



Certified Wireless USB Security

Preston Hunt

Technology Marketing Manager
Intel Corporation

*Content also provided by:
John Keys, Intel Corporation*

Introduction



- Cryptanalysis – is CCM really safe?
- Security Mechanics of the Connection

Why Do We Have Security?



Because of human ingenuity

CryptoAnalysis



- Common Initial Concerns:
 - Doesn't Required re-encryption for retransmission weaken the security?
 - I know enumeration (get dev desc, get cfg desc, etc.). Can't I use that to break the code?

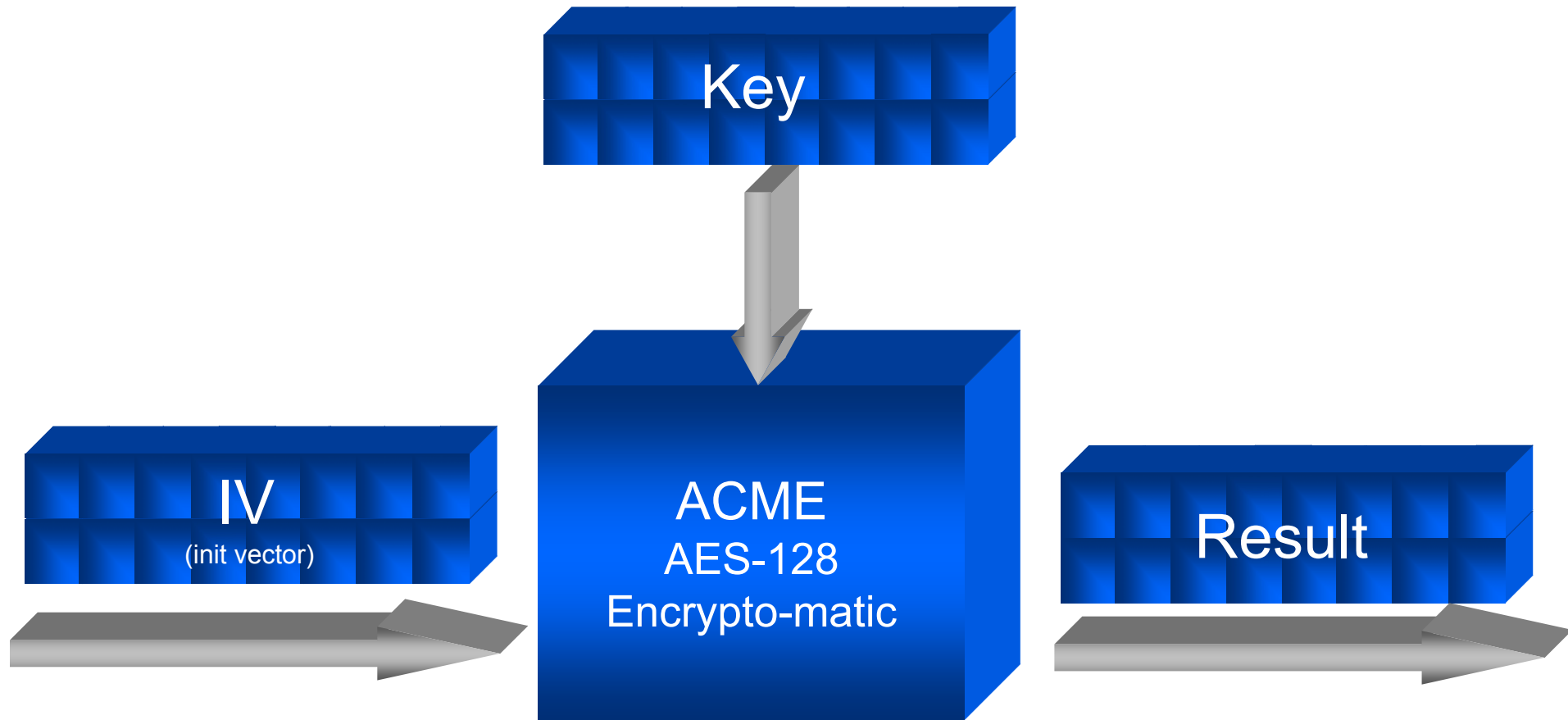
Because of CCM, the answer is NO!

