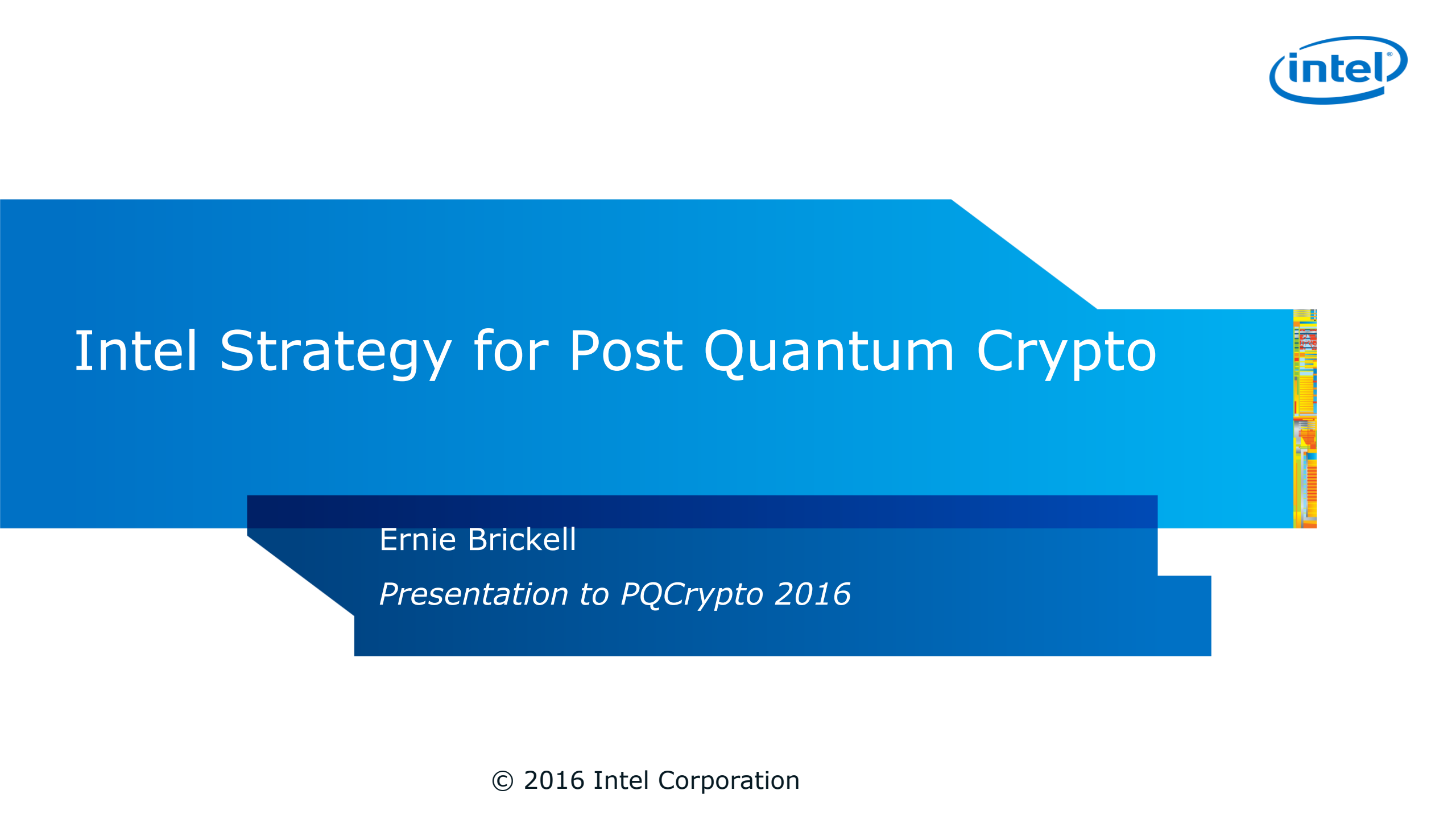


Intel Strategy for Post Quantum Crypto

Ernie Brickell

Presentation to PQCrypto 2016

Goal: Intel products must survive quantum computers

Problem:

- Quantum computers may come into existence sometime between 2030 and 2050.
- Intel products currently use crypto that would be broken by quantum computers

Goal: Intel products produced in the early 2020's would survive if quantum computing came into existence during their lifetime.

Disclaimer & Request

This presentation contains information about certain research efforts by Intel Labs. Currently, there is no product plan to implement these research efforts, which is subject to change without notice.

