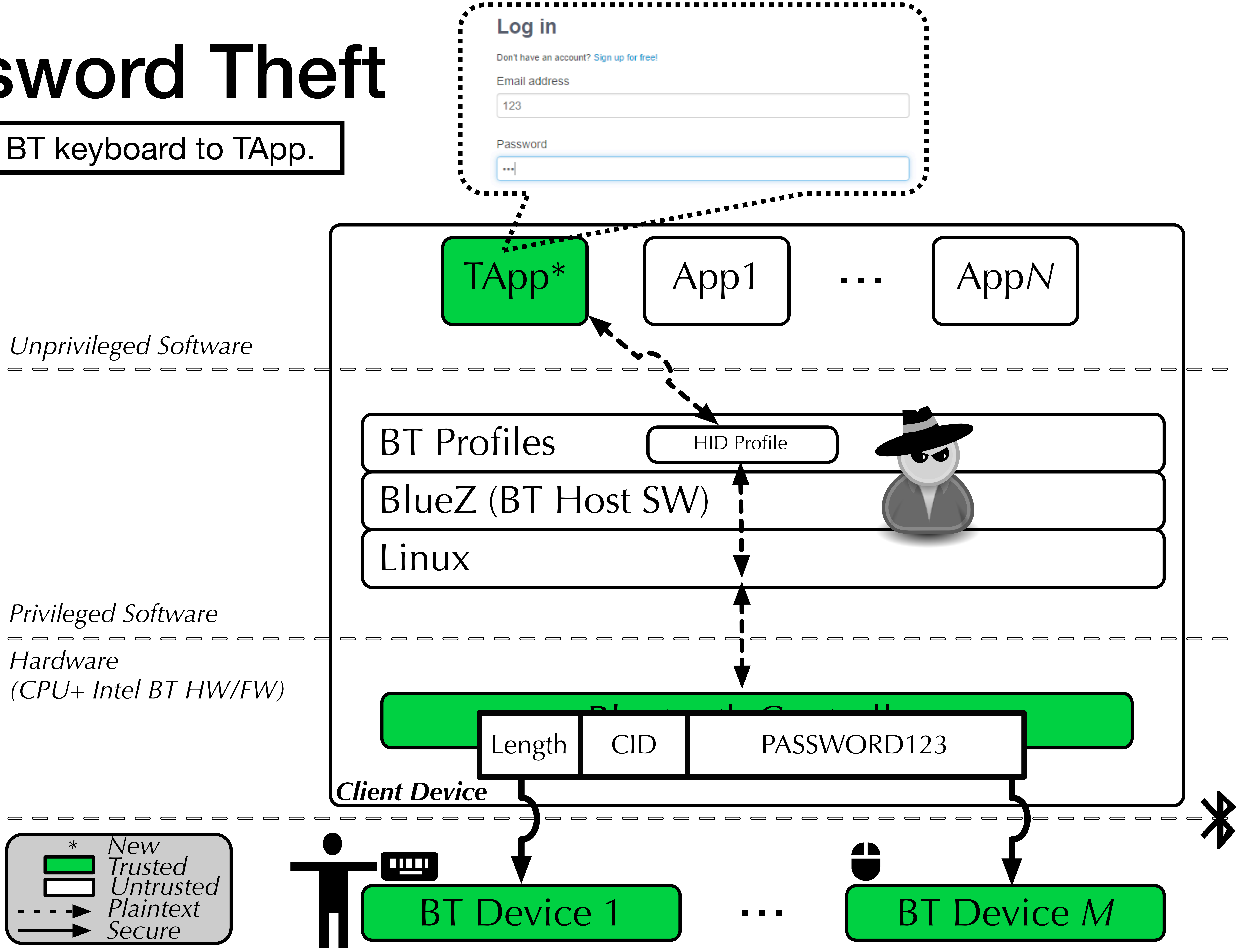
<u>Objective:</u> Secure input from BT keyboard to TApp.

1. User enters a password field and types her password…

2. The password is encapsulated within various BT protocol layers for transport and routing.

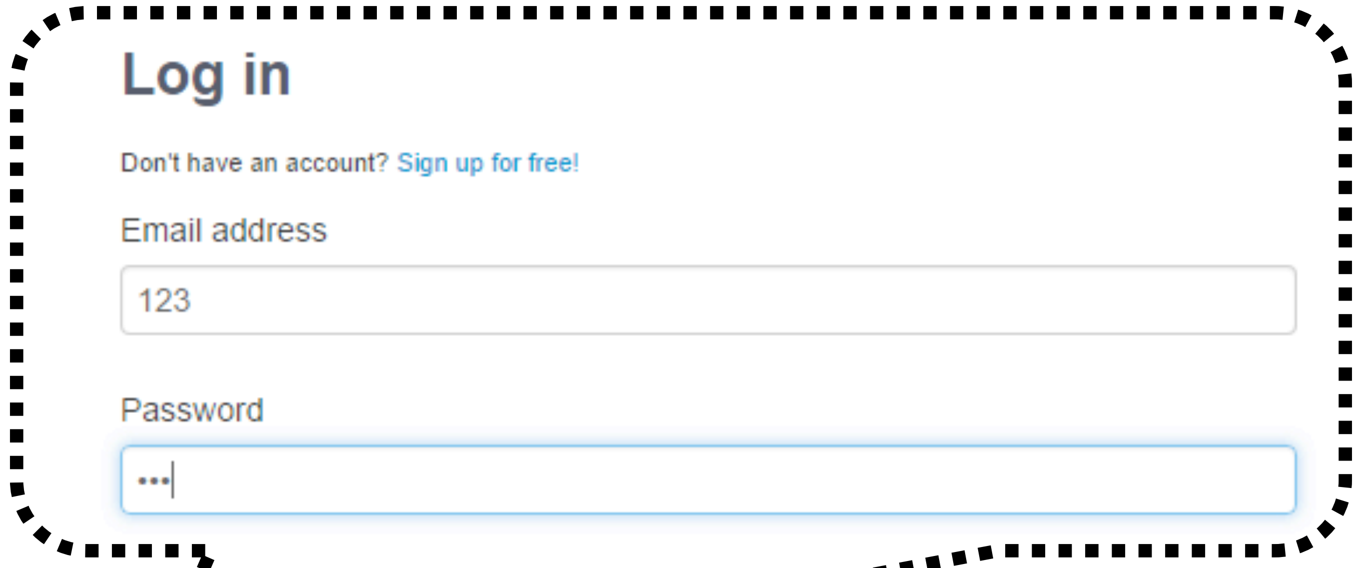
BT security protects the password during OTA transport.

The OTA packet is decrypted as soon as it arrives in the client's BT controller.



Example: Password Theft

Objective: Secure input from BT keyboard to TApp.

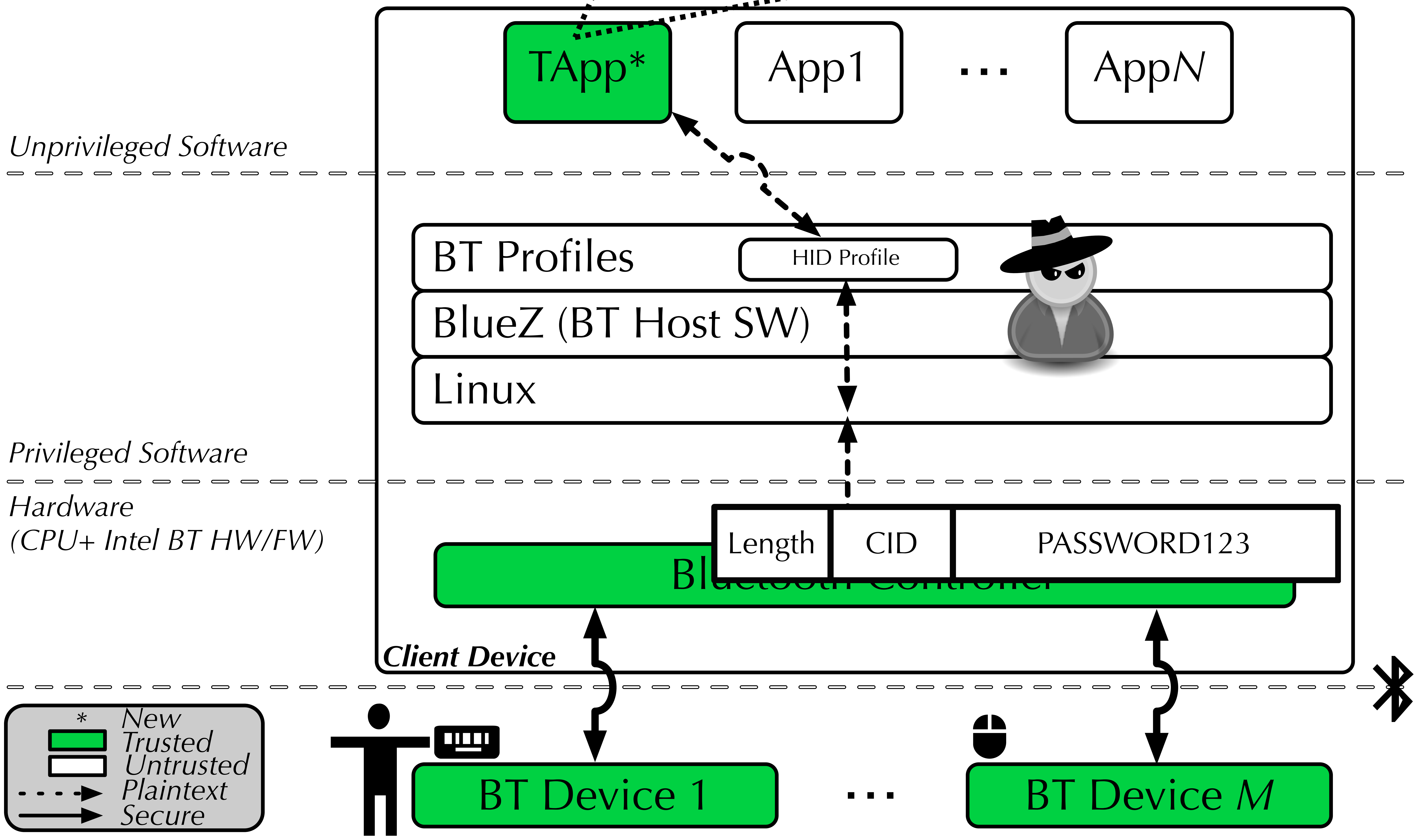


1. User enters a password field and types her password…

2. The password is encapsulated within various BT protocol layers for transport and routing.

BT security protects the password during OTA transport.

The OTA packet is decrypted as soon as it arrives in the client's BT controller.



Example: Password Theft

Objective: Secure input from BT keyboard to TApp.

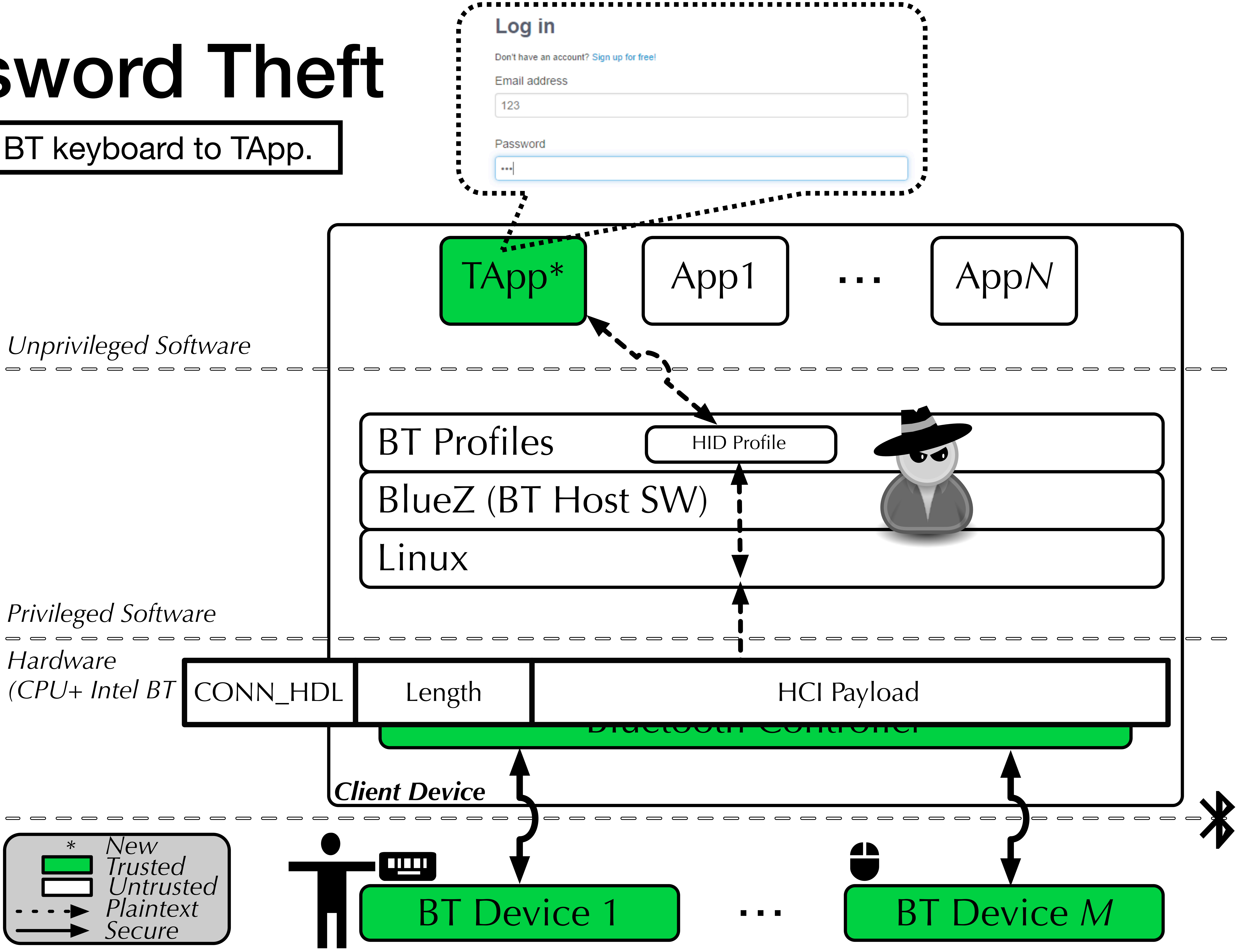
1. User enters a password field and types her password…

2. The password is encapsulated within various BT protocol layers for transport and routing.

BT security protects the password during OTA transport.

The OTA packet is decrypted as soon as it arrives in the client's BT controller.

3. HCI transport and L2CAP routing…



Example: Password Theft

Objective: Secure input from BT keyboard to TApp.

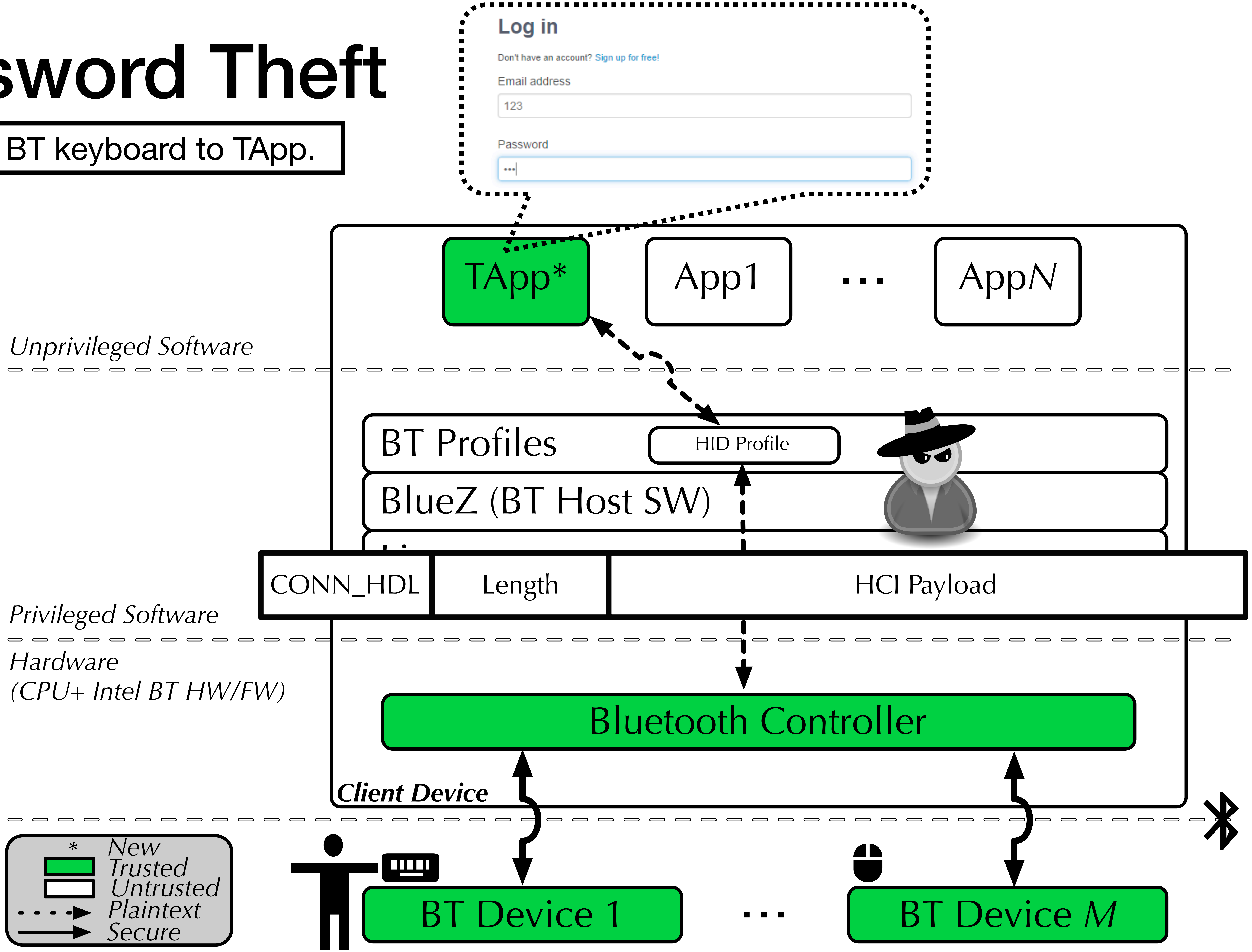
1. User enters a password field and types her password…

2. The password is encapsulated within various BT protocol layers for transport and routing.

BT security protects the password during OTA transport.

The OTA packet is decrypted as soon as it arrives in the client's BT controller.

3. HCI transport and L2CAP routing…



Example: Password Theft

Objective: Secure input from BT keyboard to TApp.

1. User enters a password field and types her password…

2. The password is encapsulated within various BT protocol layers for transport and routing.

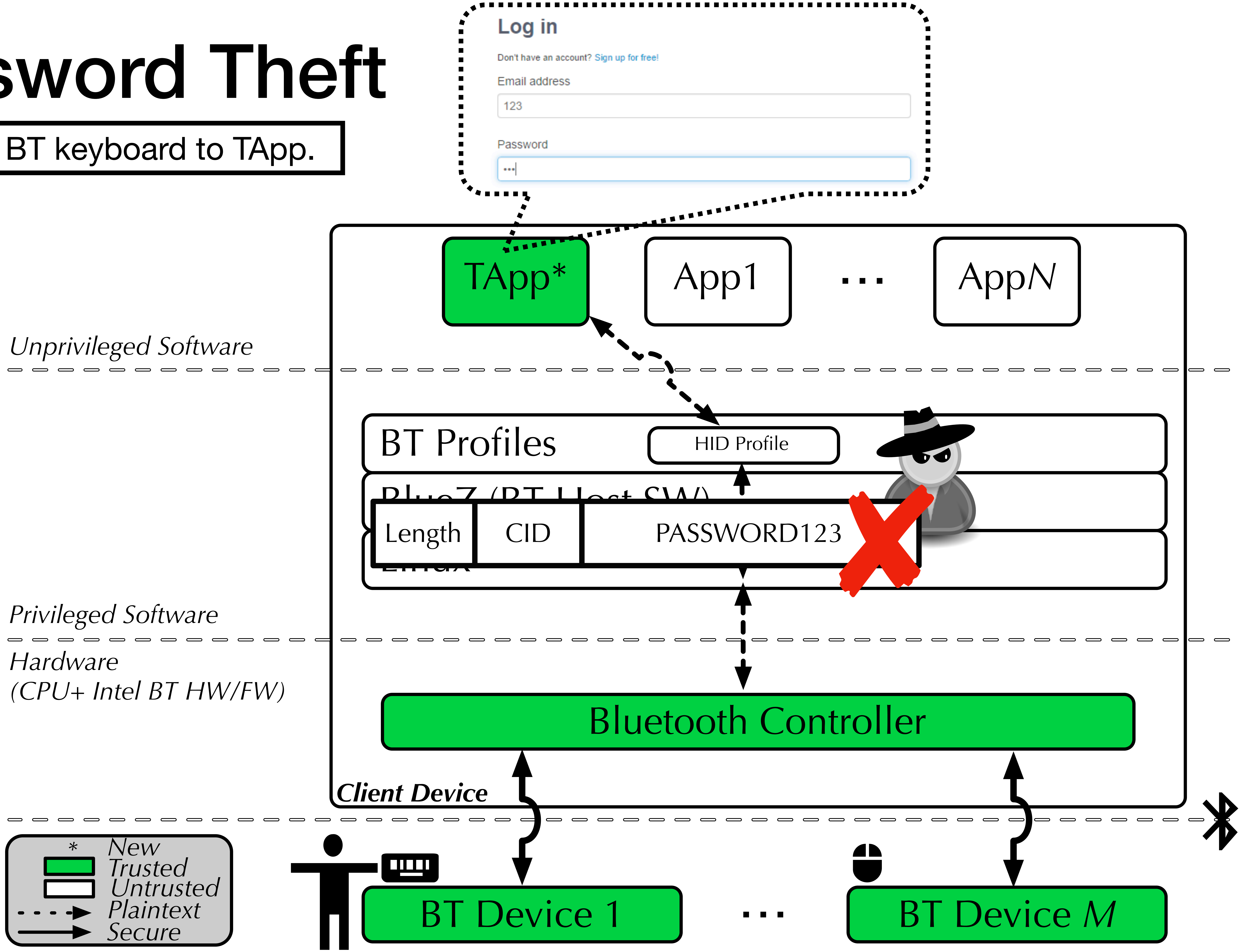
BT security protects the password during OTA transport.