One-Time Pads

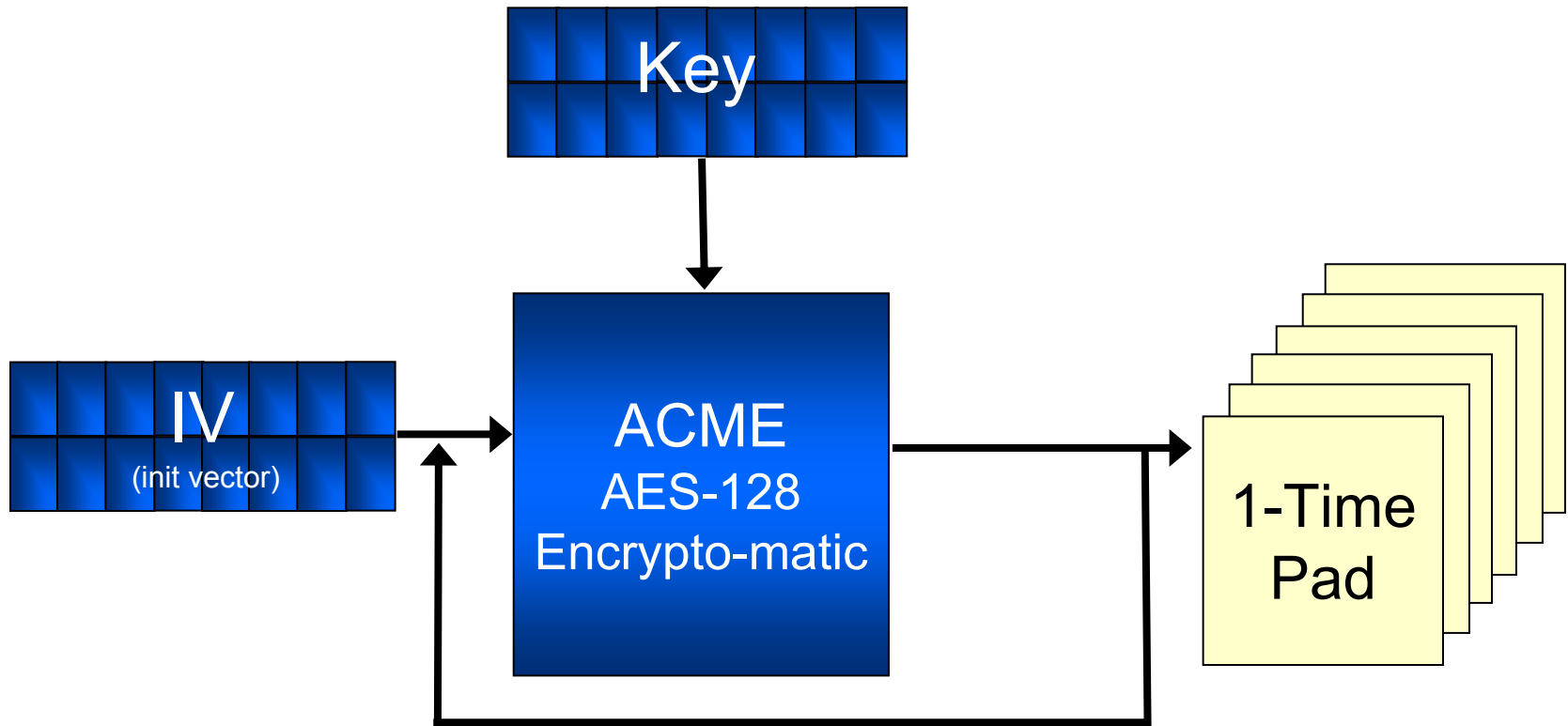


- The strongest cipher mechanism we have
 - not tied to message data
- Two parties have matching one-time pads
- Each character on each pad is used once to modify plain-text
- Each pad is a random-number series