Motivation for presentation: Motivate priorities for standards and research for Post Quantum Crypto

Request: Feedback on the current proposals

Use of Crypto inside Intel products

Verified Boot, including FW load

FW update

Sealed Secrets

Memory Encryption

Attestation

Secure Key communication between factory and Secure Servers

DRM

Cryptographic Assumptions

- Must have a symmetric encryption algorithm and a hash function

Algorithm	Attack	Ops on Classical Computer	Ops on Quantum Computer
AES – 256 bit key	Cryptanalysis	2^{256}	2^{128}
SHA2 or SHA3 – 384 bit hash	Find Pre-image	2^{384}	2^{192}
	Find Collision	2^{192}	2^{128} with 2^{128} RAM

- Easy part of strategy:
  - Replace all use of symmetric crypto with AES-256
  - Replace all use of cryptographic hash with SHA2 or SHA3 -384
- Security Goal: Minimize need for any other crypto assumptions

For all Intel uses of crypto that cannot be modified in the field, are these minimal crypto assumptions sufficient?

Likely Implementation Assumption

Hash based signatures are great.

Digital Signatures where only cryptographic assumption is security of hash algorithm.

However, implementation risk for signer due to managing state

- MUST use each key at most once.
- MUST maintain state for which keys were used.

Stateless hash based signatures exist, but are very inefficient.

Implementation Assumption: For the uses of hash based signatures for non-modifiable usages, the signer will manage state robustly.

Standards

Risks to Intel:

- We implement something before standards evolve, and standards go a different direction.
- Standards are not optimal for our usages
- We implement a standard and the standard changes.

We would like world wide standards.

We would like standards soon.

Uncertainty:

- Will the stateful hash based signatures be approved by standards bodies due to the risk of implementation error?

Verified Boot

Verified boot is validating a signature on FW and or SW during the boot of a platform.

Algorithm requirements

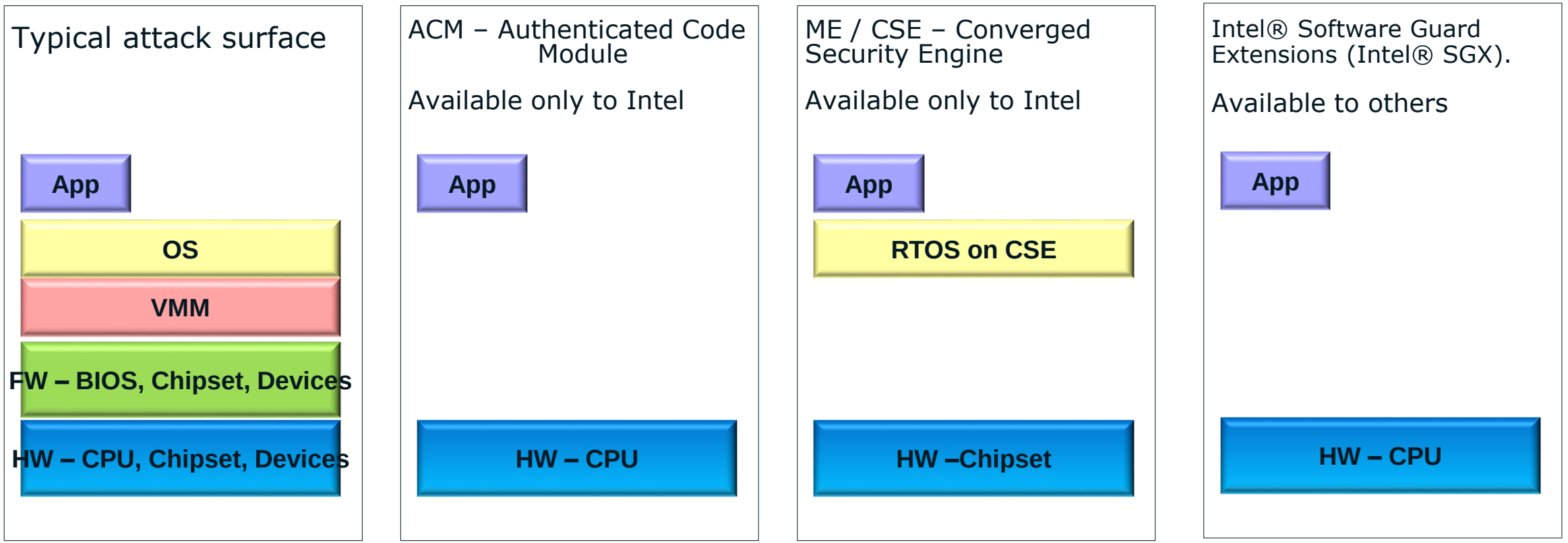
- Multiple components to verify during boot
  - Time to verify is critical
- Updates to boot components are rare
  - Time to sign is not critical
  - Small number of signatures in lifetime of platform
  - Verified boot algorithm is not field upgradable

Conclusion:

- Stateless Hash based signatures are not an option because of the lengthy verification time.
- Stateful hash based signatures with limits on number of signatures is a good option.
  - Concern: Implementation errors in maintaining state.
- Most efficient option is local conversion of a signature to a unique keyed MAC in a local isolated execution environment.