The OTA packet is decrypted as soon as it arrives in the client's BT controller.

3. HCI transport and L2CAP routing…

Password is stolen!

Without Trusted I/O, data is vulnerable!



Trusted I/O

Key Insight: Break path into two subpaths (E1-E2, E3-E4).
Re-encrypt data between E1-E2 (enclave-controller).
Use existing OTA security between E3-E4 (client-device).

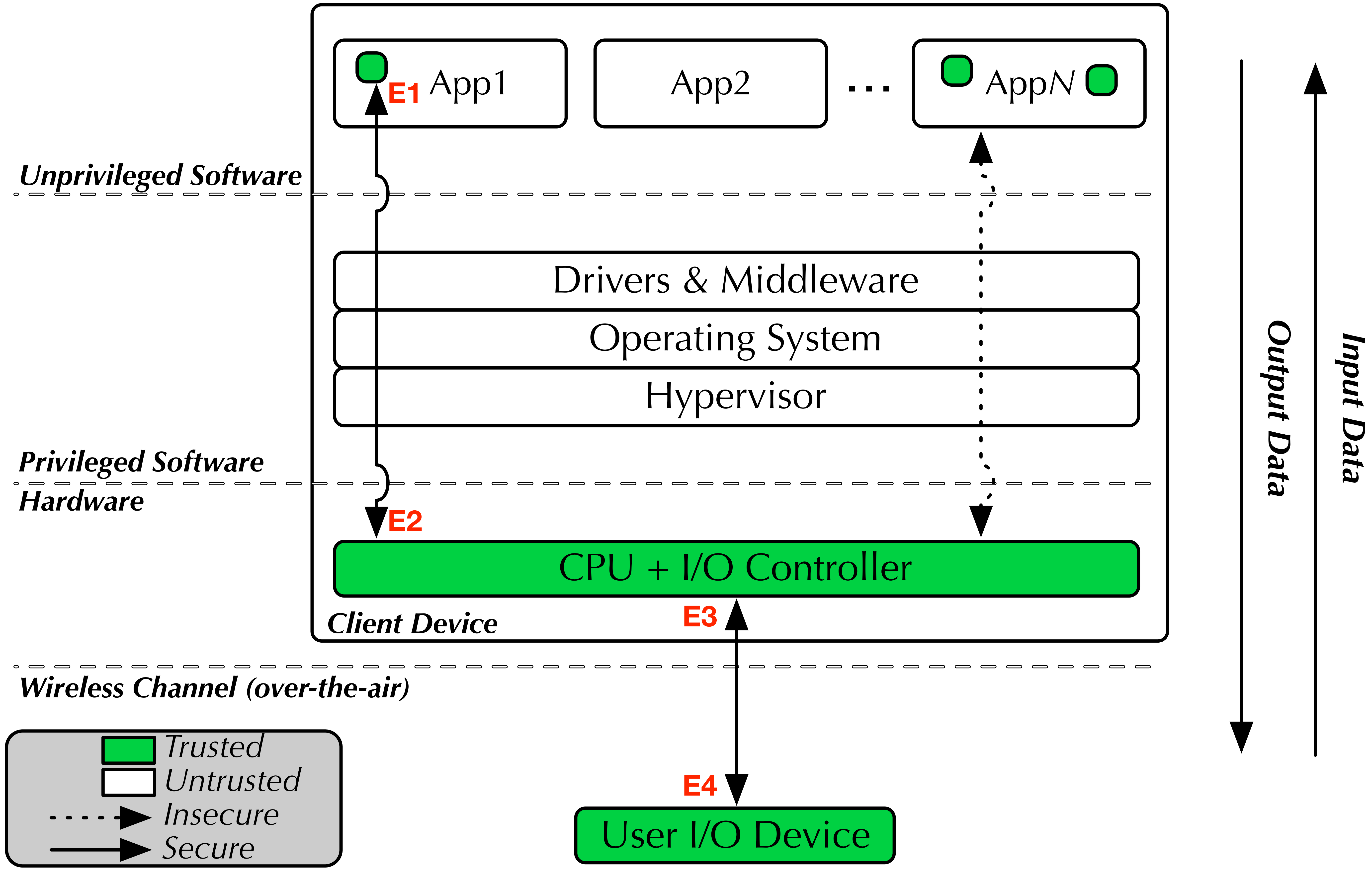
Our Goal:

- E2E security for select I/O data
- No new HW
- No changes to BT stack/devices
- No dependency on system SW

⟶ Minimal TCB!

This paper/talk:

- Focus on feasibility
- Secure *input* data from keyboard



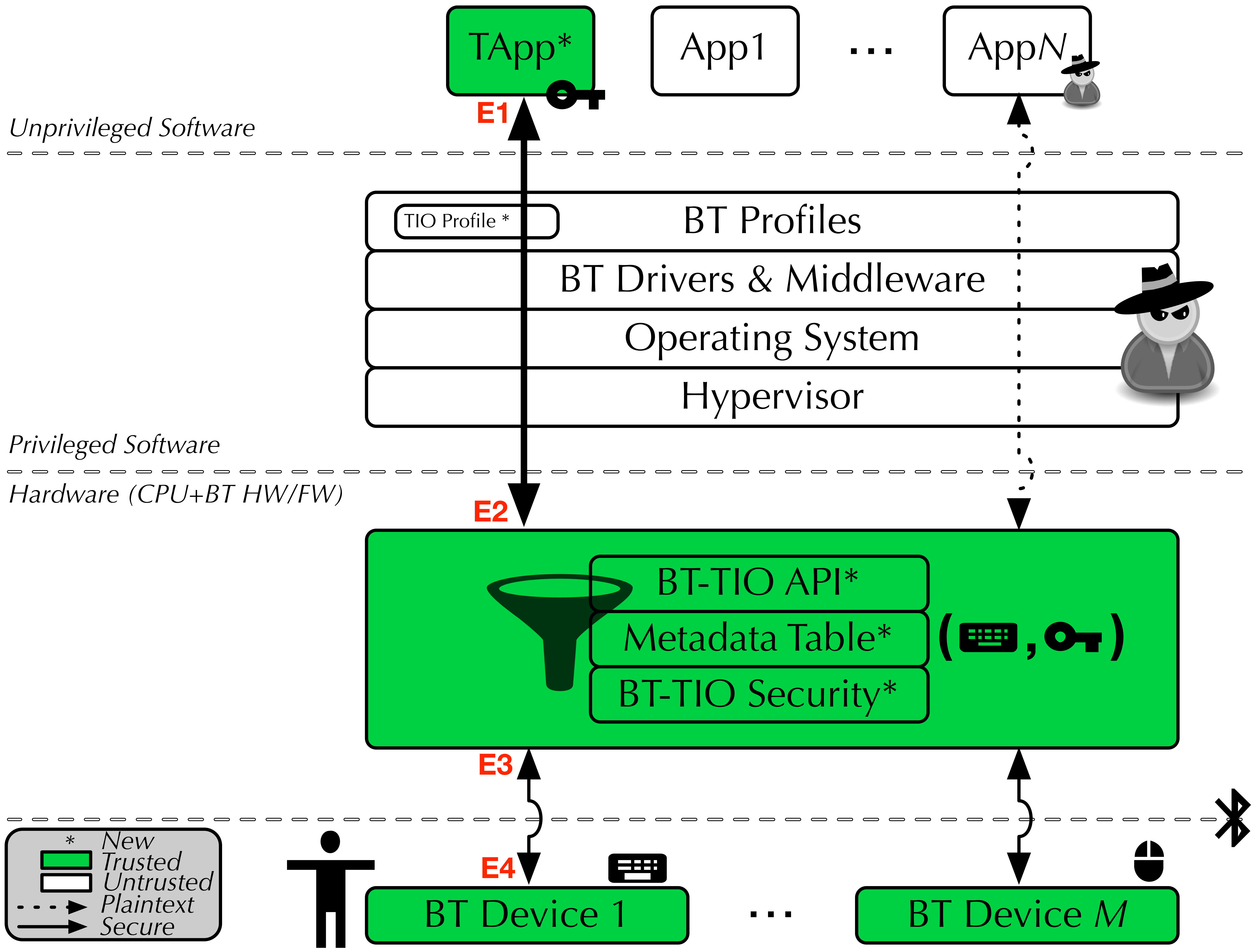
Proposed Architecture: BASTION-SGX

Bluetooth Trusted I/O Monitor & Filter
- Monitor *all* ingress/egress packets
- Update Metadata Table according to BT channel/connection-related events
- Send packets matching security policy to BT-TIO Security Module

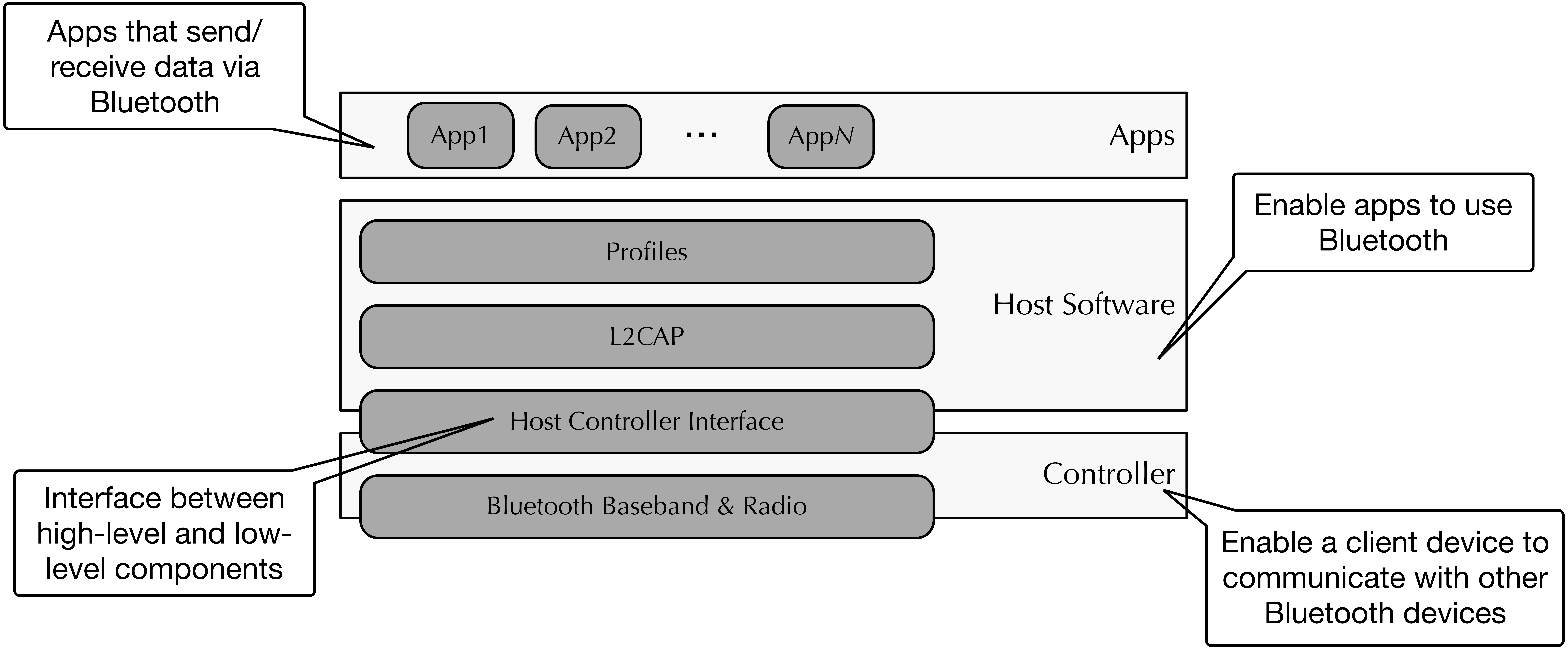
Bluetooth Trusted I/O Metadata Table
- Store connection/channel metadata

Bluetooth Trusted I/O API
- Enable apps to program *security policies* (i.e., tuple of (CHANNEL-ID & KEY))
- Use extensible interface for 3rd party features (Vendor Specific Debug Commands)

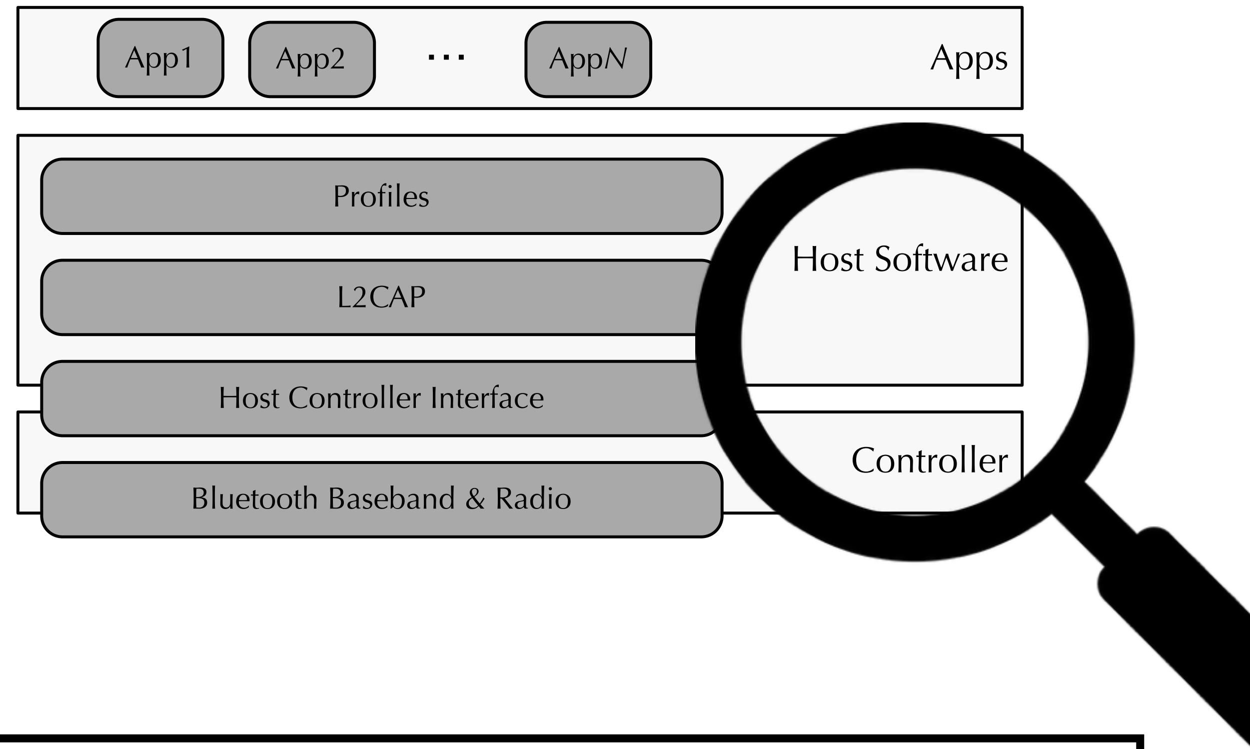
Bluetooth Trusted Security Module
- Cryptographic operations (e.g., encryption, decryption)

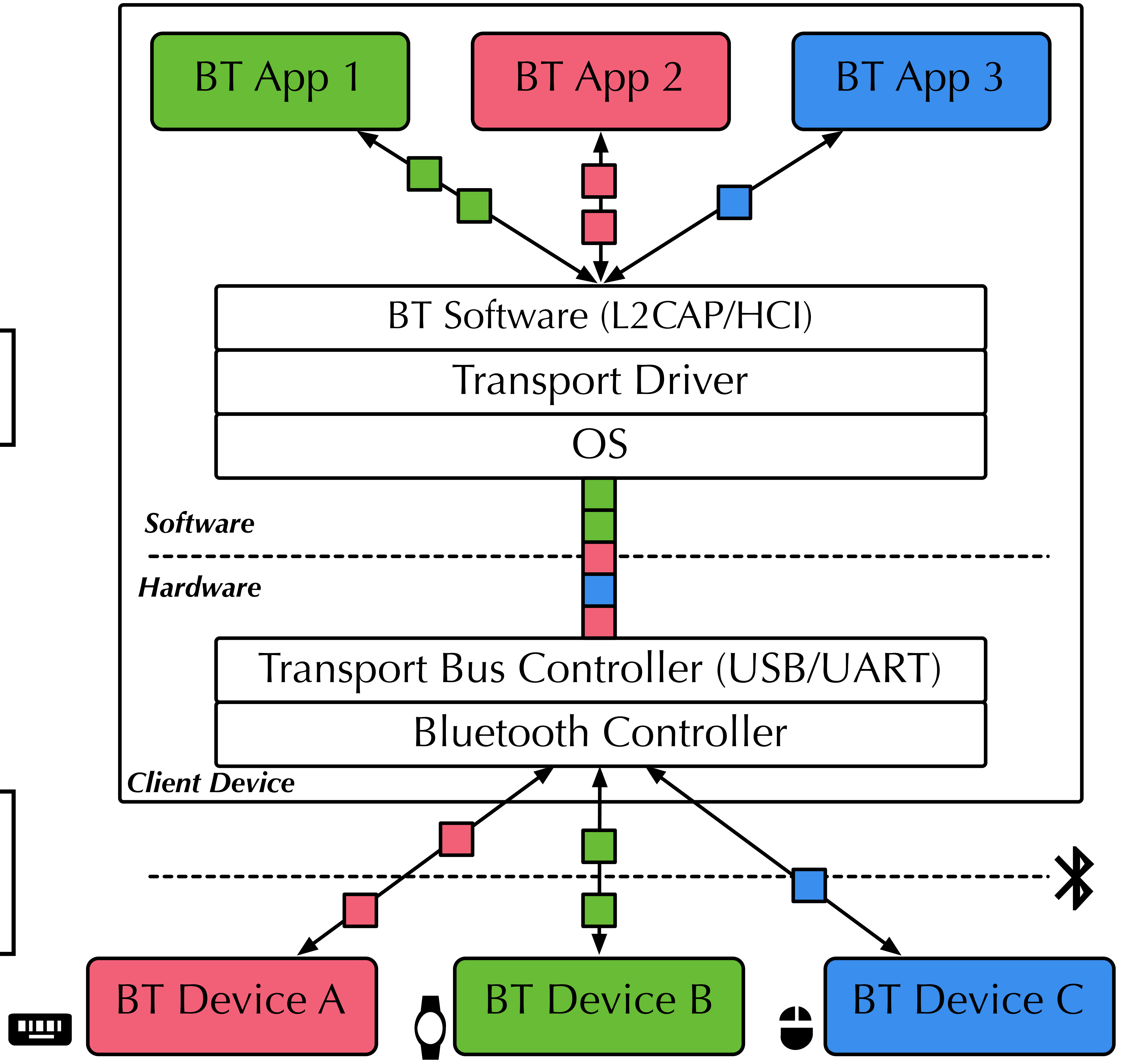


Bluetooth Architecture Overview



Requirements & Challenges





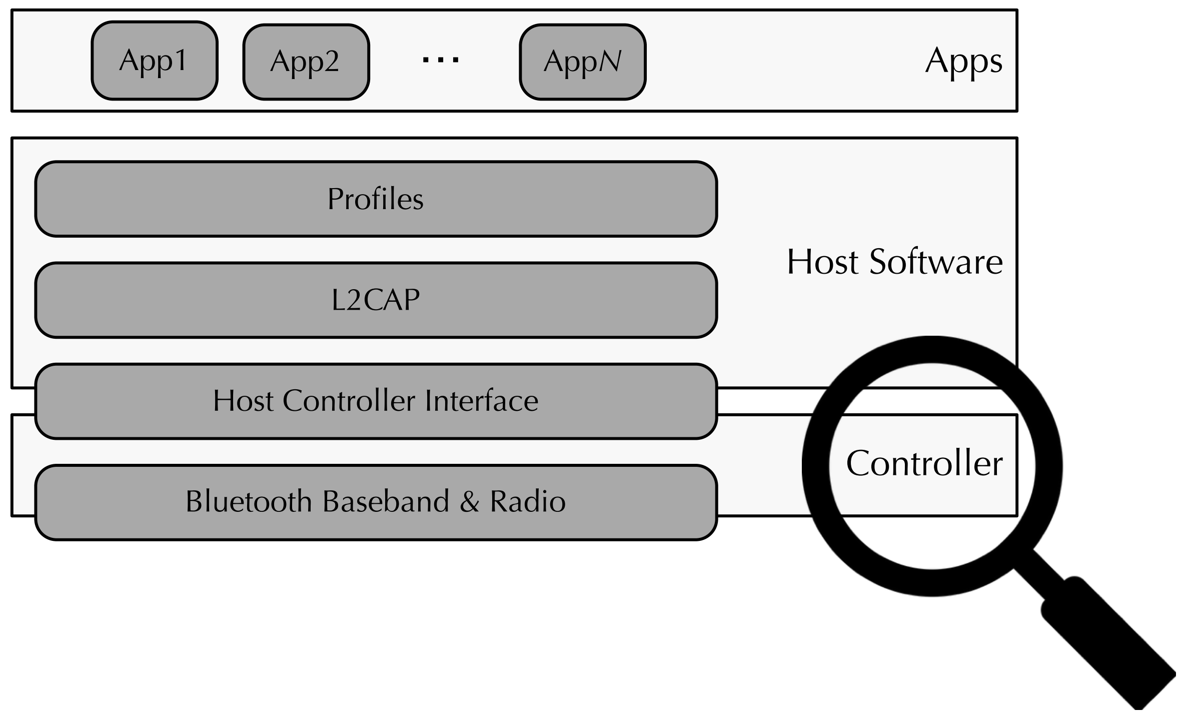
4. Security applied to one channel should not affect other BT channels.