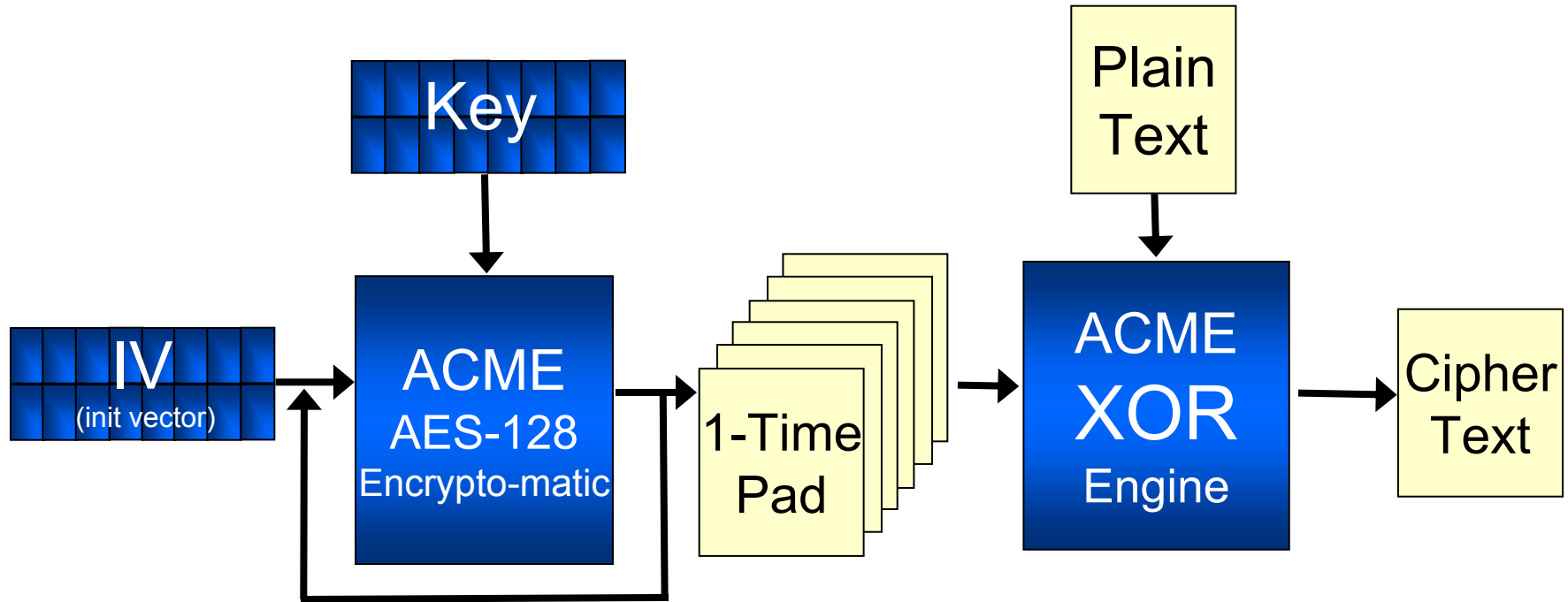
AES-128 Encryption



CCM Using AES-128

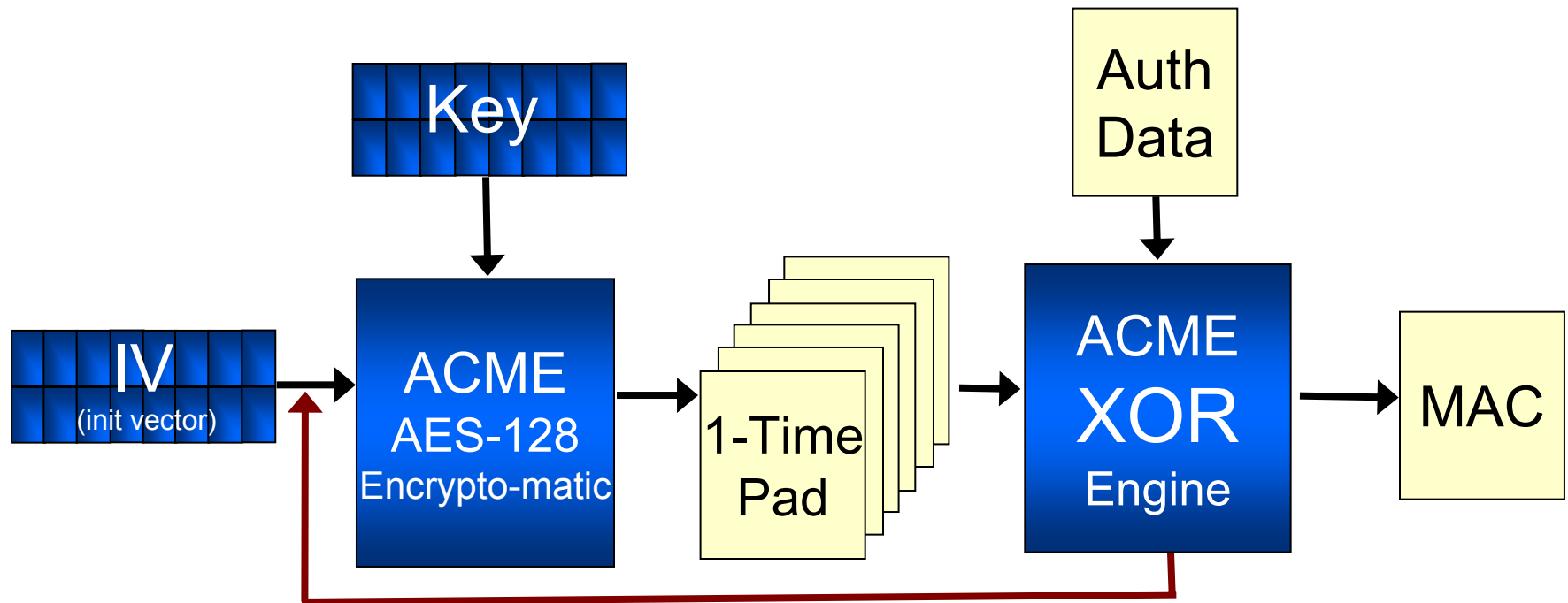


CCM Encryption/Decryption



Encryption/Decryption Identical: $PT \text{ XOR } n = CT$, $CT \text{ XOR } n = PT$

CCM Authentication



The Benefits



- I can give you both plain and encrypted text for a message
 - No help for solving old or new messages
- Just like the pad: having the last pad doesn't help guess the next pad
- Retransmission doesn't matter because we always use a fresh "pad" to encrypt

Connections



AKA



How the Host and Device come together, enable security, and live happily ever after or at least until one of the two reboots... or gets lost in the noise floor... or gets turned off... or gets carried away... or is eaten by the family dog...

Connections: Host Startup



- Beginning of time – Host Starts Up
 - Creates GTK and gives to HC
 - Enables HC security
 - Starts sending MMCs with HostInfo IE
- Why secure with no connections?
 - Eliminates special “first-time” case



Secure MMC

Secure MMC

Secure MMC

Connections: Device Startup



Secure MMC → 

(Host is sending secured, plain-text MMCs)

- Device scans for MMCs
 - Security not enabled
 - MMC received as “raw packet”, security info is present, but not processed. Device must know how to parse
 - If device finds HostInfo IE of interest, device makes unsecured Connect Request



CONNECT →



Connections: Back to Host



(Device is sending unsecured Connect Requests)

- Incoming unsecured DNs are accepted
- Connect ACK sent in subsequent secured MMC



- Host queries security info descriptors
- Host starts 4-way handshake



Connections: Handshake



- All Handshake setup data delivered inside secured MMCs (not encrypted)
- All device responses delivered inside unsecured data packets. Receipt by host is legal
- Handshake payload data protected out-of-band as per command definitions



Connections: HS Completion



- Host – Install derived PTK, enable security for device



- Device – Install derived PTK, but defer enabling of security. Wait for next operation, SetKey(GTK)



Connections: SetKey(GTK)