Isolated Execution Environment - Intel implementations

- Provides execution of code that is protected from some of the SW, FW, HW on the platform



Conversion of signature to local MAC

Properties:

- Have a "Boot Key," a unique symmetric key on processor, not known outside the processor.

At boot time:

- During boot, the processor uses the Boot Key to produce and check a MAC to verify code for execution.

Conversion to MAC:

- Have a "Conversion App" in an Isolated Execution Environment with the rights to access the Boot Key.
- When the platform gets an updated signed boot code segment, the platform launches the Conversion App, which
  - Attempts to verify the signature.
  - If verified, creates a MAC of this boot code using the Boot Key.

Conclusion:

- With this method, the signature verification time is much less relevant
- The stateless hash based signatures would thus be a possibility for many devices
- Could verify multiple signatures in the Conversion App.

Attestation keys

Keys on processor used to

- Prove that the processor is an Intel processor
- Used to sign properties of some software environment on the platform
  - Could be entire BIOS and VMM and OS.
  - Could be the code in an Isolated Execution Environment
  - Could be the hash of the software, or a public key that was used to verify a signature of the software.
- There could be privacy properties in the signature scheme.

Requirement for attestation:

- Intel must certify the public key used to verify attestation signatures.

Attestation public key certification requirement

Currently met using a secure communication between an Intel Secure Facility and each manufacturing site.

Different methods

- Keys generated at Intel Secure Facility and shipped to factory
- Shared secret generated at Intel Secure Facility and shipped to factory for later use to authenticate device for provisioning of attestation keys.

Key Exchange for secure communication from factory floor to Intel server

	Risks
Use multiple couriers	Human compromise of all couriers
PQC key exchange – optimized for security – perhaps McEliece or MDPC	Additional Crypto assumption
Quantum crypto key exchange	Potential implementation vulnerability, Key exchange between quantum crypto termination and factory floor

Perhaps cryptographic combo of first two of the above methods.

Alternate solution: Remove need for secure communication, and thus additional crypto assumption.

Removing need for secure communication to factories

Fundamental idea:

- Generate public private key pair on the part during manufacturing
- Sign the public key on the manufacturing floor and send to Intel secure facility

Cost reality:

- Time powering a individual part in manufacturing is really expensive

Easy theoretical solution today: Generate an RSA key pair on the part

- Not a practical solution:
  - Generating an RSA key pair takes too long
  - Random time per part on manufacturing floor is craziness

More practical solution: Generate a DSA or ECDSA key pair on the part

- Not PQ secure

PQ Secure solution: Generate a Hash Based Signature key pair on the part

- Limited in number of signatures

Removing need for secure communication to factories in Post Quantum Computing World

(Joint work with Rachid ElBansarkhani)

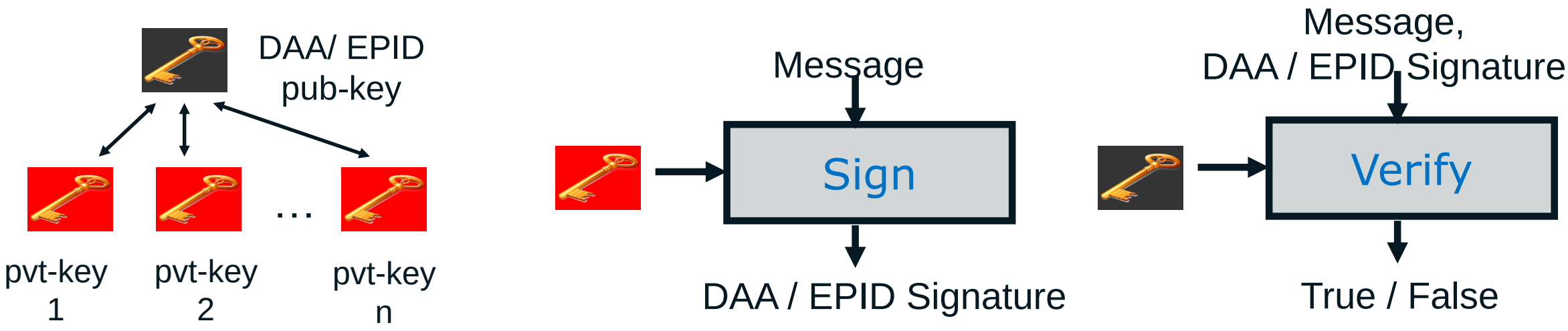
Method 1:

1. Generate a one time (or several time) hash based signature on manufacturing floor.
2. Export public key and sign on manufacturing floor
3. Send to Intel Secure Facility
4. Later, generate a multi-time public key
5. Sign the multi-time public key with the one time key for certification.
6. With a several time hash based signature in step 1, could repeat steps 4 and 5 several times.