3. Security should only be applied to ***data*** packets, not ***control*** packets.

2. Host SW is responsible for using **HCI** and **L2CAP** packet headers for HCI transport and routing.

1. All packets are multiplexed within the Client's Bluetooth Controller & sent to Host SW in a single stream.

Anatomy of BT Connection

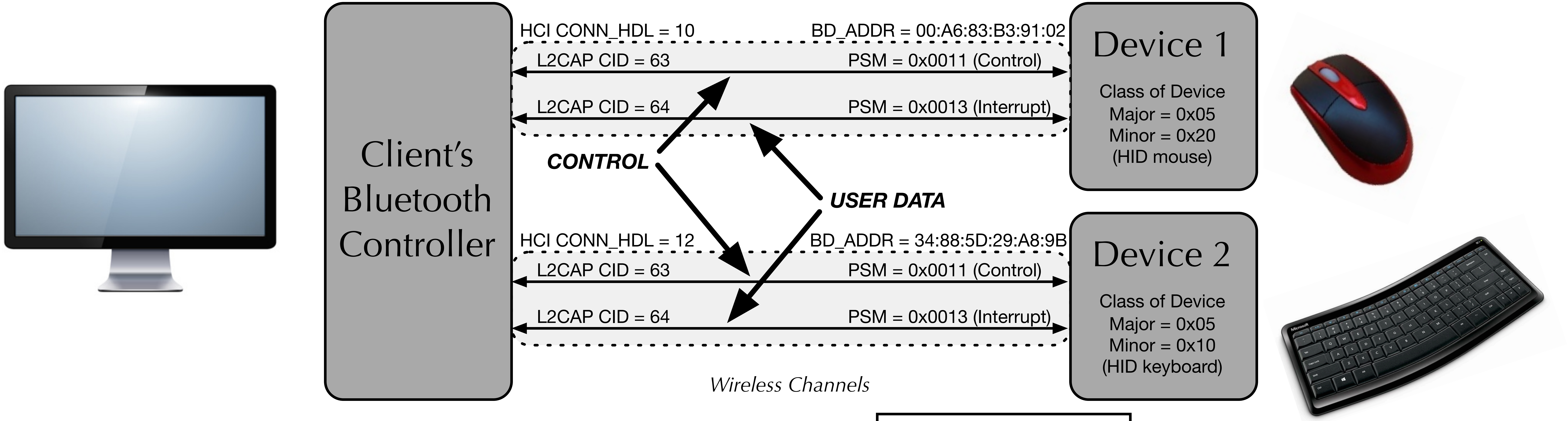


Q: *How can a Bluetooth Controller identify <u>specific channels</u> over which to enforce Trusted I/O security?*

CONN_HDL/BD_ADDR identifies specific device connection

L2CAP CIDs identify individual channels within a connection

CoD defines device type



PSM defines protocol/service type

Case Study: Securing HID Input

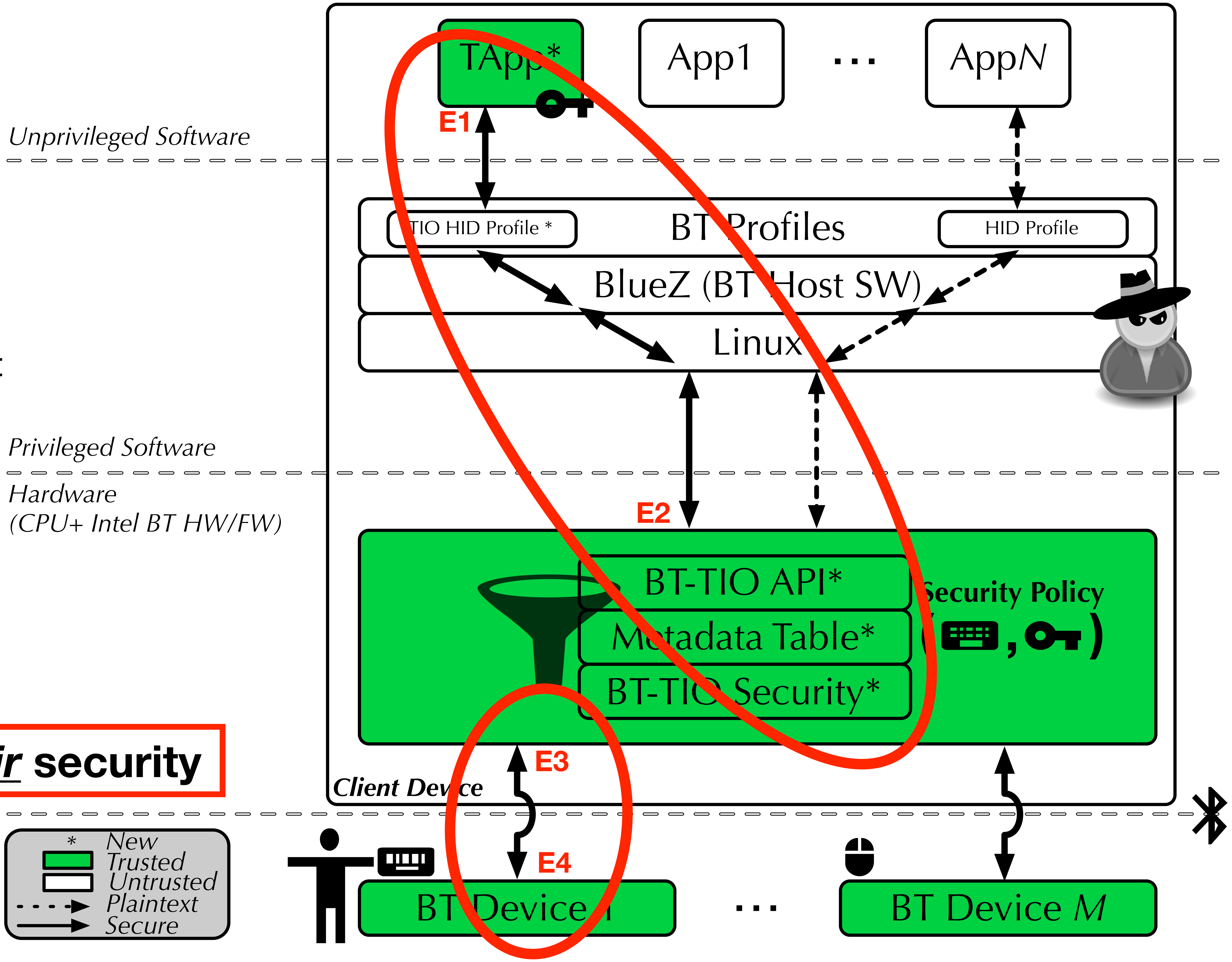
Setup:
- Implement BASTION-SGX architecture (Section 4)
- Implement trusted app (TApp) for password input
- Install privileged keylogger malware — logs *all* HID data

Goals:
- Validate Bluetooth Controller's capabilities (re: fine-grained channel selection)
- Validate that even privileged malware cannot decipher input while security policy is programmed into the Bluetooth Controller

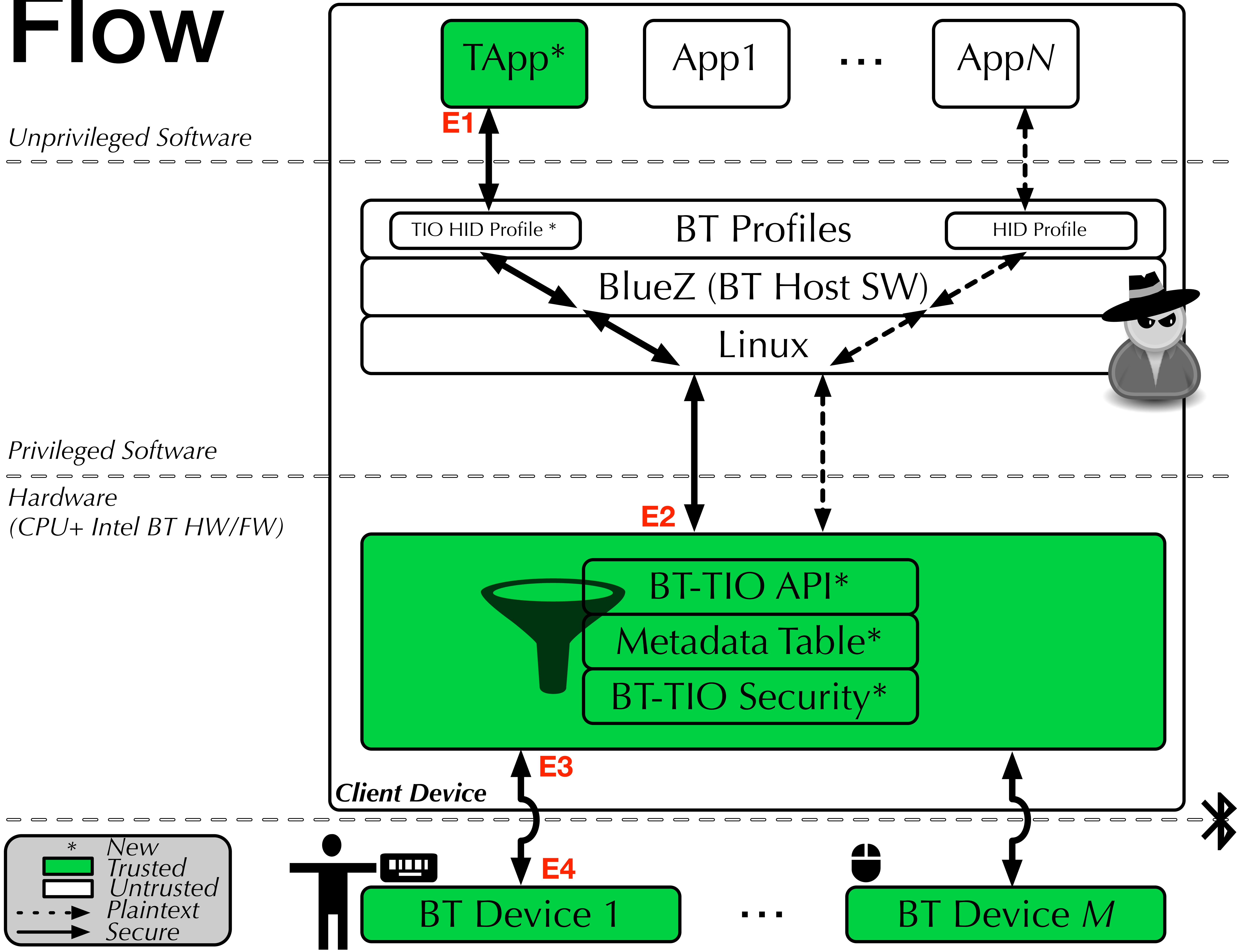
We show that end-to-end (device-to-app) security is possible where....