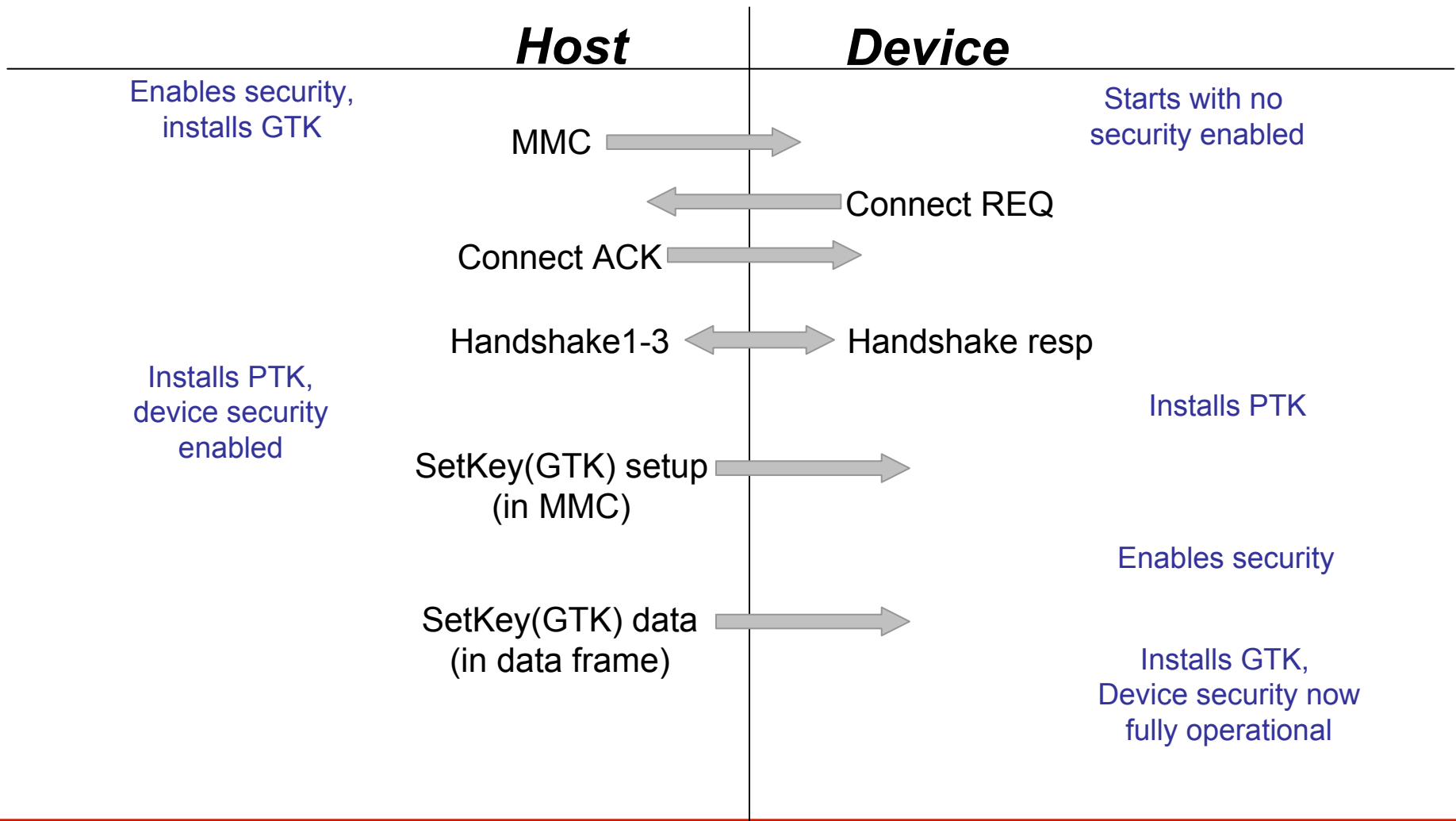


- Host – security for device fully enabled, sends SetKey(GTK) to device: MMC and secured data packet
- Device – receives MMC first:
 - Notes SetKey(GTK) setup,
 - Latches SFN from MMC
 - Enables security
- Device – receives GTK key data
 - Unencrypted with PTK (part of receive processing)
 - Installs GTK with SFN from MMC as replay counter

Device and Host now both fully operational



Connections: Message Chart



Thank you!



Questions?



Developers Conference 2006

Taipei, Taiwan