Method 2:

1. Generate a random secret and one way function of secret on the manufacturing floor.
2. Export output and sign output on manufacturing floor
3. Send to Intel Secure Facility
4. Later, generate a multi-time public key
5. Authenticate the multi-time public key with the knowledge of the pre-image of the OWF
6. With an iterated OWF in step 1, i.e. C_0 = Rand, C_k = OWF(C_{k-1}), could repeat steps 4 and 5 several times.

Attestation and Privacy: Review of Digital Anonymous Attestation (DAA) (B, Camenisch, Chen) and Intel® Enhanced Privacy ID (Intel® EPID) (B, Li)



DAA / EPID is a digital signature scheme with privacy properties

- Join protocol allows a member to obtain a private key which the issuer does not know
- Anonymous: Even Issuer cannot open a signature and identify the signer

Privacy Features and Revocation DAA/EPID

During a sign process, the Device generates a pseudonym, a pair (b, b^f), that is cryptographically tied to the signature.

- f is a component of the private key of the signer

Modes of EPID signatures

- Random base: b is chosen at random by the device.
  - Signatures are **unlinkable**
- Name Base**:** – b is chosen by the verifier
  - Signatures using same name base are **linkable**, as pseudonym (b, b^f) is the same if b and f are both the same.
  - Signatures still **unlinkable** with different names.

Revocation

- Known Private key revocation (DAA and EPID)
- Verifier blacklisting using name base (DAA and EPID)
- Signature based revocation (EPID only)
  - When signing, signer proofs that his f was not the one used in a revoked pseudonym

Importance of EPID to Intel

We use EPID for providing attestation for our Isolated Execution Environments: CSE and SGX.

We have announced that we will be promoting EPID as an industry wide standard for providing security and privacy in IoT.

Attestation with Privacy – Post Quantum

Current EPID / DAA algorithms based on RSA, Discrete logs, or ECC

We have found no PQC method to replace EPID or DAA.

Options:

- Digital signature without privacy – Lattices or hash based
- Two options for privacy
  - Trusted Third Party
  - Group signatures where issuer can open any signature
    - Lattice based group signatures are not efficient with large groups
    - Hash based group signature scheme is efficient, but stateful.
      - New result from Rachid ElBansarkhani and Rafael Misoczki

Attestation keys - Summary

Create few (k) time authentication key for authenticating to Intel

- Timing – During Manufacturing
- Algorithm – Not field updateable
- Key – Not field updateable

Authenticate

Create many time signature key for authenticating to Intel

- Timing – After Manufacturing
- Algorithm – Not field updateable
- Key – Field updateable k times

Authenticate

Create anonymous signature key for general attestation

- Timing – After Manufacturing
- Algorithm – Field updateable
- Key – Field updateable many times

For all Intel uses of crypto that cannot be modified in the field, are these minimal crypto assumptions sufficient?

Use of Crypto inside Intel products

- Verified Boot, including FW load
  - Hash based signatures and MAC
- FW update
  - Hash based signatures
- Sealed Secrets
  - AES
- Memory Encryption
  - AES
- Attestation
  - Hash based signatures and Hashes for non updateable portion
  - Looking for better solution for updateable portion
- Secure Key communication between factory and Secure Servers
  - Removed this requirement
- DRM
  - Uses attestation, then updateable algorithms provided by DRM industry

Yes!

Additional Intel requirements

Want standards for crypto for wireless and internet protocols that minimize resources, especially power and memory.

Request for standards and research

Would like to see standards produced that are appropriate for:

- Stateful hash based signatures used for verified boot
- Stateless hash based signatures used for conversion to local MAC for verified boot
- Key exchange where long term security is imperative and efficiency doesn't matter to use between factory and secure facility
- Maintaining security of internet using algorithms efficient in hardware and small devices

Would like research to solve the following:

- PQC alternatives to EPID
- Reduce risk of vulnerabilities in implementation of PQC algorithms
- Convincing evidence that we can trust PQC algorithms other than hash and symmetric, especially RLWE

Intel is sponsoring PQC research at TU Darmstadt, U. Sao Paulo, and previously at National Taiwan University

Questions? Comments?