E1-E2 is secured w/ <u>new</u> <u>*in-host*</u> security

E3-E4 is secured w/ <u>*existing*</u> <u>*over-the-air*</u> security

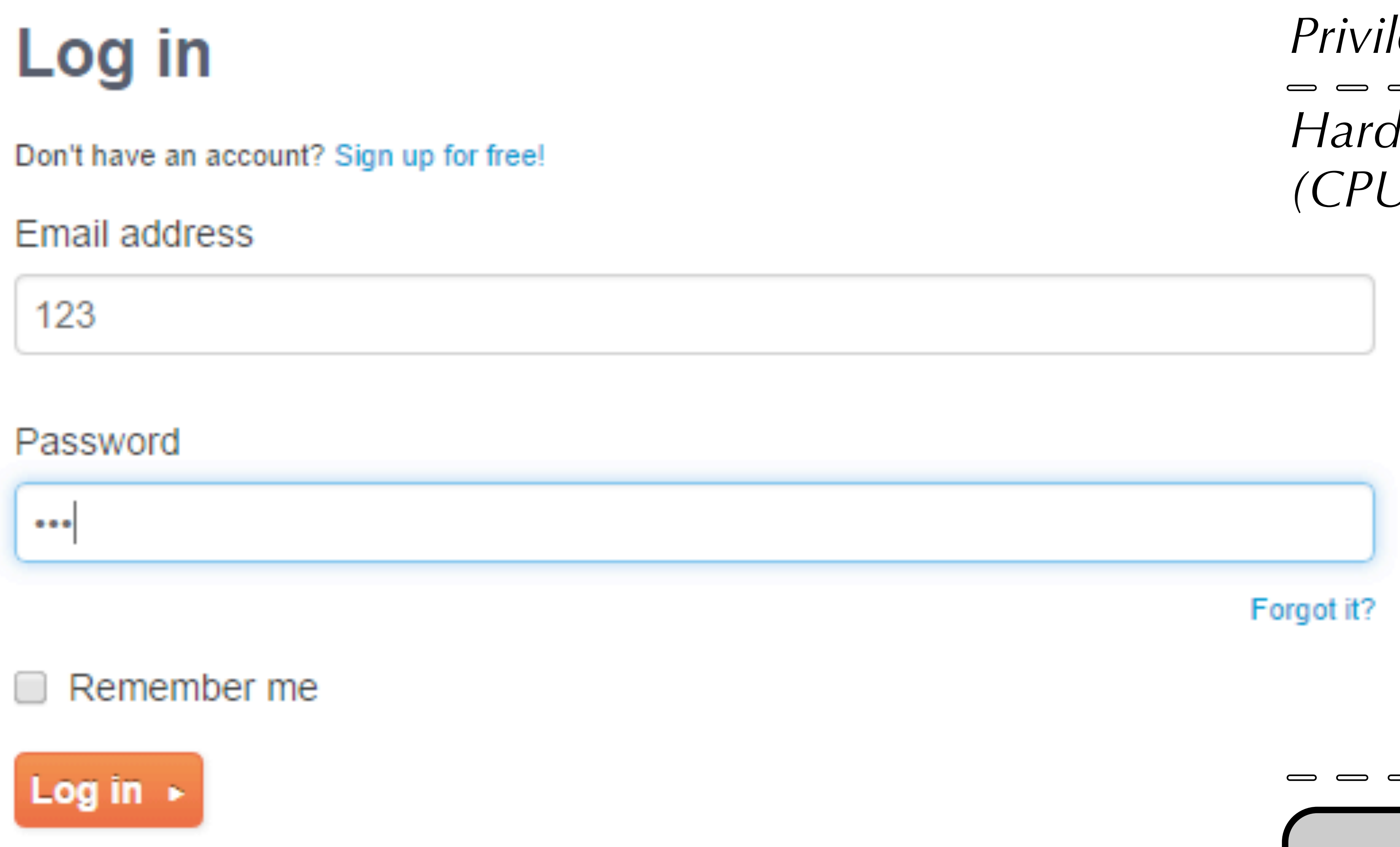


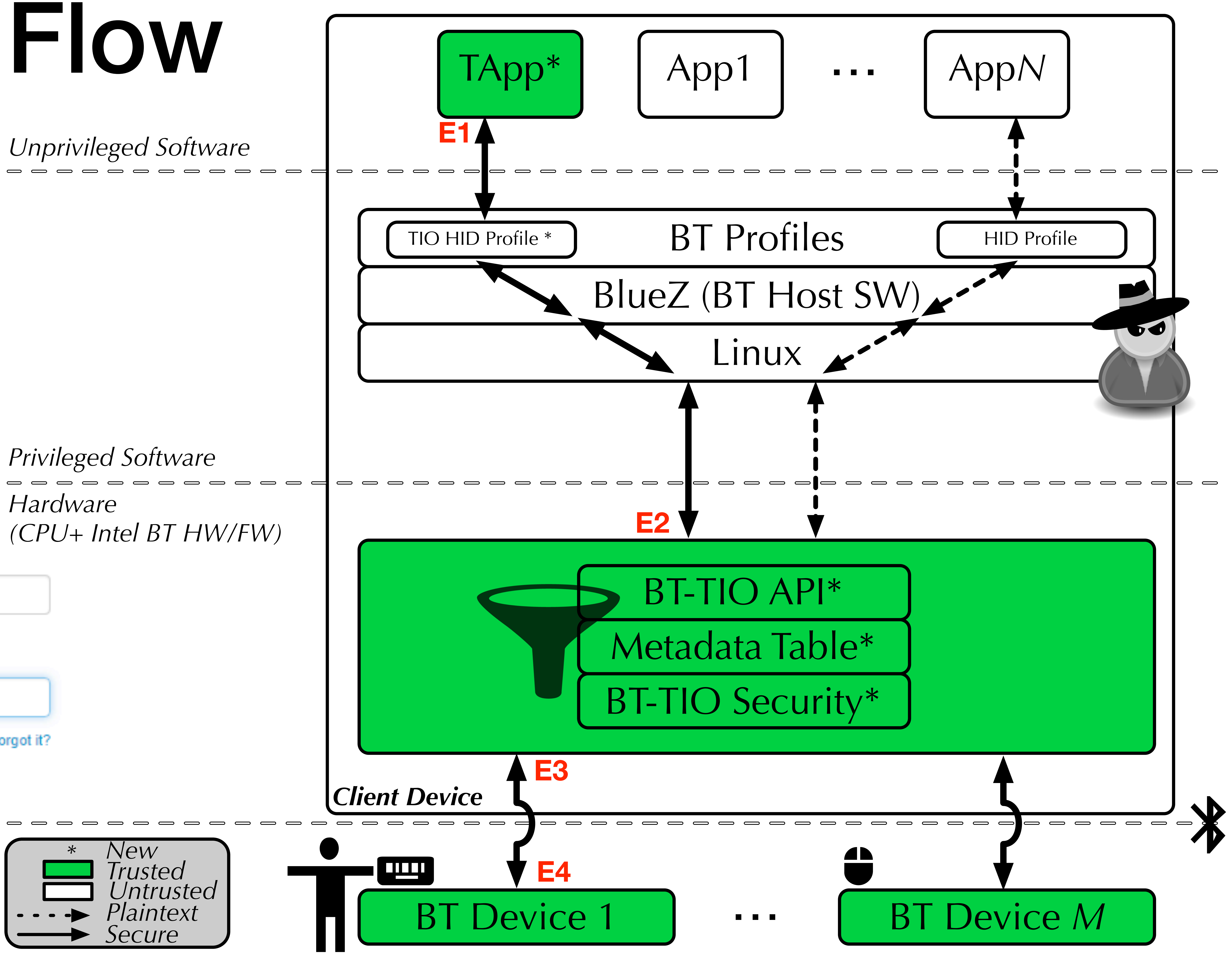
Secure Input Flow



Secure Input Flow

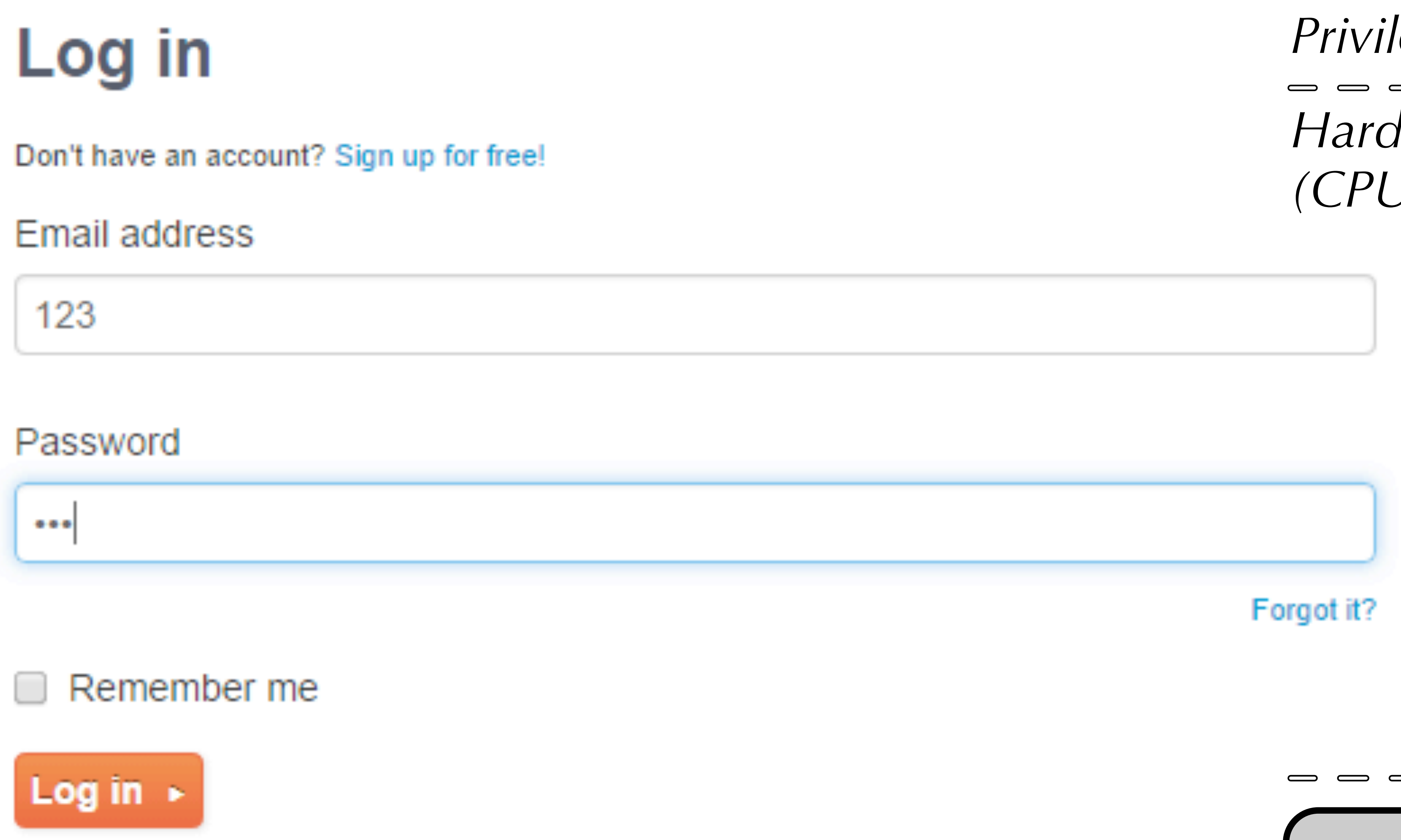
1. User enters password field context - TA generates a symmetric key ( ) and programs security policy into Controller.

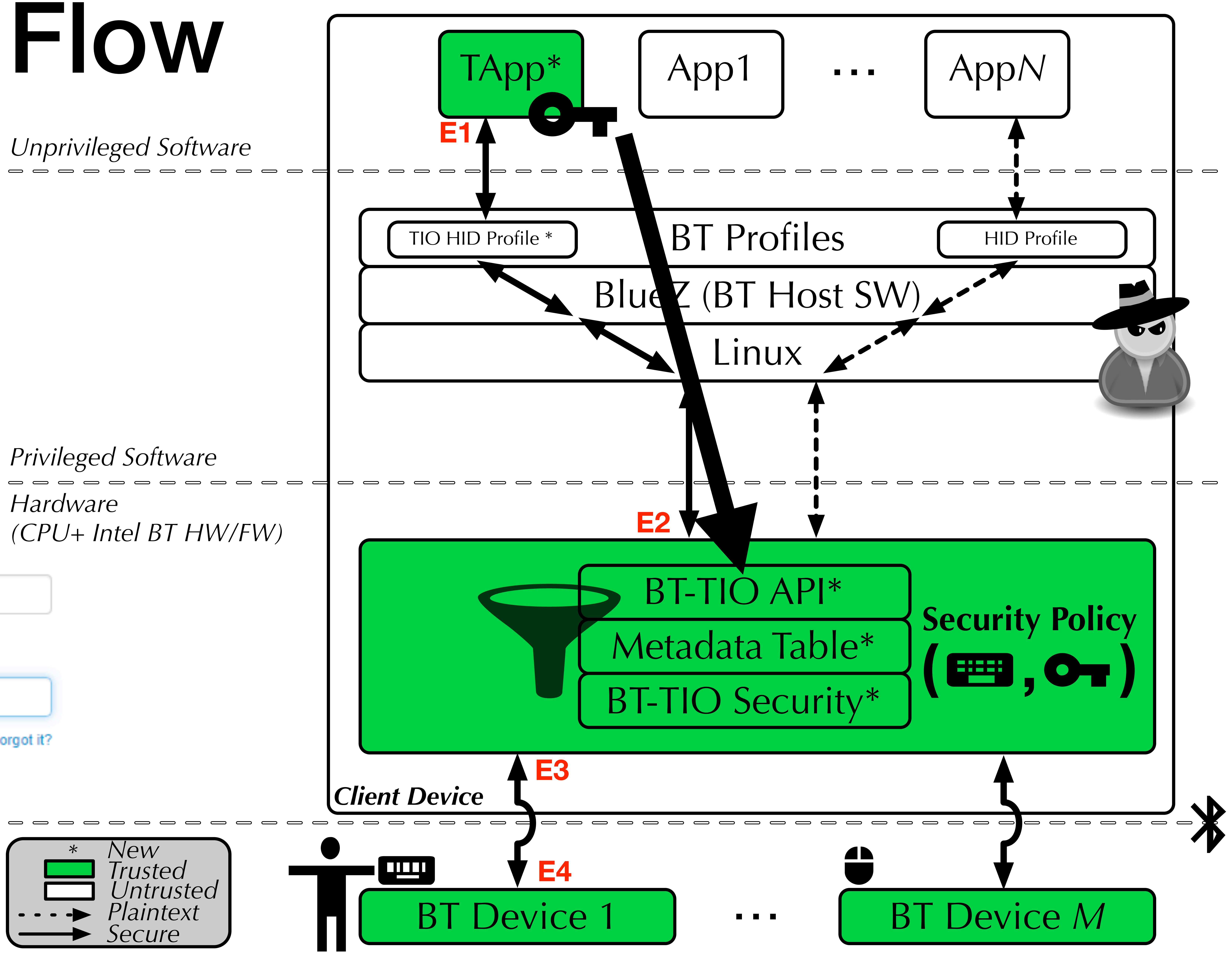




Secure Input Flow

1. User enters password field context - TA generates a symmetric key ( ) and programs security policy into Controller.

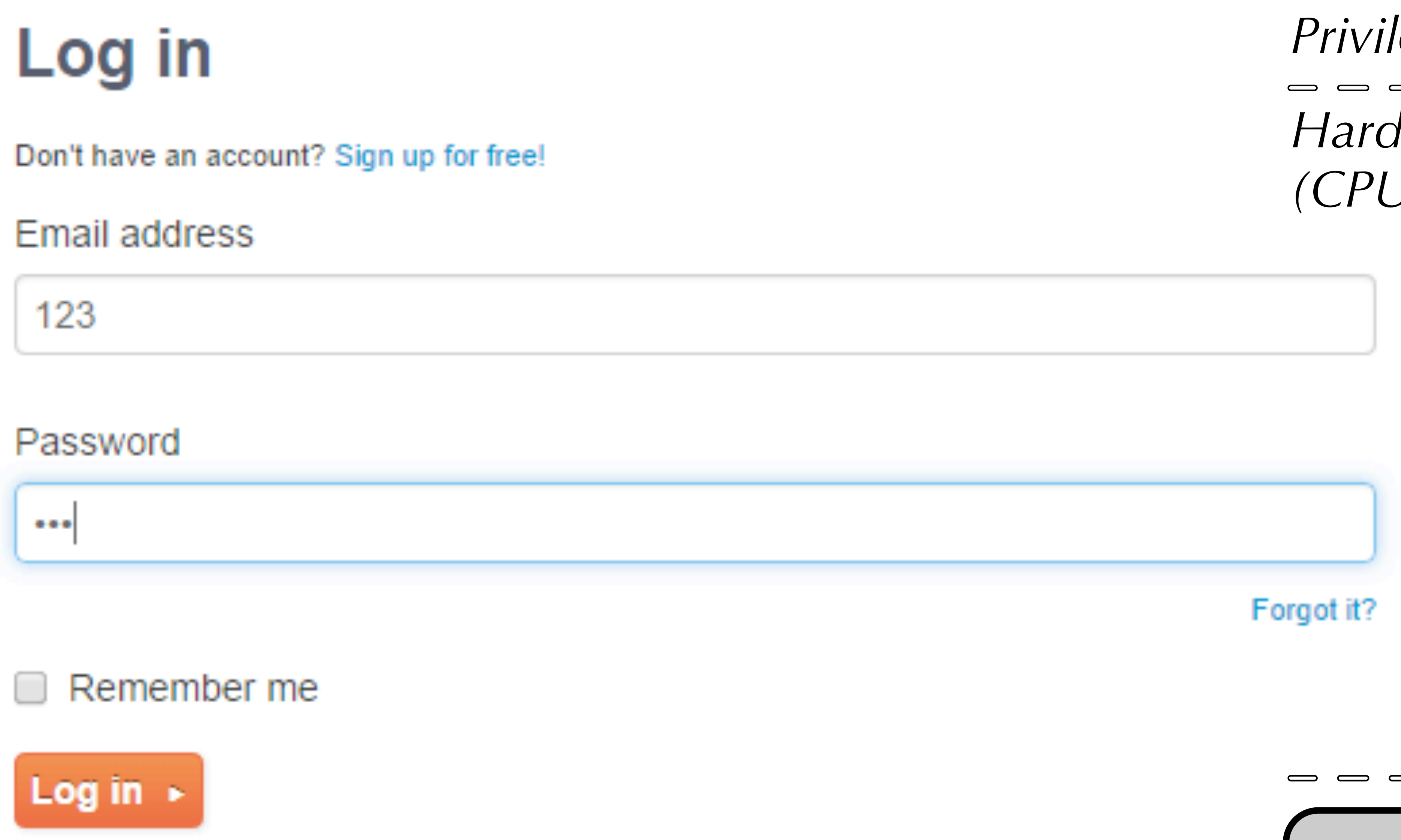


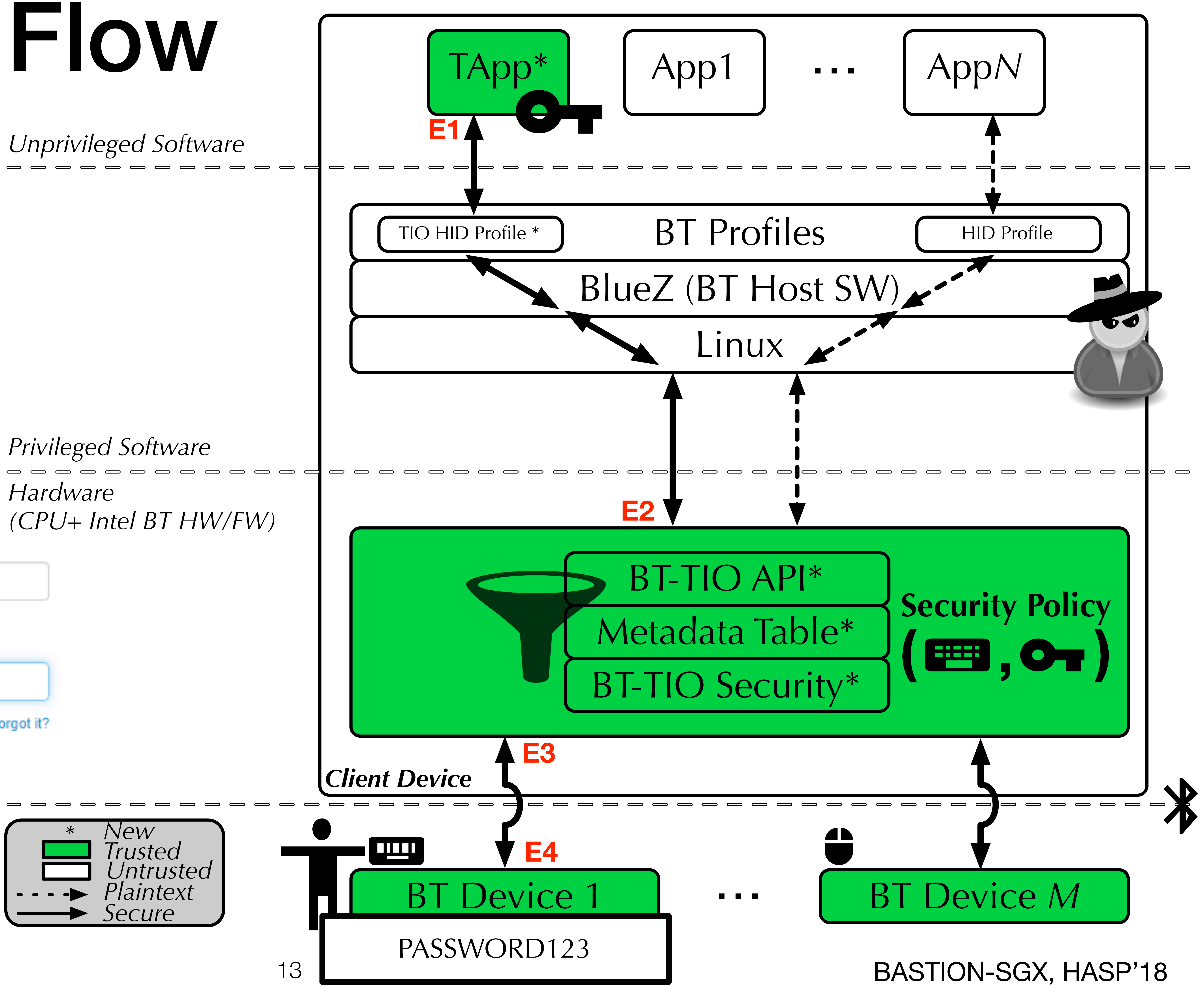


Secure Input Flow

1. User enters password field context - TA generates a symmetric key ( ) and programs security policy into Controller.

2. User types password

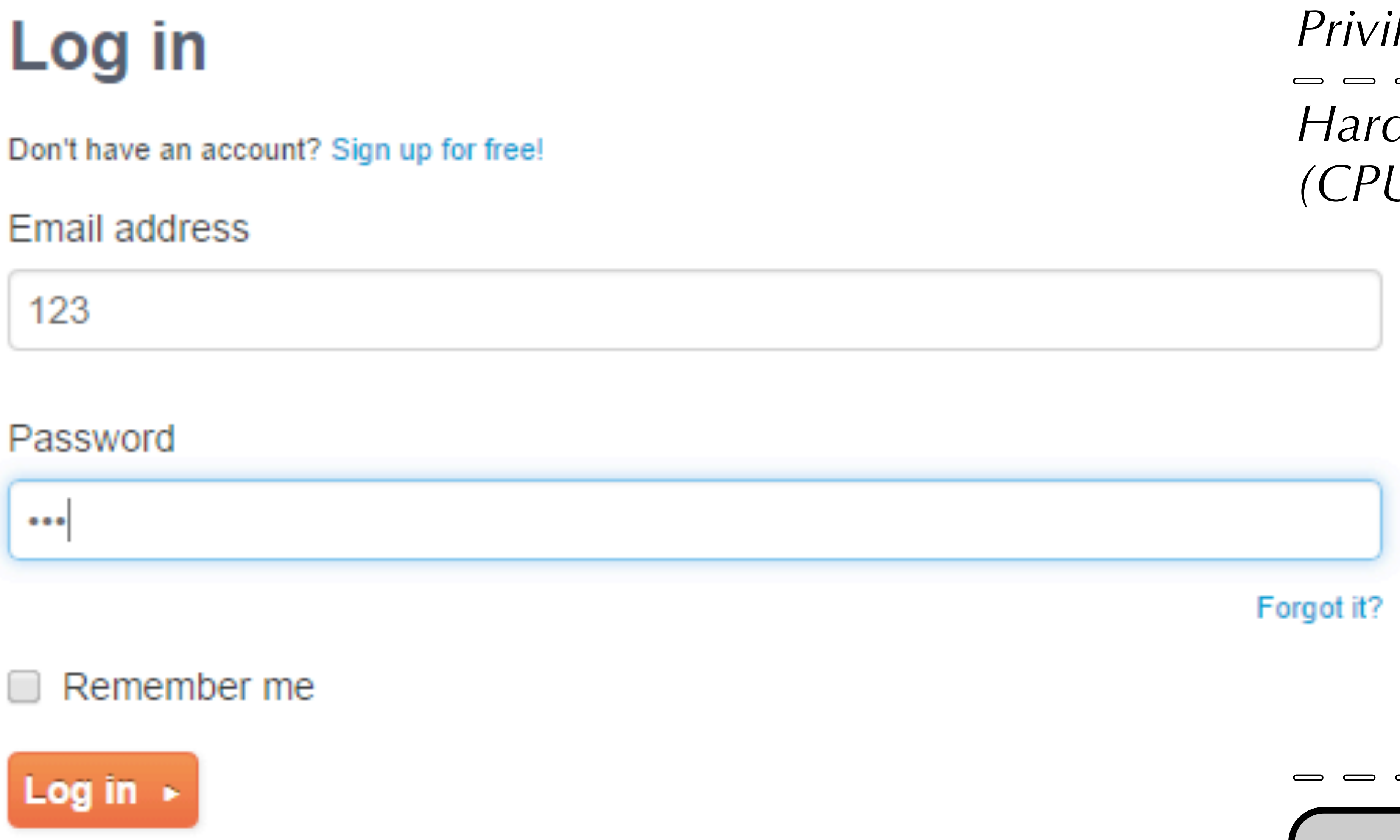


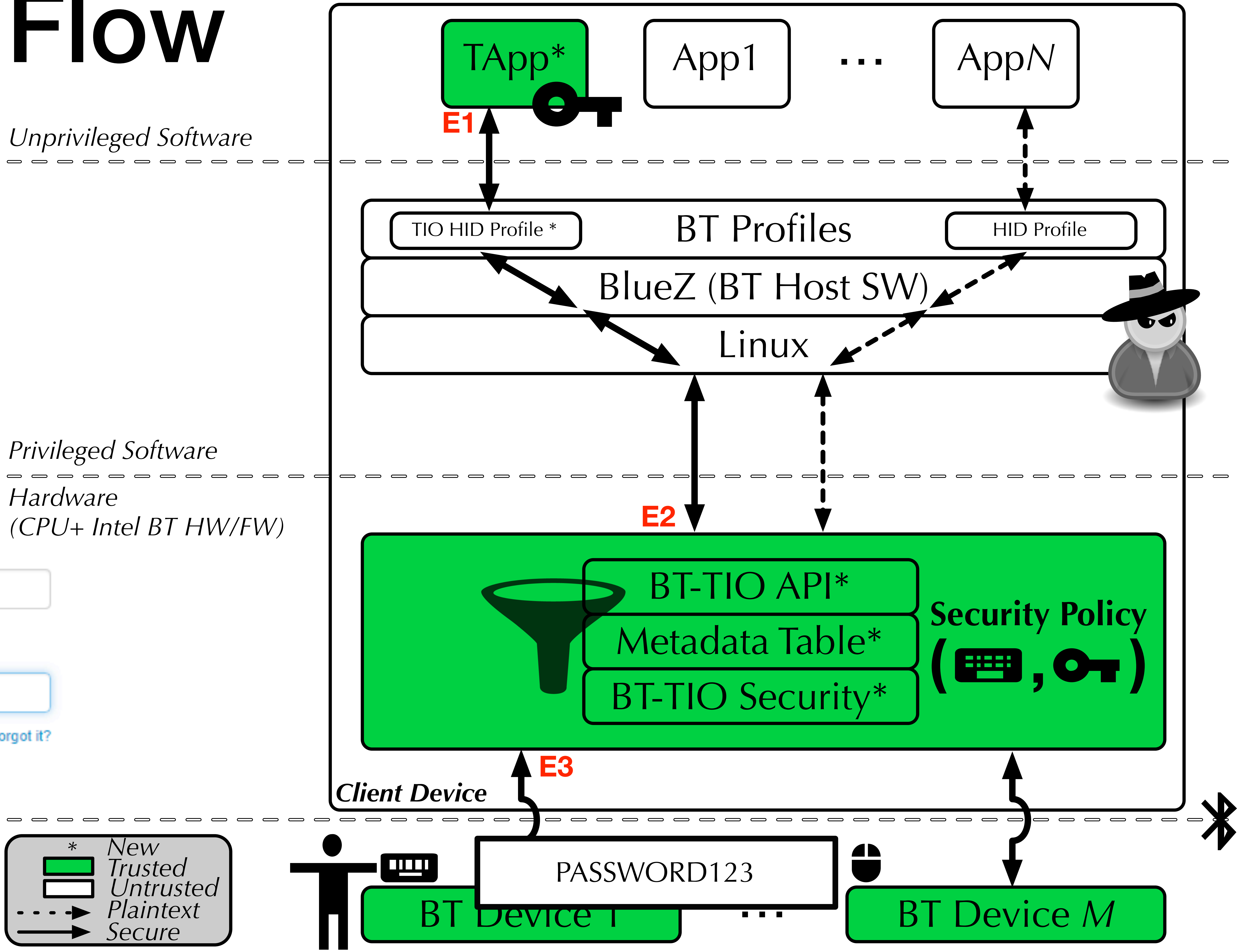


Secure Input Flow

1. User enters password field context - TA generates a symmetric key ( ) and programs security policy into Controller.

2. User types password

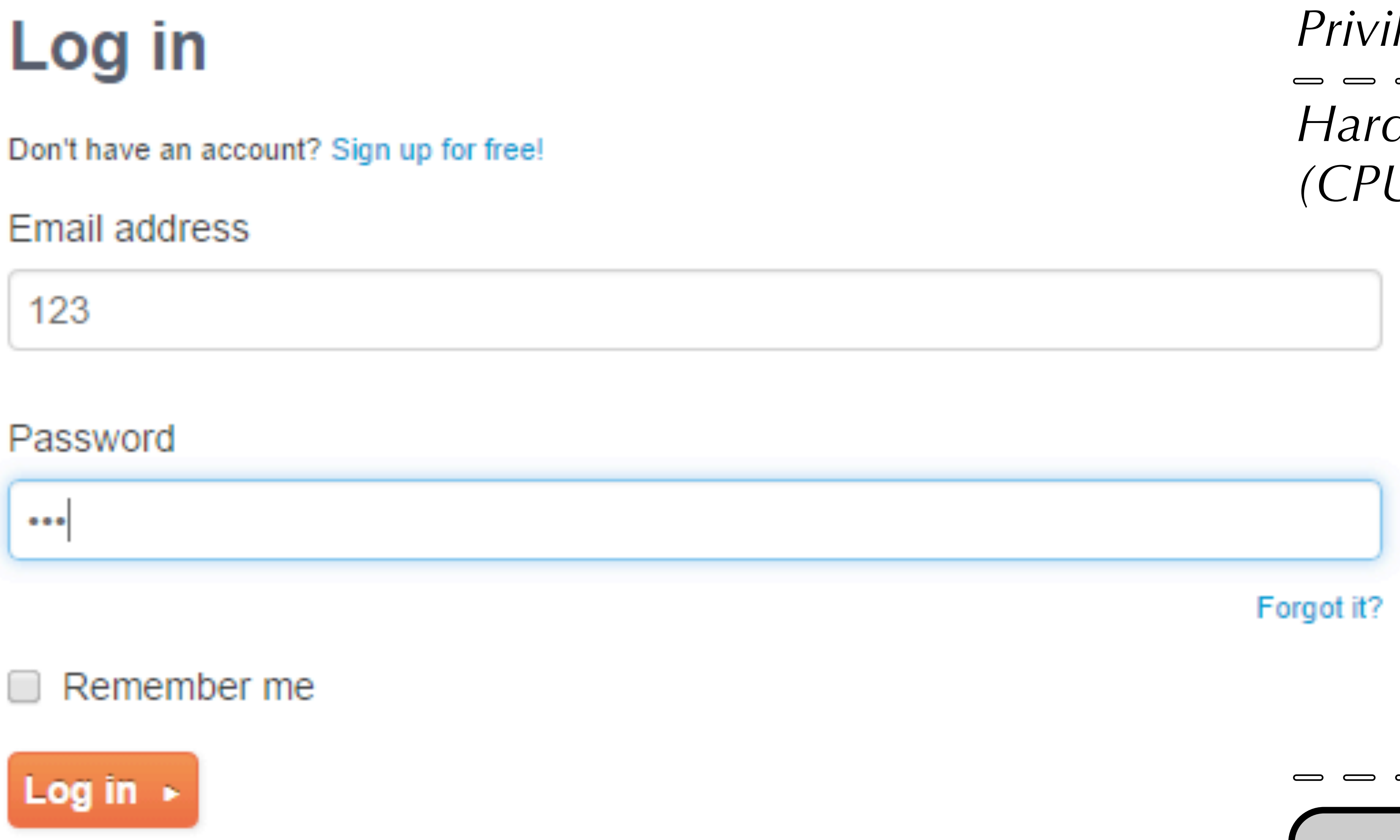


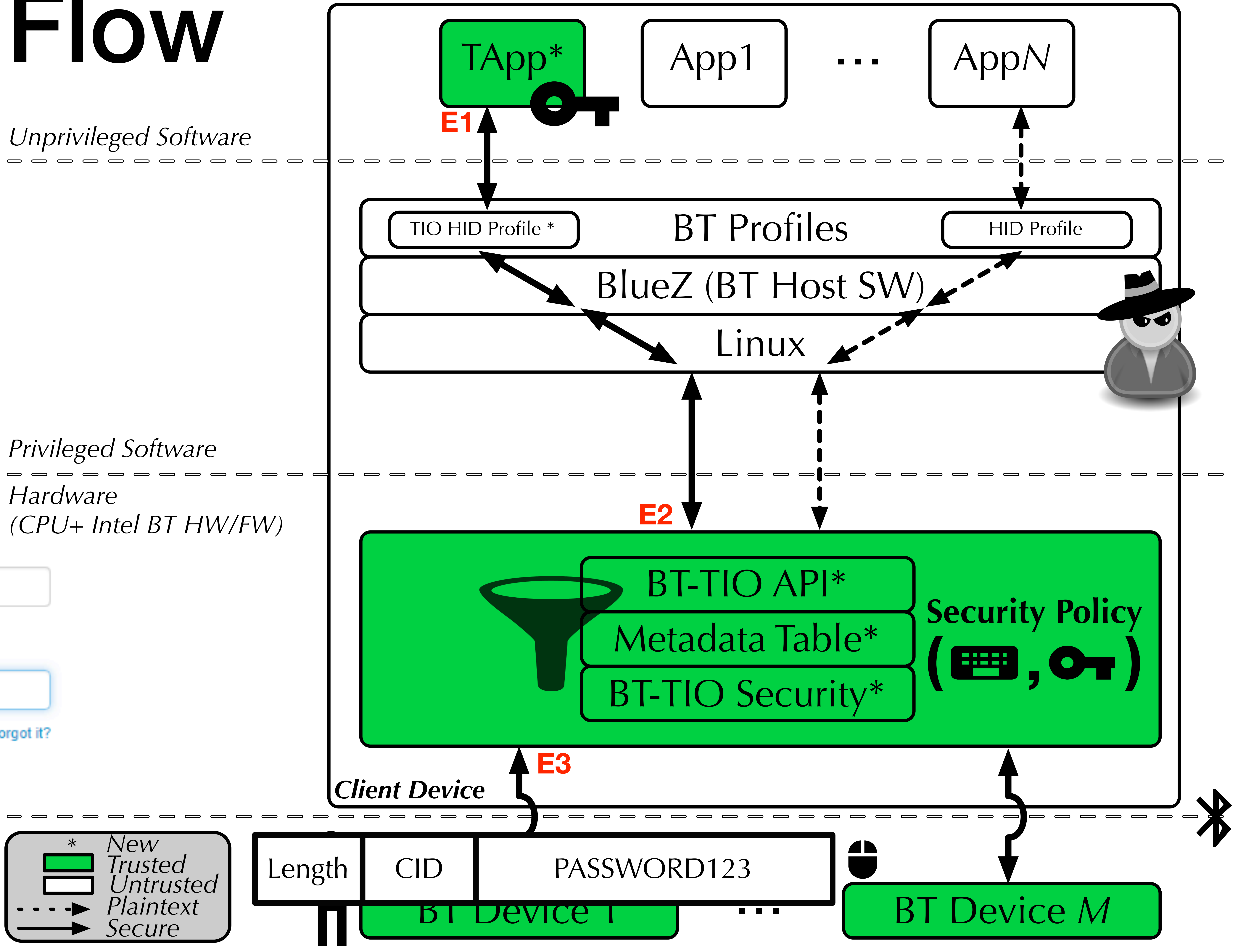


Secure Input Flow

1. User enters password field context - TA generates a symmetric key ( ) and programs security policy into Controller.

2. User types password

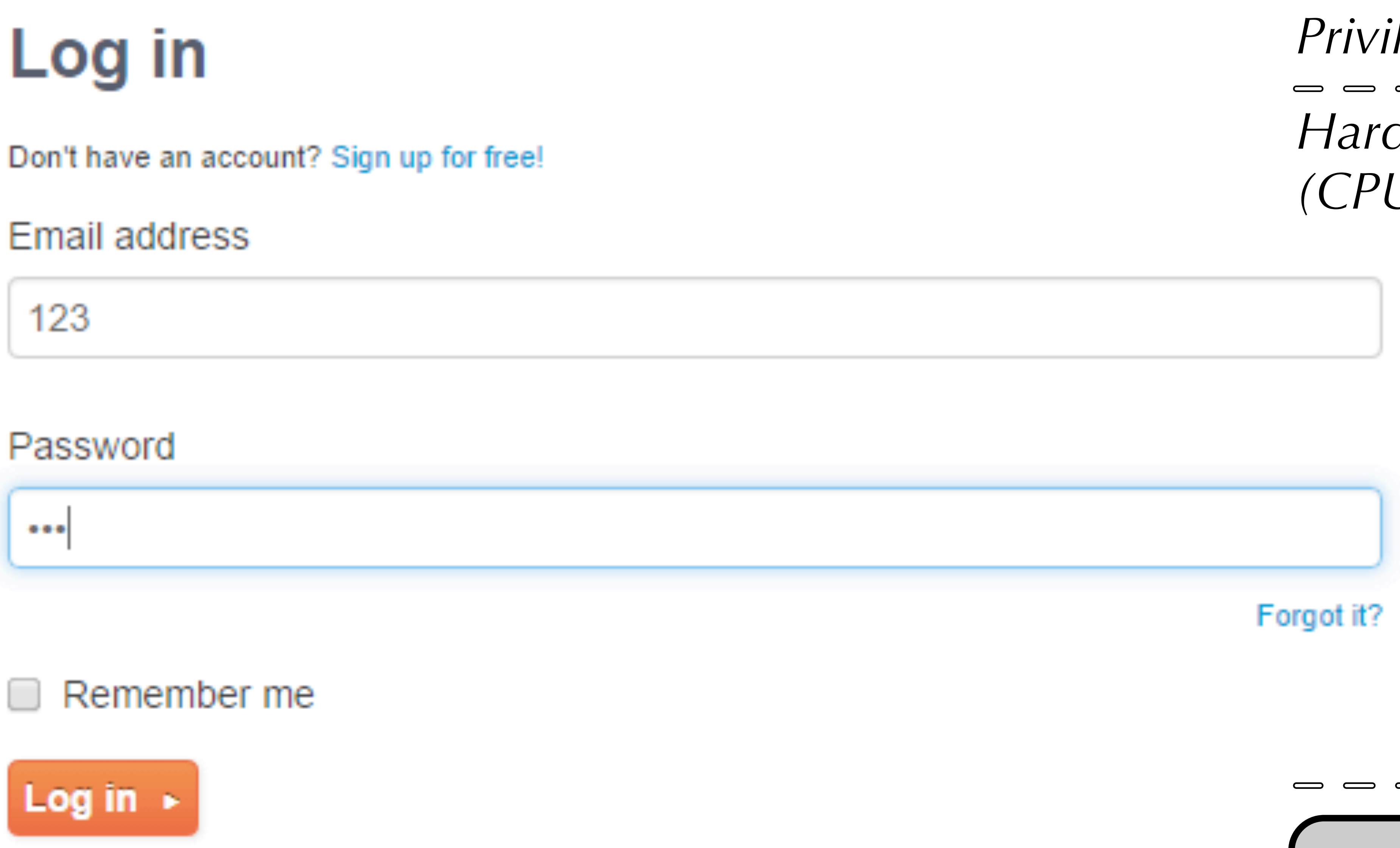


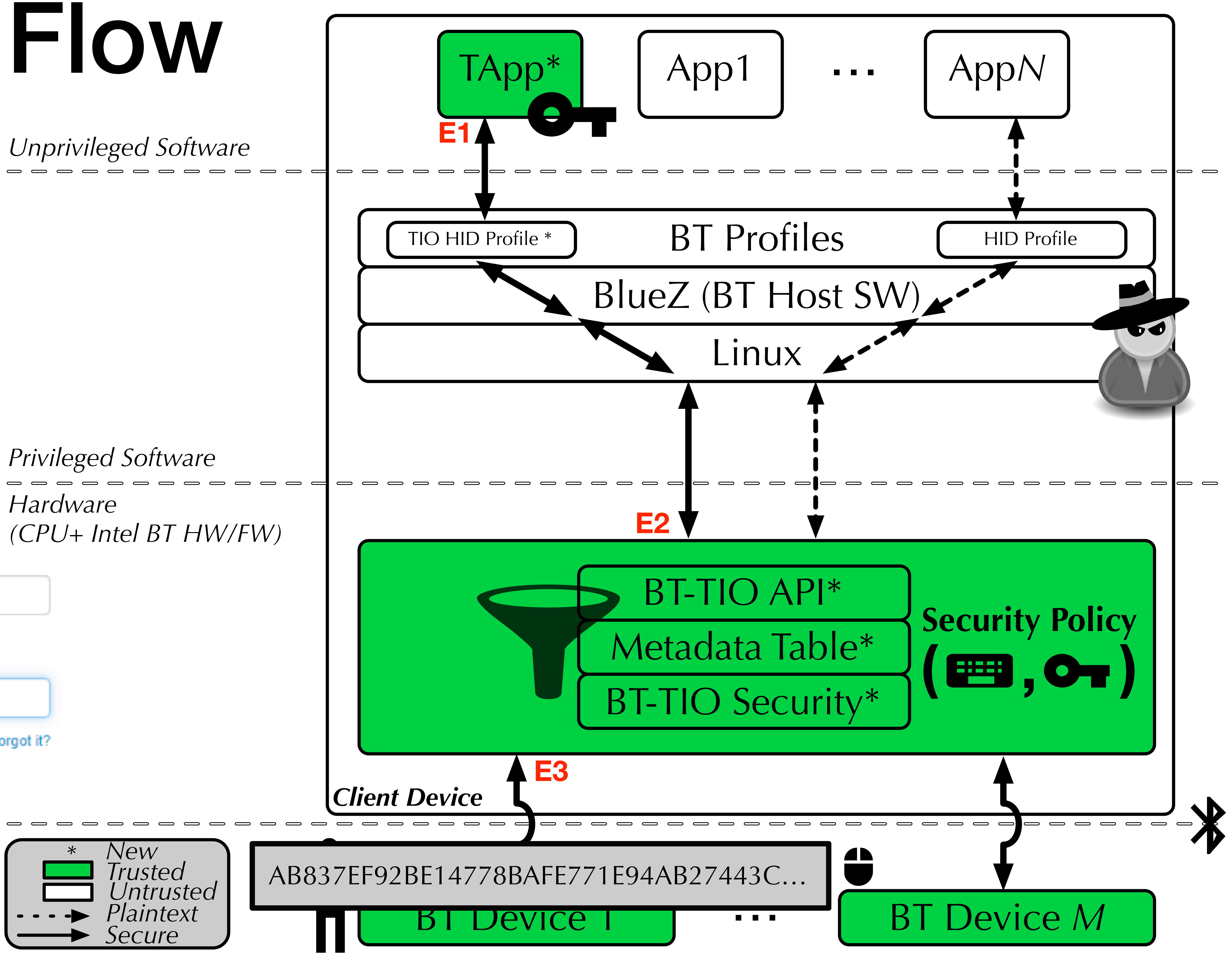


Secure Input Flow

1. User enters password field context - TA generates a symmetric key ( ) and programs security policy into Controller.

2. User types password

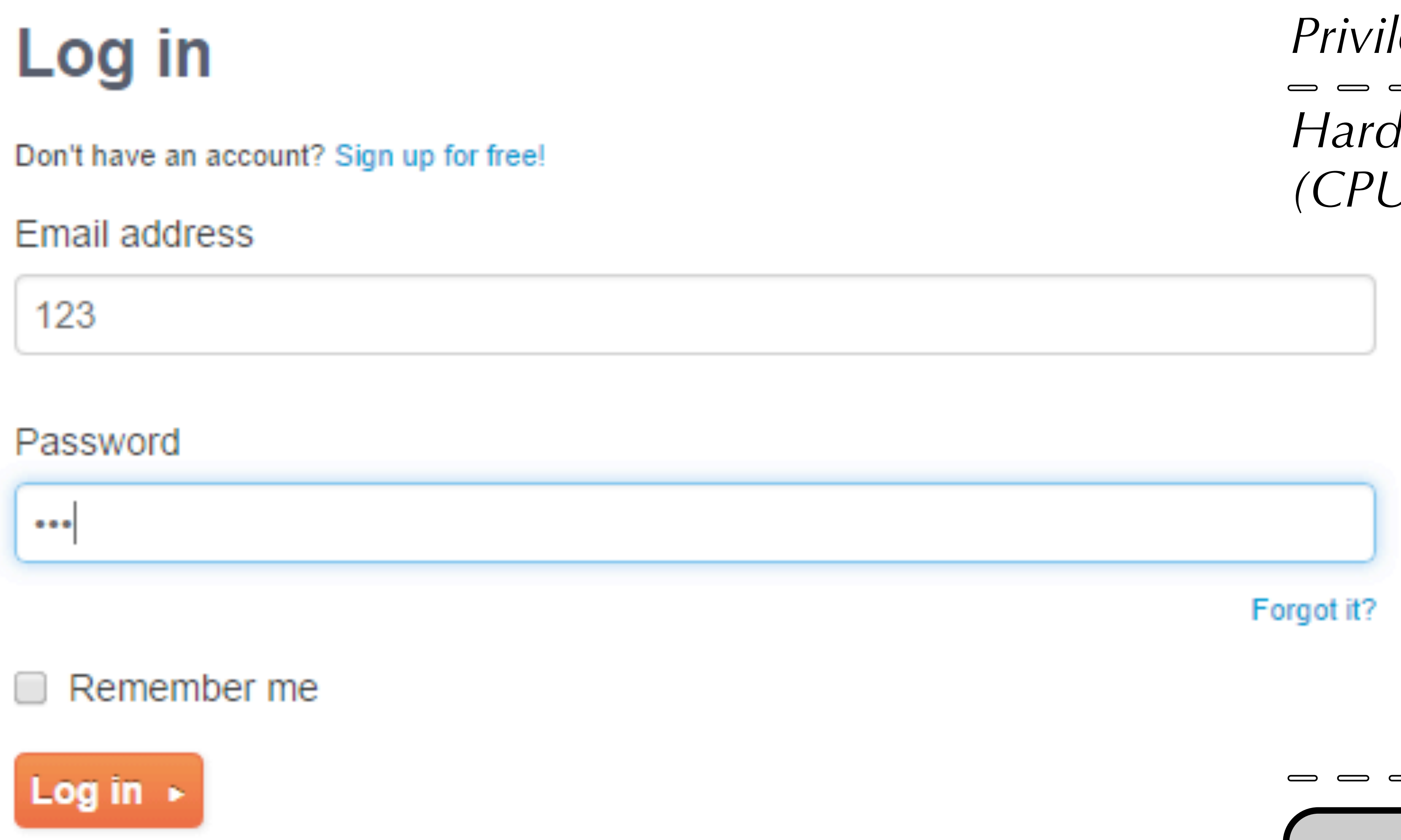


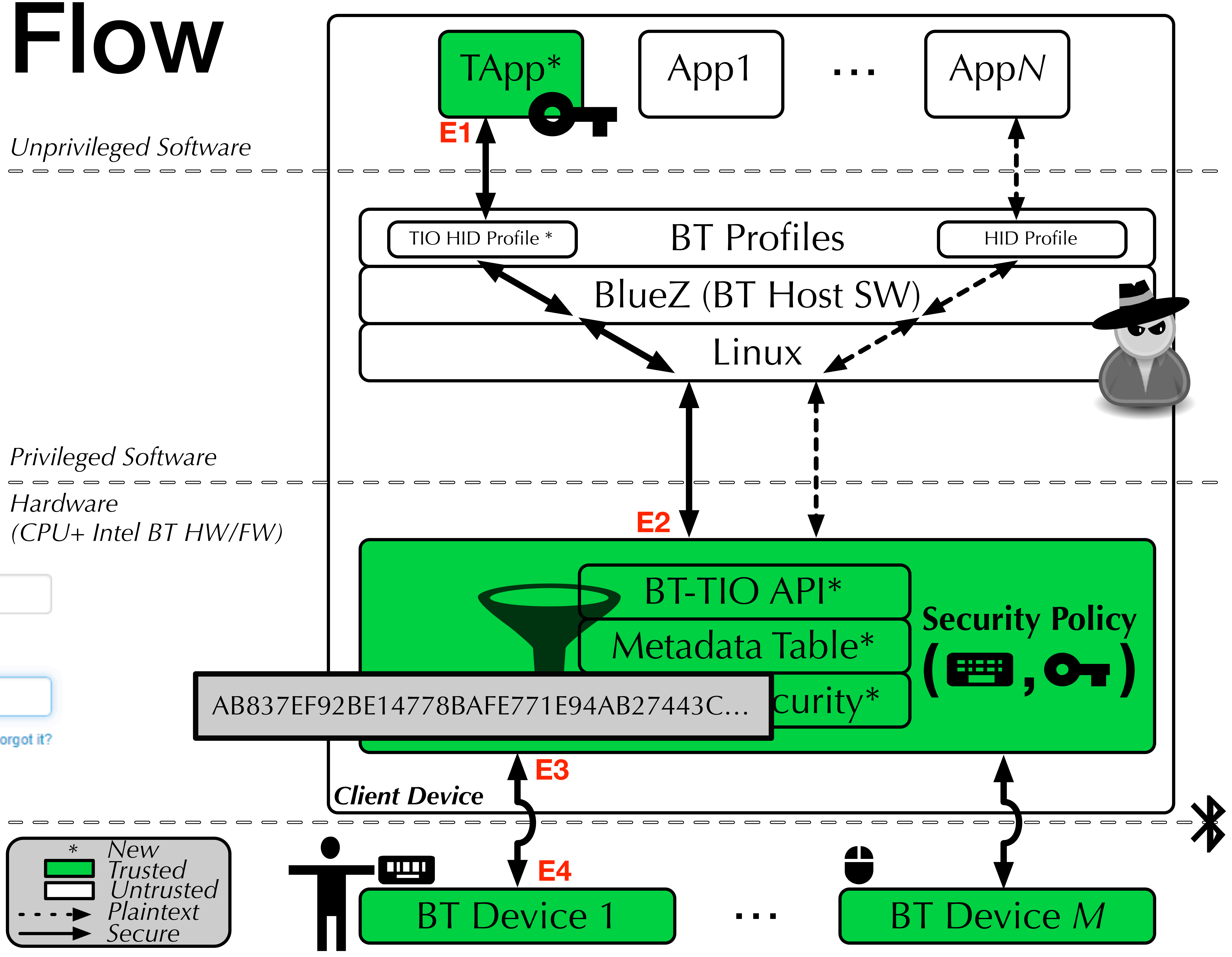


Secure Input Flow

1. User enters password field context - TA generates a symmetric key ( ) and programs security policy into Controller.

2. User types password





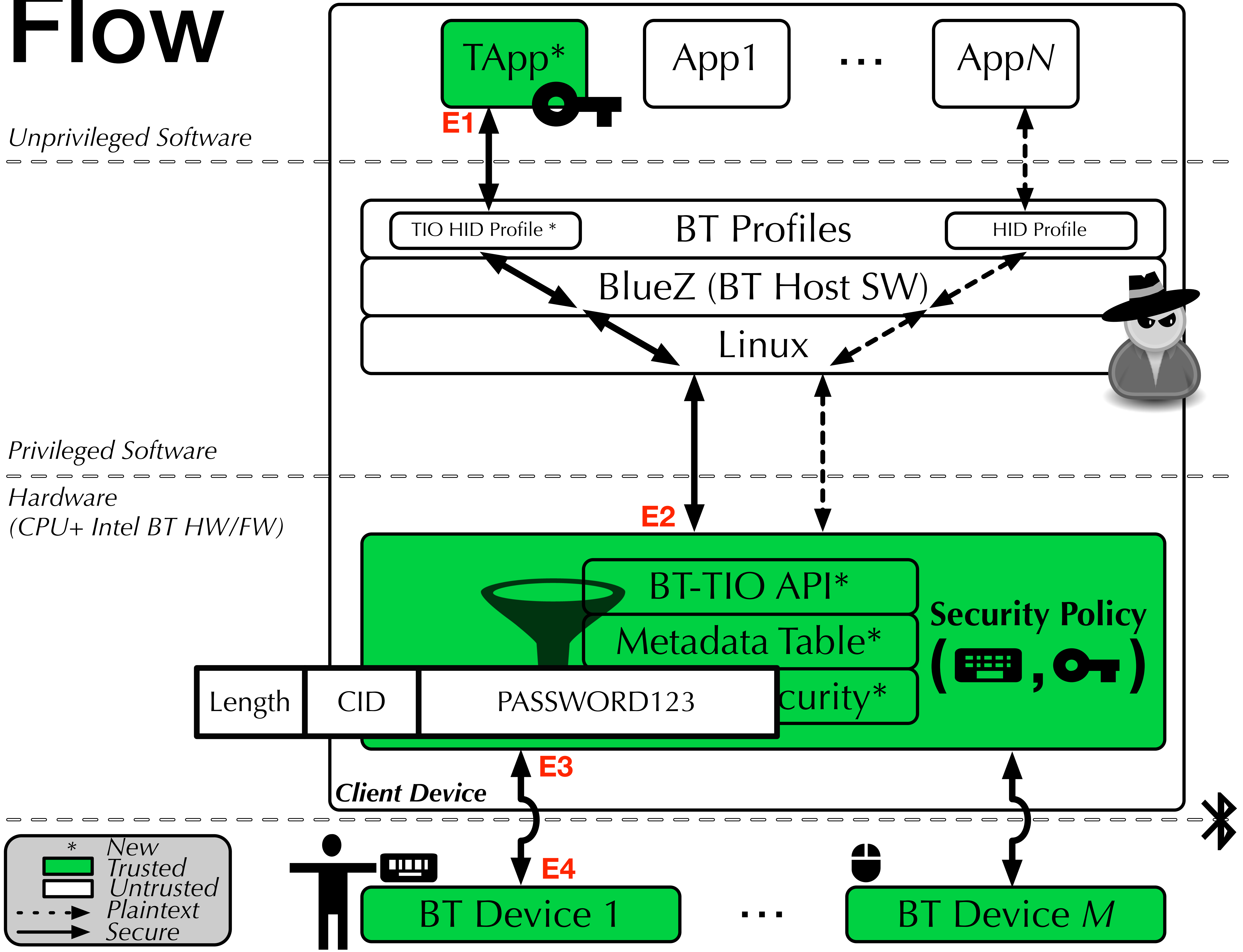
Secure Input Flow

1. User enters password field context - TA generates a symmetric key ( ) and programs security policy into Controller.

2. User types password

3. Controller filters packets matching *any* programmed security policy (,).

Matching packets are sent to BT-TIO host security module before transporting to host SW (use to secure payload).



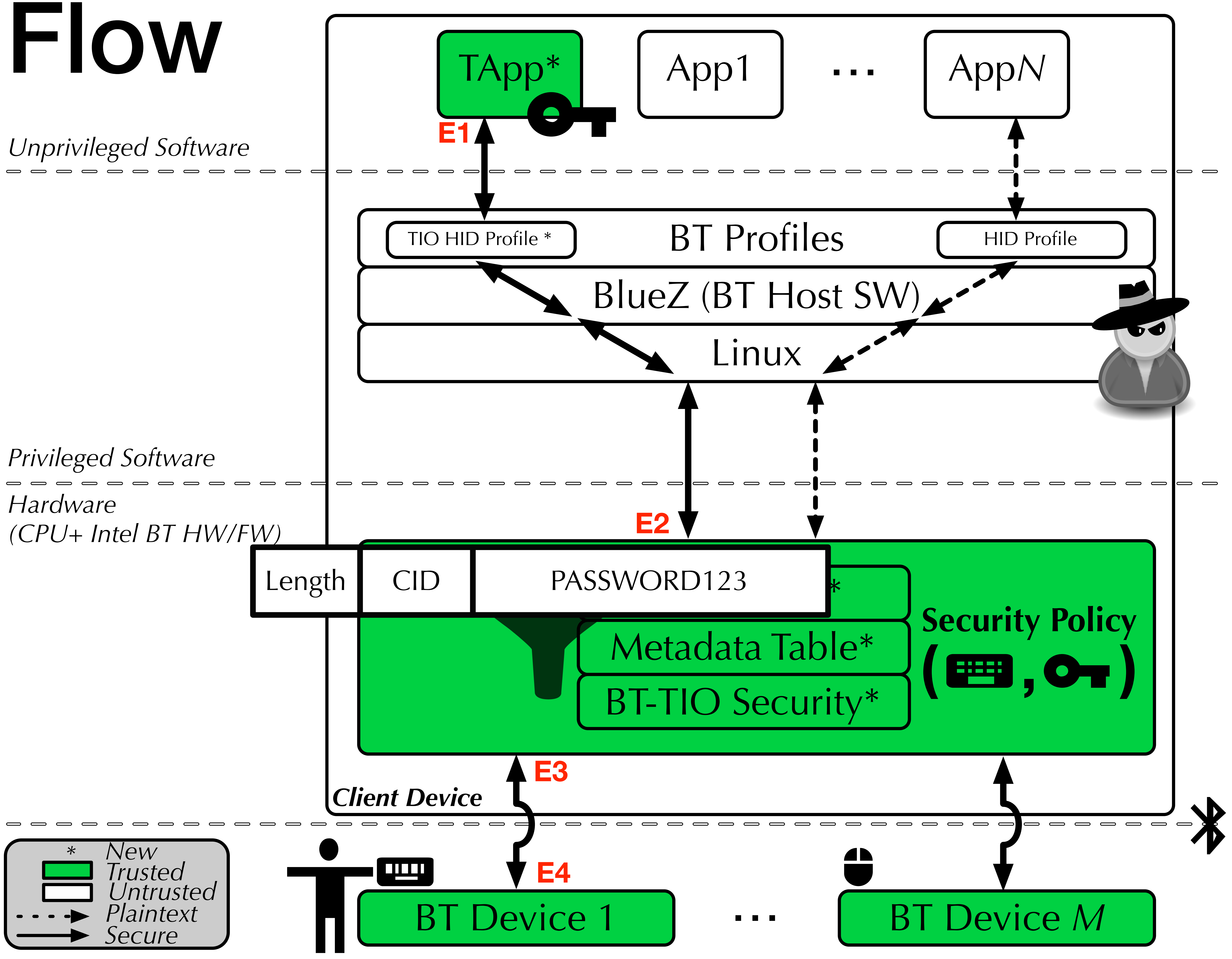
Secure Input Flow

1. User enters password field context - TA generates a symmetric key ( ) and programs security policy into Controller.

2. User types password

3. Controller filters packets matching *any* programmed security policy (,).

Matching packets are sent to BT-TIO security module before transporting to host SW (use to secure payload).



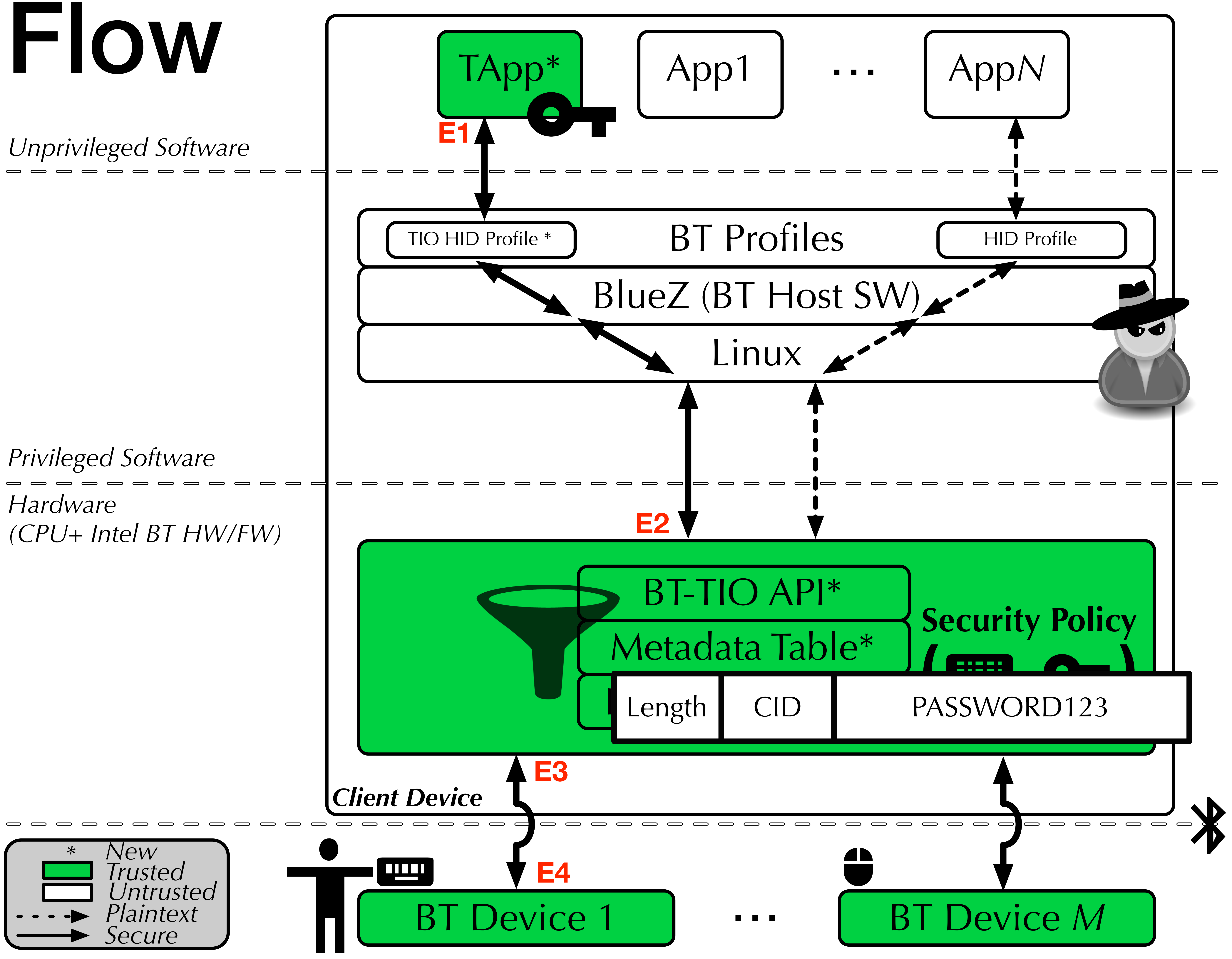
Secure Input Flow

1. User enters password field context - TA generates a symmetric key (🔑) and programs security policy into Controller.

2. User types password

3. Controller filters packets matching *any* programmed security policy (⌨, 🔑).

Matching packets are sent to BT-TIO security module before transporting to host SW (use 🔑 to secure payload).



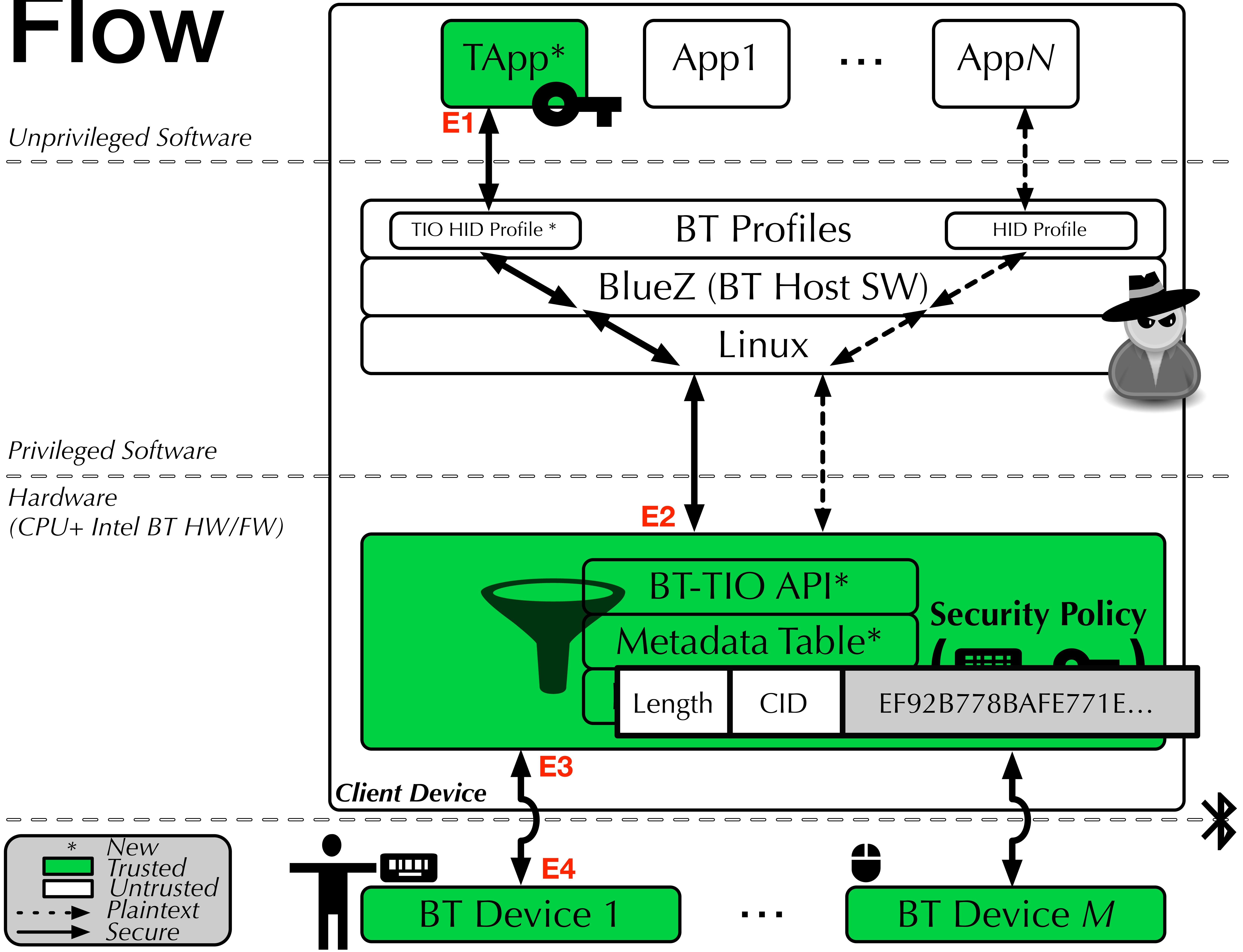
Secure Input Flow

1. User enters password field context - TA generates a symmetric key ( ) and programs security policy into Controller.

2. User types password

3. Controller filters packets matching *any* programmed security policy (,).

Matching packets are sent to BT-TIO security module before transporting to host SW (use to secure payload).



Secure Input Flow

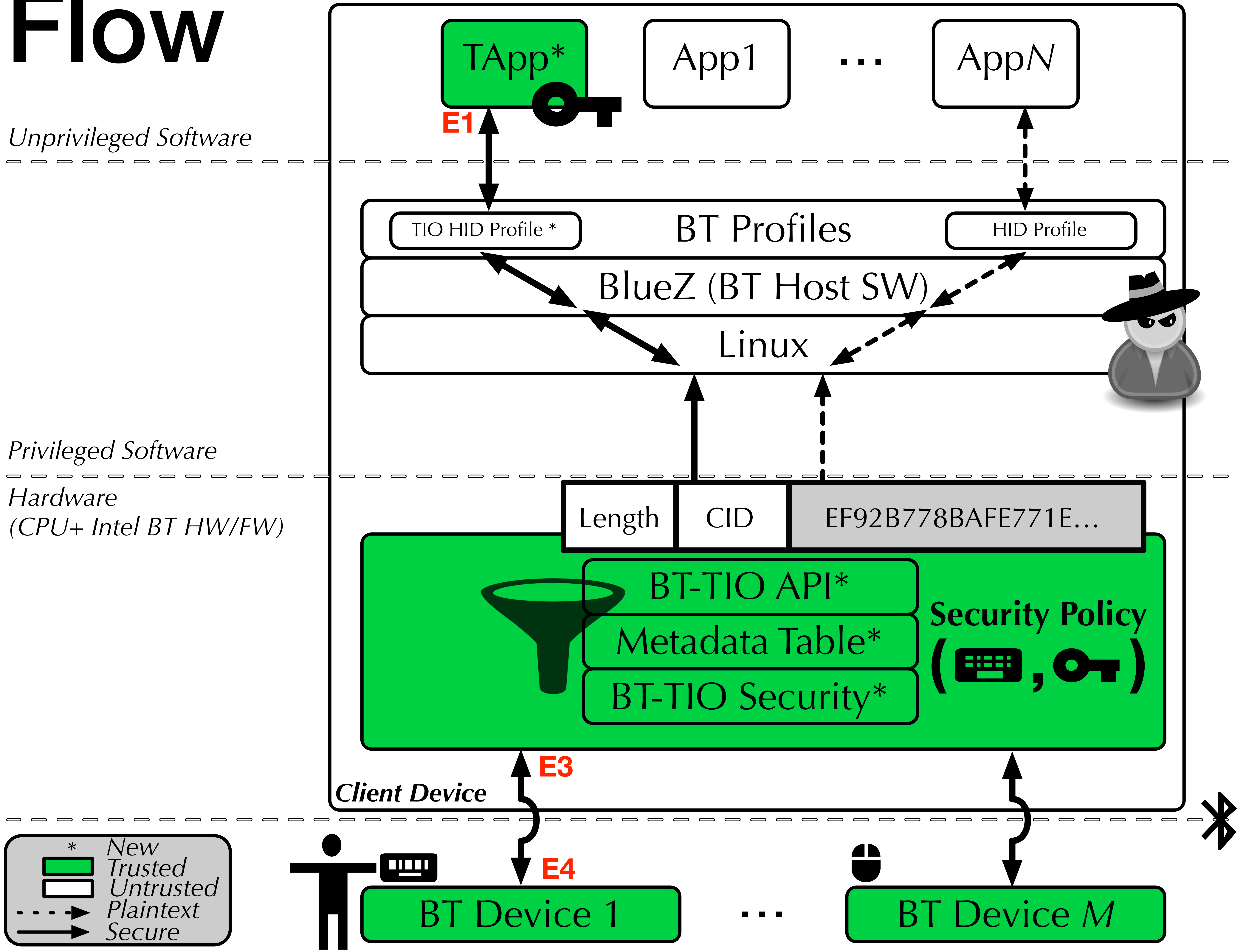
1. User enters password field context - TA generates a symmetric key (🔑) and programs security policy into Controller.

2. User types password

3. Controller filters packets matching *any* programmed security policy (⌨, 🔑).

Matching packets are sent to BT-TIO security module before transporting to host SW (use 🔑 to secure payload).

4. HCI transport and L2CAP routing



Secure Input Flow

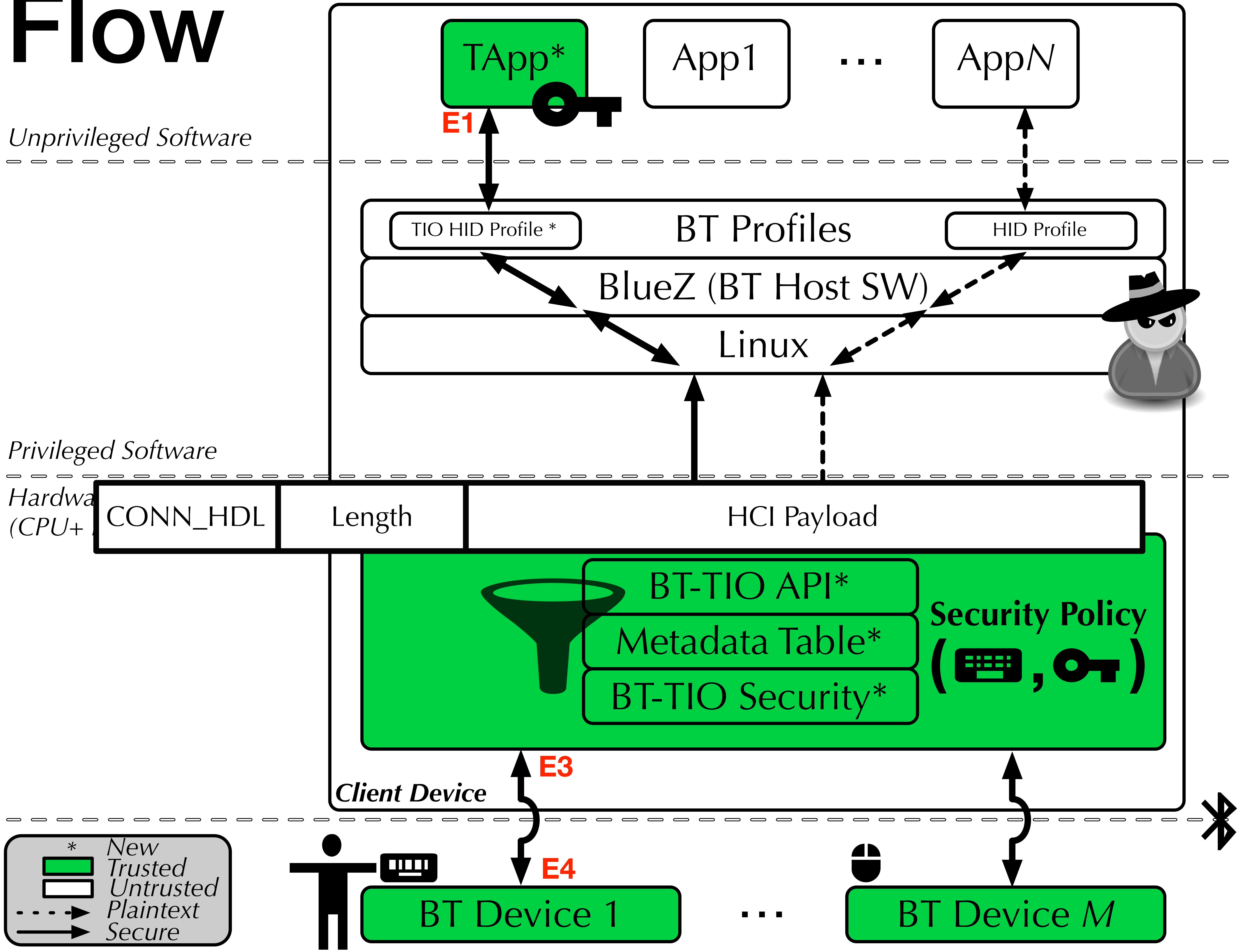
1. User enters password field context - TA generates a symmetric key ( ) and programs security policy into Controller.

2. User types password

3. Controller filters packets matching *any* programmed security policy (,).

Matching packets are sent to BT-TIO host security module before transporting to host SW (use to secure payload).

4. HCI transport and L2CAP routing



Secure Input Flow

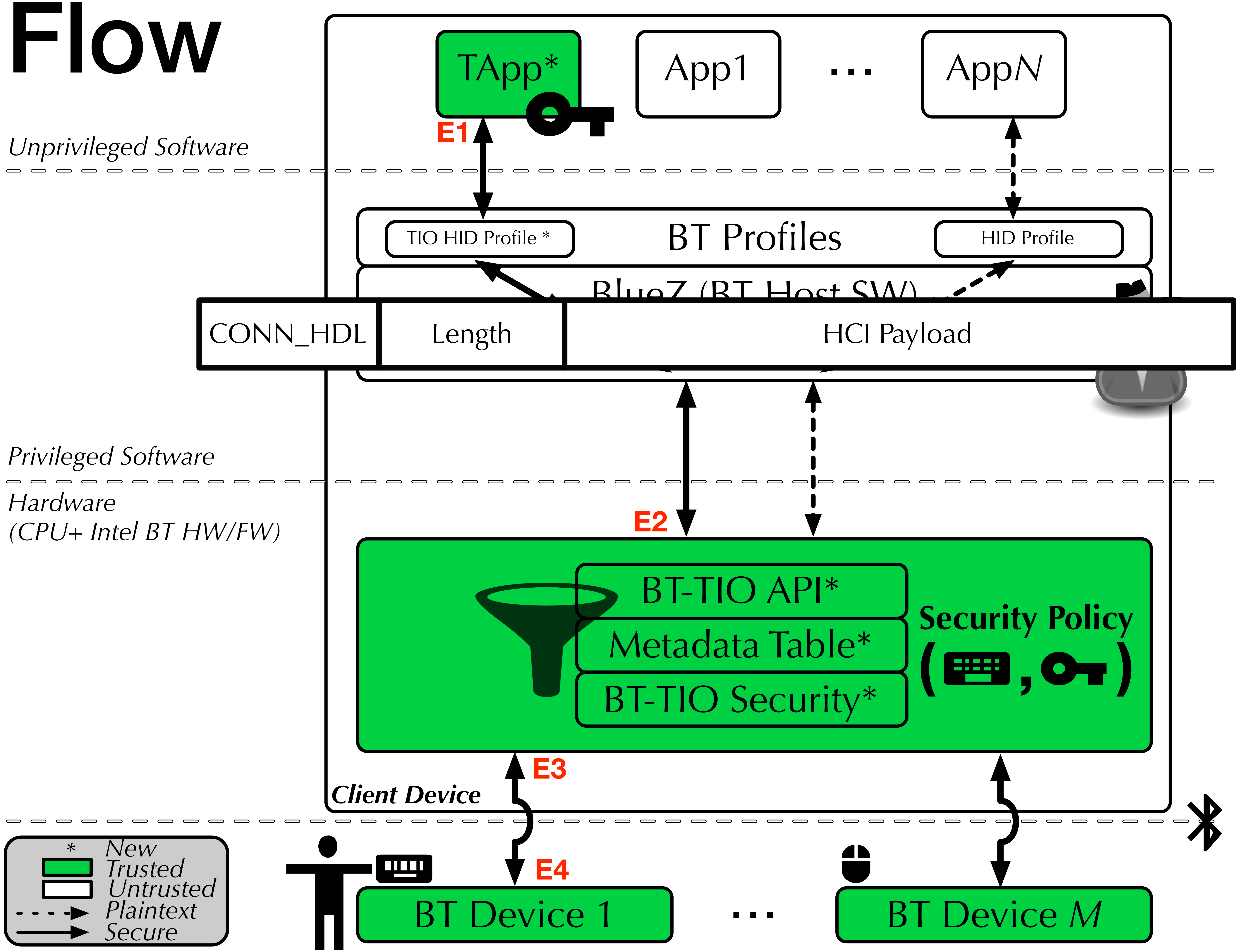
1. User enters password field context - TA generates a symmetric key ( ) and programs security policy into Controller.

2. User types password

3. Controller filters packets matching *any* programmed security policy (,).

Matching packets are sent to BT-TIO security module before transporting to host SW (use to secure payload).

4. HCI transport and L2CAP routing



Secure Input Flow

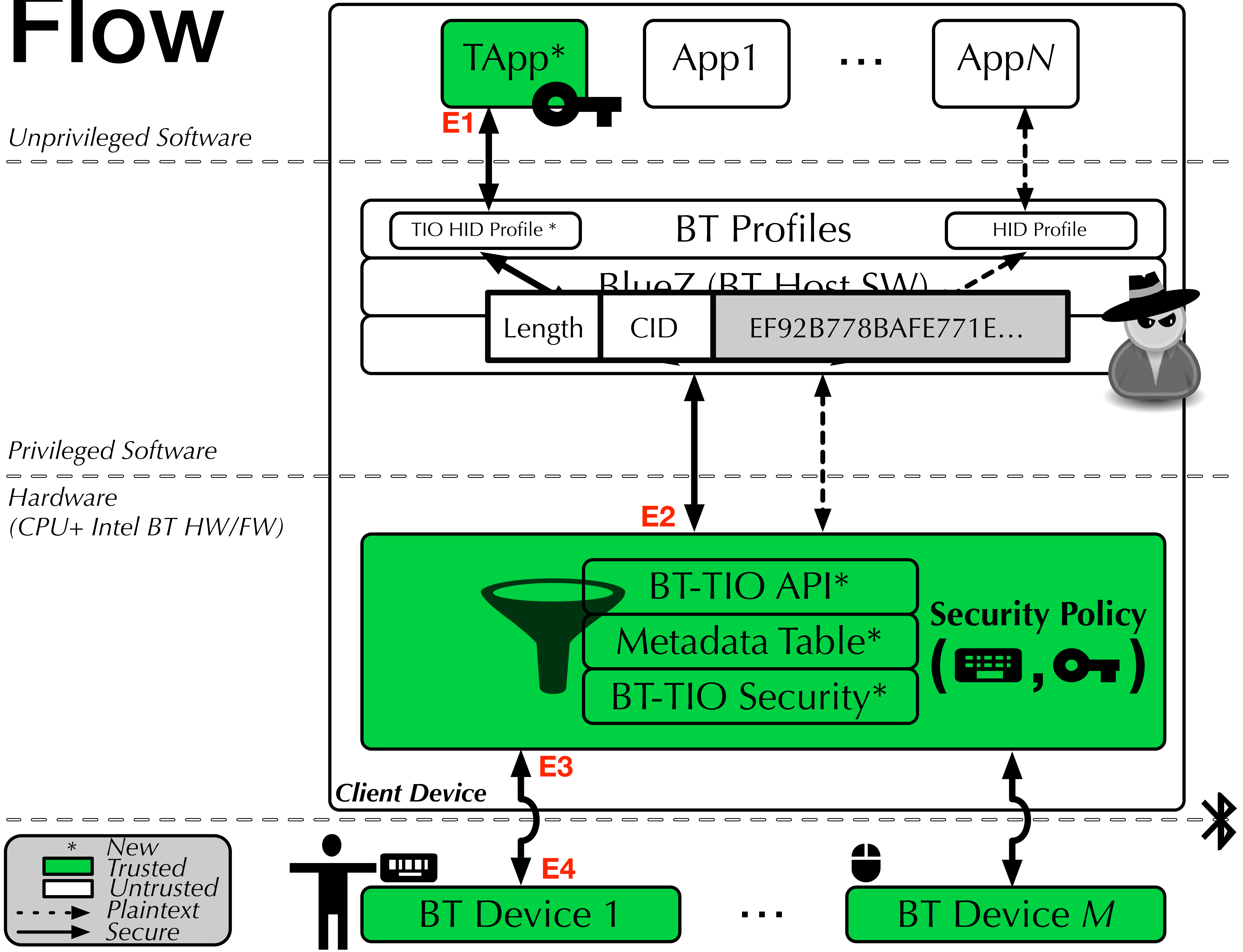
1. User enters password field context - TA generates a symmetric key ( ) and programs security policy into Controller.

2. User types password

3. Controller filters packets matching *any* programmed security policy (,).

Matching packets are sent to BT-TIO security module before transporting to host SW (use to secure payload).

4. HCI transport and L2CAP routing



Secure Input Flow

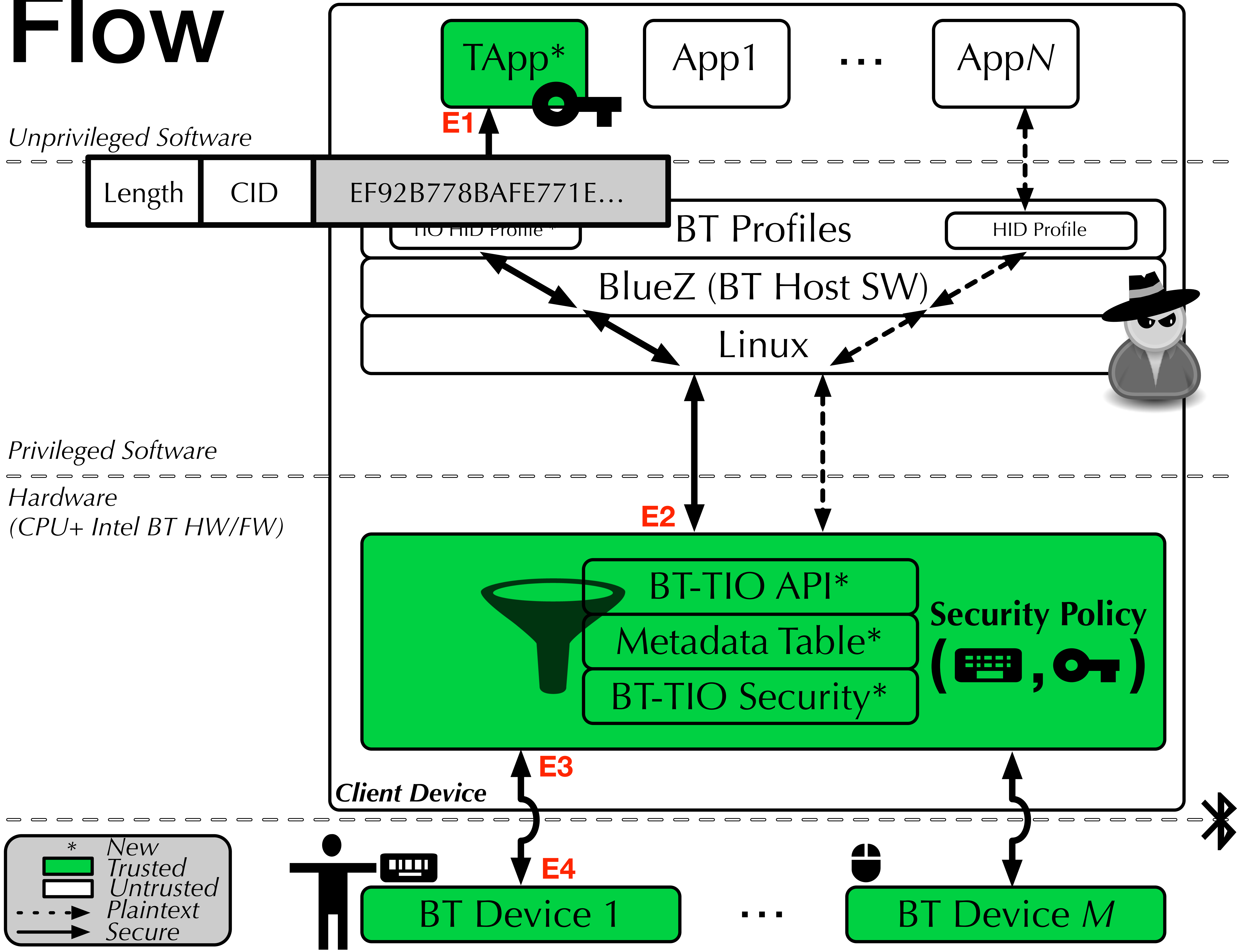
1. User enters password field context - TA generates a symmetric key ( ) and programs security policy into Controller.

2. User types password

3. Controller filters packets matching *any* programmed security policy (,).

Matching packets are sent to BT-TIO security module before transporting to host SW (use to secure payload).

4. HCI transport and L2CAP routing



Travis Peters (traviswp@cs.dartmouth.edu)

Secure Input Flow

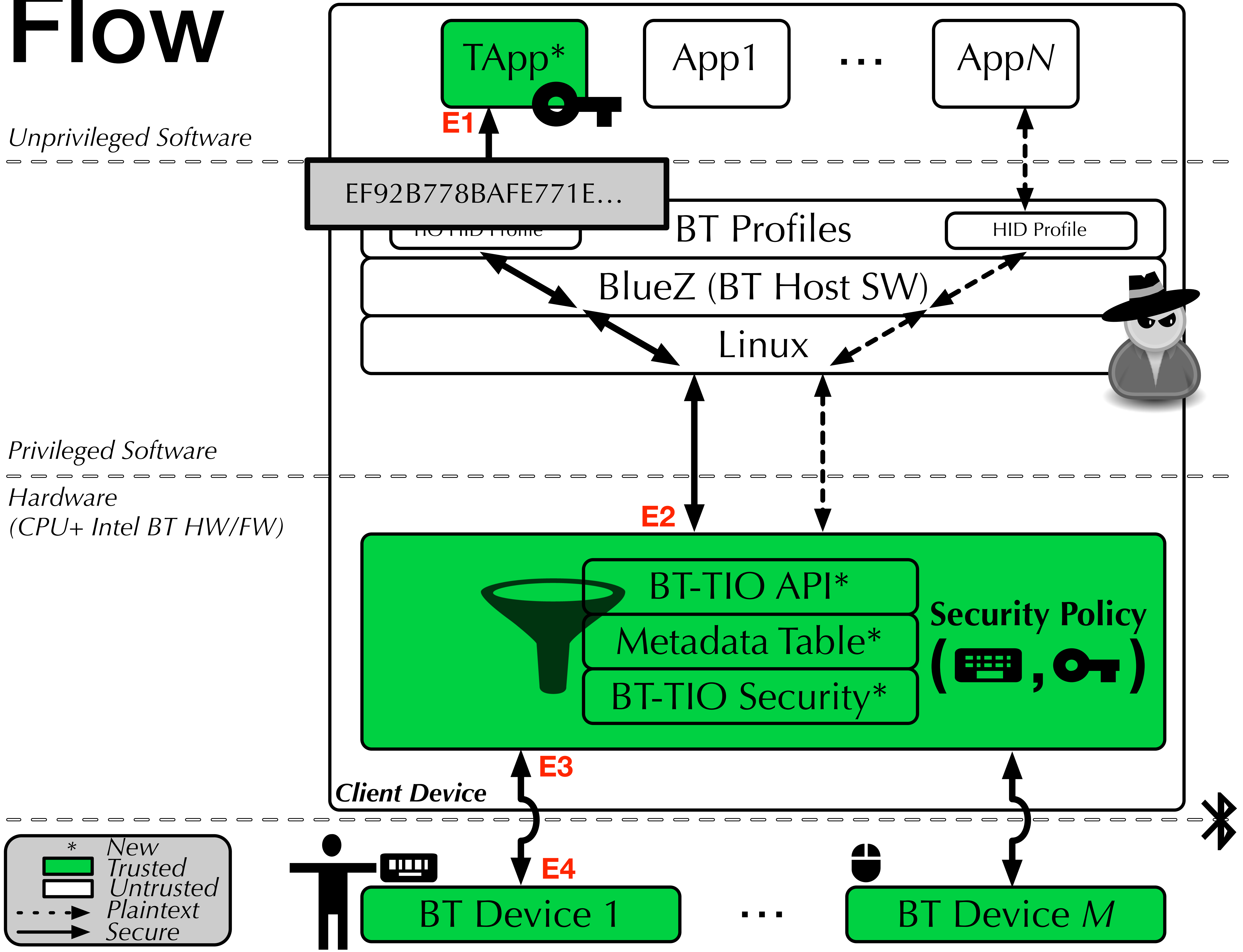
1. User enters password field context - TA generates a symmetric key (🔑) and programs security policy into Controller.

2. User types password

3. Controller filters packets matching *any* programmed security policy (⌨, 🔑).

Matching packets are sent to BT-TIO security module before transporting to host SW (use 🔑 to secure payload).

4. HCI transport and L2CAP routing



Secure Input Flow

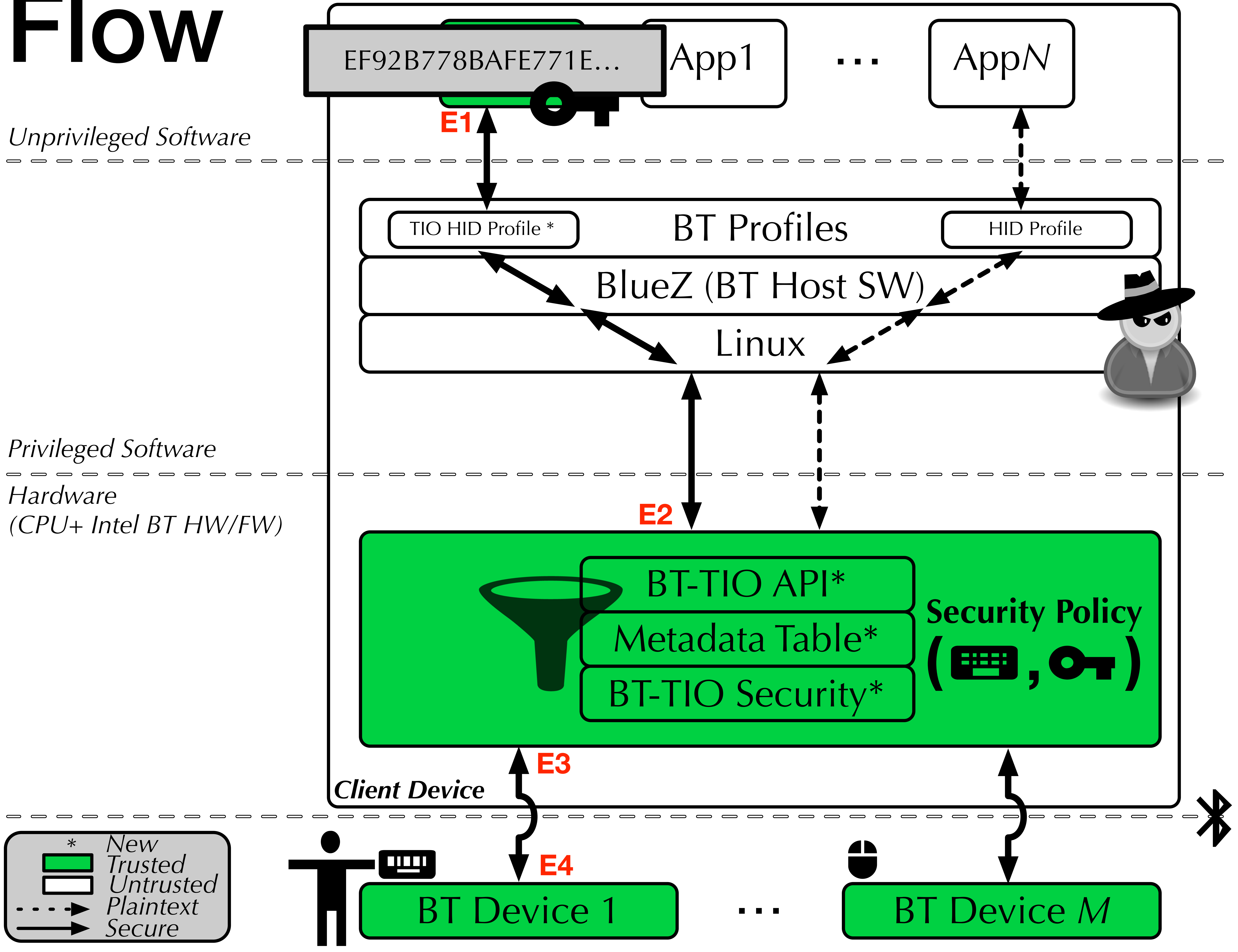
1. User enters password field context - TA generates a symmetric key ( ) and programs security policy into Controller.

2. User types password

3. Controller filters packets matching *any* programmed security policy (,).

Matching packets are sent to BT-TIO security module before transporting to host SW (use to secure payload).

4. HCI transport and L2CAP routing



Secure Input Flow

1. User enters password field context - TA generates a symmetric key ( ) and programs security policy into Controller.

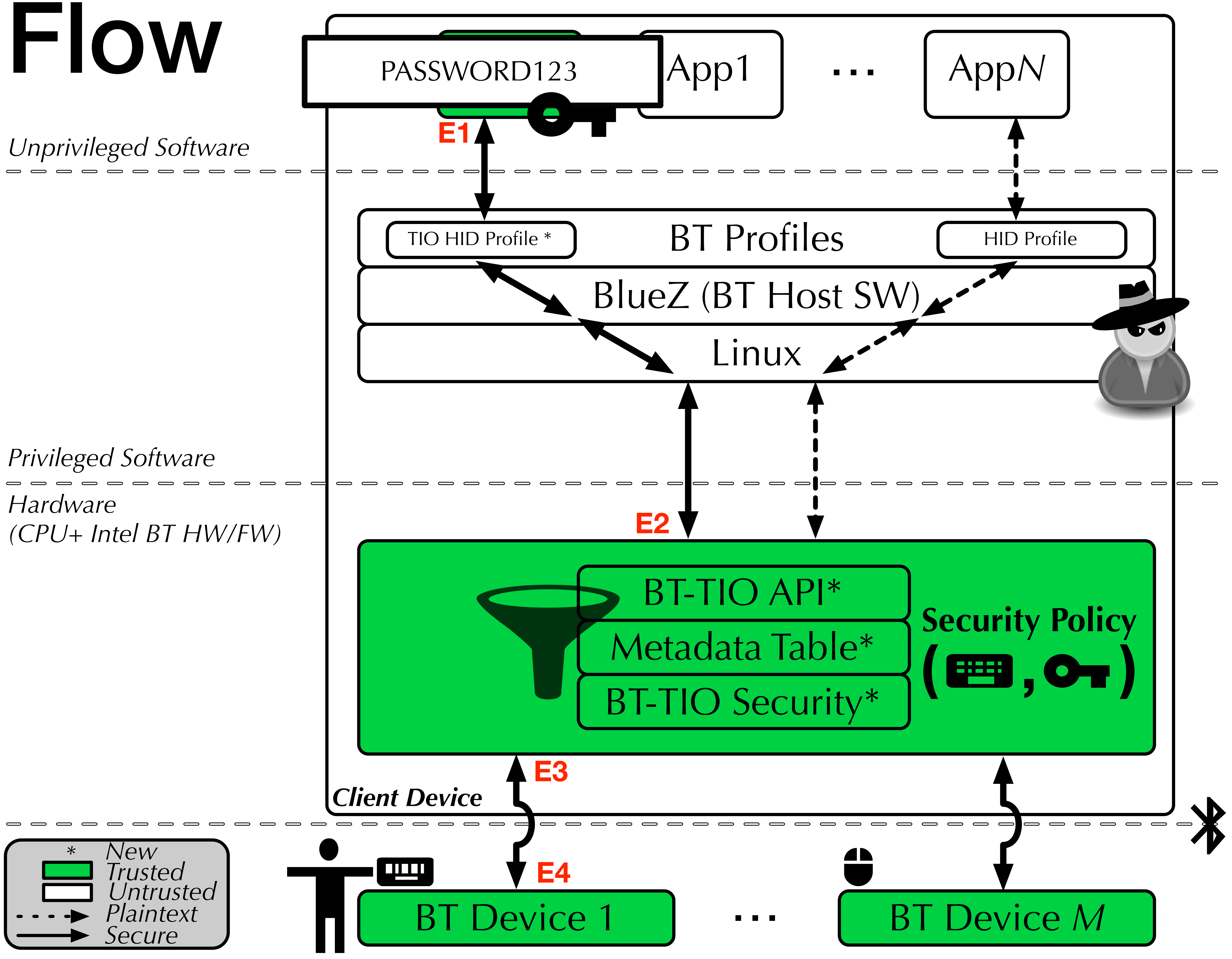
2. User types password

3. Controller filters packets matching *any* programmed security policy (,).

Matching packets are sent to BT-TIO security module before transporting to host SW (use to secure payload).

4. HCI transport and L2CAP routing

5. TApp decrypts and consumes the password; then clears security policy.



Secure Input Flow

1. User enters password field context - TA generates a symmetric key ( ) and programs security policy into Controller.

2. User types password

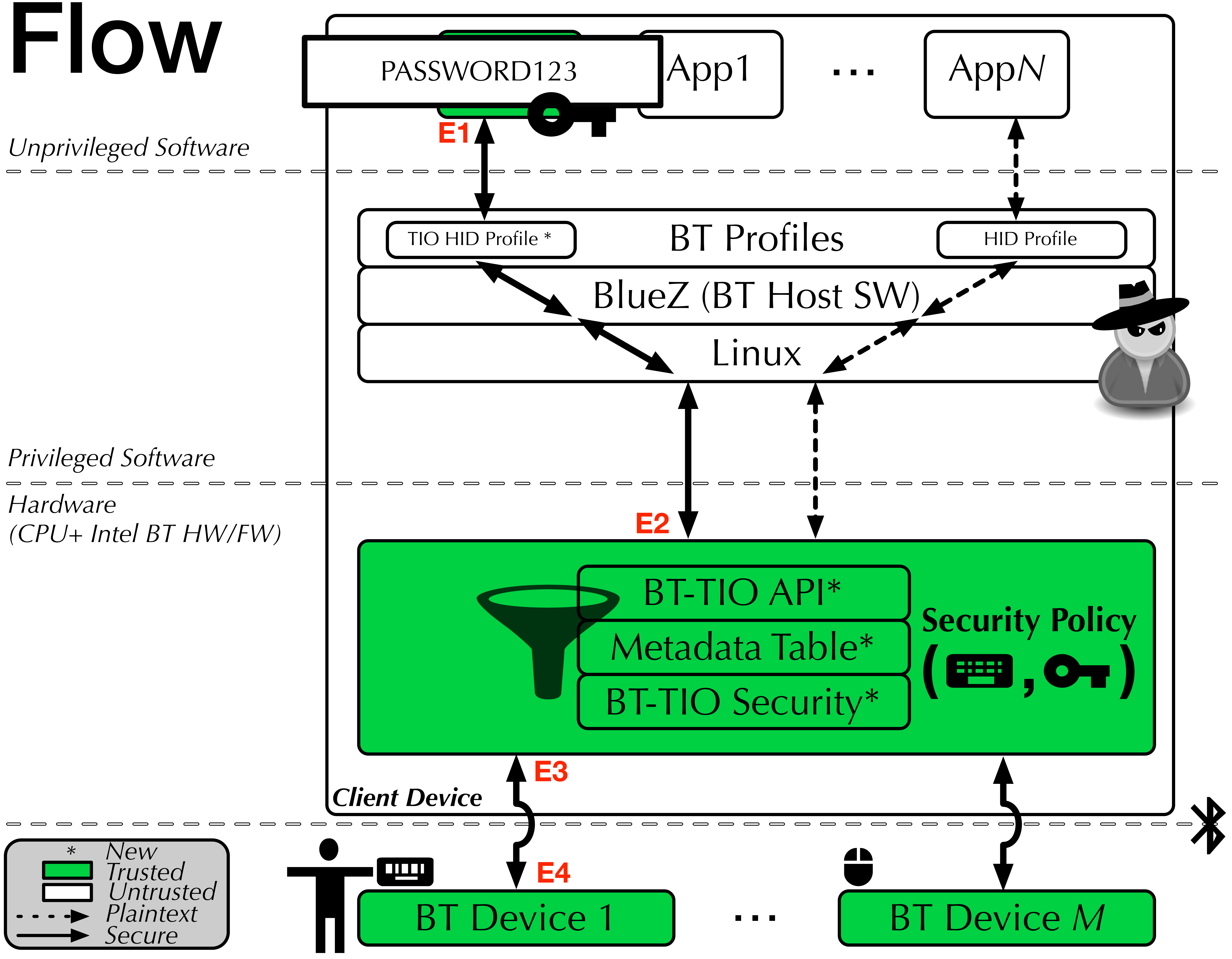
3. Controller filters packets matching *any* programmed security policy (,).

Matching packets are sent to BT-TIO security module before transporting to host SW (use to secure payload).

4. HCI transport and L2CAP routing

5. TApp decrypts and consumes the password; then clears security policy.

Trustworthy Input!



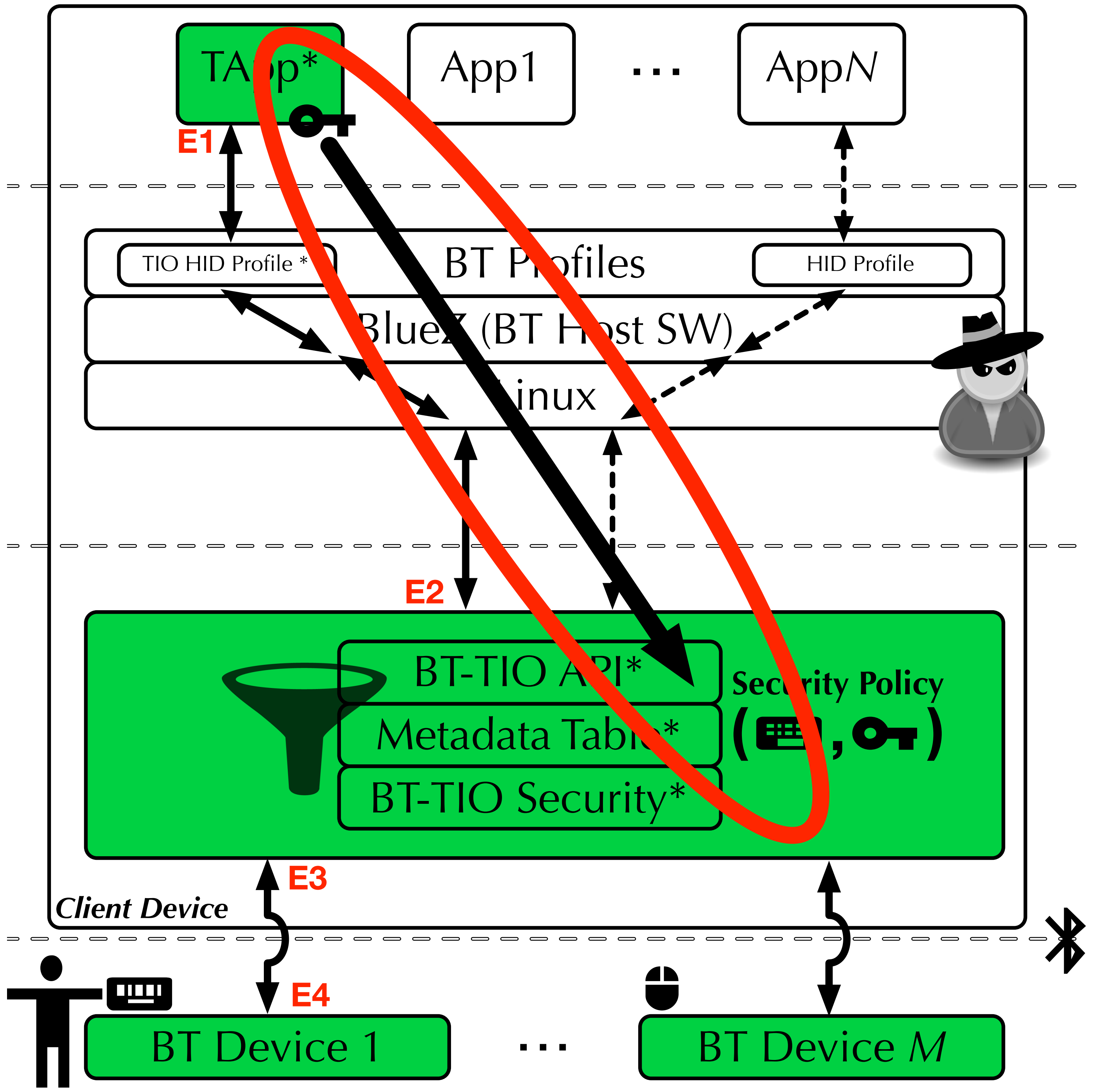
Conclusion

Take-aways

- Achieved E2E (app-to-device) security by extending the Bluetooth Controller firmware.
- Our extensions unobtrusively collect per-connection/per-channel metadata for Bluetooth Trusted I/O.
- Use metadata to secure Bluetooth I/O data between SGX app and Bluetooth Controller ***without…***
  - relying on untrusted host software.
  - requiring changes to SGX, Bluetooth device, or Bluetooth standard.
- PoC demonstrates how privileged keylogger cannot access user input data from connected Bluetooth device (keyboard).

Look in the paper* for details on…

- Dynamic key provisioning (Section 4.1.4) to establish secure channel for security policy key programming — re: PCIe & USB-C approach
- Future considerations
  - Extensions to other I/O paths (e.g., Wi-Fi, NFC)
  - Performance evaluation



**BASTION-SGX: Bluetooth and Architectural Support for Trusted I/O on SGX*

Thanks You!

Questions? Comments?

Please contact me at ***<u>traviswp@cs.dartmouth.edu</u>*** *if you'd like to talk more!*

BASTION-SGX: Bluetooth and Architectural Support for Trusted I/O on SGX

<u>Travis Peters</u>[1], Reshma Lal[2], Srikanth Varadarajan[2], Pradeep Pappachan[2], David Kotz[1]
Dartmouth[1], Intel[2]

Hardware and Architectural Support for Security and Privacy (HASP)
@ ISCA 2018

June 2nd, 2018
Los Angeles, CA, USA