Open IoT Blockchain Trust Engine Architecture

D. Jeff Dionne, Jennifer McCann, Kenneth Tang, Core Semiconductor & Jeff Garzik, Bloq Inc.

Abstract—**Building networks of IoT devices which can provide and access services efficiently and securely is a difficult problem given today's hostile cyber network environment and the constrained computing power of most IoT devices. Blockchains provide a solution which provide a strong defense against network level attacks but require large compute resources and trusted storage for example for private key material. Ironically, some of the world's top cybersecurity experts only trust printing out their private keys on paper and storing them in a safe or bank safety deposit box to secure them. Current trusted platform modules that provide trusted storage and secure boot amongst other functions are closed designs which are often fallible to security holes and do not provide continuous runtime protection to devices. Herein, we describe an open architecture for building heterogeneous networks of devices, where the hardware itself provides a Trust Engine to facilitate security for all parties in a trustless, multivendor environment.**

Index Terms—**ASIC, Blockchain, cybersecurity, IP Cores, IoT, Open Architecture, Open source hardware, Trusted Computing Engine, Trusted Platform Module**

I. Introduction

THE so-called Internet of Things (IoT) is projected to touch almost all aspects of life and society, with the number of IoT devices connected reaching 100's of millions in the next few years. The management of these devices is fraught with security risks for many reasons, chief drivers of which are the underlying interests of device manufacturers (sales, profit, and market monopolization) being misaligned with that of the consumers of such devices. If the platform upon which the device is built provides certain inherent security guarantees, manufacturers can focus more on development of their product functionality while users can be guaranteed robust and comprehensive security provided inherently by the platform and the network.

Such an system architecture requires careful design of security partitioning, with the hardware providing some basic security guarantees and the network providing a robust set of defenses against foreseeable and unforeseeable but inevitable attacks. By breaking the problem space down into zones of trust, and isolating breaches of security to the zone where it occurs, a resilient network of devices with unique, differing product functionality can be constructed. In order to prove that the system is secure without trusting the manufacturers, the design also needs to be open to scrutiny, and therefore open source. Fully open source hardware (including as far as the processor microarchitecture level) is also necessary to arm developers with the required information to develop code that is resistant to sophisticated hardware attacks such as timing analysis attacks [1].

This paper was released in initial version at <Conference> 2018.

Reviewers are…

We propose to combine open source, trusted hardware compute engines with cryptographic accelerators to create a low cost, inherently secure, platform for IoT devices. Since the hardware and the software are open source, malicious designs not conforming to the requirements described later in this paper, as well as conforming designs under the control of malicious actors (individual or colluding) are of course possible. A blockchain network architecture can provide network level security for this set of threats. Therefore we propose the IoT devices would be interconnected in a blockchain network to create an overall decentralized IoT system, with no central point of control or single point of potential failure, resilient to virtually all known types of attacks. This should naturally include the support of smart contract based, decentralized applications running on the blockchain especially to provide common network services which are resilient to network level attacks.

Devices based on this platform would typically be implemented as an ASIC chipset to reduce per unit cost and provide both traditional IoT functionality, including interaction with the real world via built-in sensor and actuator interfaces, signal processing, network connectivity, and participation as nodes in a distributed and trustless blockchain network to provide registration, management, upgrade, and payment services. The network itself provides a multi-vendor, scalable X-as-a-Service platform to these devices.

Exploring the security implications leads to a set of required properties for the hardware platform itself. We then describe an open hardware trusted compute engine architecture to address these requirements along with the need to verify correctness of the implementation. A complete system architecture is developed, including the OS and blockchain middleware.

II. Previous Work

Much of the work in the IoT field has been on thin and low power RF communication system architectures. Meanwhile, it is common perception that blockchain systems require enormous computing, and therefore energy resources. There have been a few attempts to produce consensus algorithms that allow blockchains to scale up to the size of a practical IoT network with some of these running on practical IoT node hardware.

A. Blockchain Architectures

i. Polkadot

The focus of Polkadot is to scale networks of blockchains by providing a general purpose platform to move transactions between multiple, possibly lightweight parallel chains (parachains) and public or private blockchains. Polkadot provides a relay chain to achieve this. While the primary focus of Polkadot may not be IoT nodes, the individual parachains use a lightweight consensus mechanism to avoid the necessity of running compute intensive Proofs of Work on the nodes. Polkadot

[2] parachains build on simple Byzantine Fault Tolerant [3] consensus algorithms, perhaps used first for blockchain applications by Tendermint [4].

Polkadot remains a work in progress, but the architecture points the way for immediate implementation of systems more limited in scope, and for application specific purposes. Individual parachains are clearly a good match for collections of IoT devices operating on private and/or application specific chains. As noted by the authors of Polkadot, many of the problems proposed to be solved in the Polkadot network, specifically around selection of validators, can be avoided in more limited Proof of Authority blockchain systems.

ii. Hdac

The Hdac [5] design makes certain assumptions about the limitations on computational resources and architecture of IoT nodes, and builds out the network from there. While Moore's law has slowed, it has been noted many times and is still valid that software evolves far more slowly than hardware does. Therefore Hdac's set of assumptions seems of questionable validity, or at least may not be valid for very long. Hdac proposes to use a derivative of the Bitcoin [6] engine, with one stated reason that C++ seems a better match for commodity IoT hardware and related engineering practices.

The consensus mechanism for the Hdac network, called ePoW, locks out a successful miner from mining activity for a calculated but variable amount of time. This limits the energy consumption for each node, and as a result, for the network as a whole. Without careful analysis, it remains an unproven assertion that the limitation in complexity and therefore hashing power implied will not compromise the security of the design when attacked by a more capable adversary. Indeed, it is a principle of the Nakamoto PoW consensus algorithm [7] that the complexity of the problem must be proportional to the total network hashing power in order to maintain the level of difficulty of mounting a successful attack. This problem is solved in other bitcoin derived blockchain implementations for similar use cases, e.g. QTUM [8], by replacing bitcoin PoW with a Proof of Stake consensus algorithm.

Hdac proposes to use relatively small private blockchains to which either IoT nodes or gateways would be attached. These private chains would use (presumably trusted) relay nodes to provide transaction forwarding to public or other private chains. Overall, the security of this system appears dubious [9].

iii. IOTA

IOTA takes a different approach to solve the scalability problem of blockchains for IoT by using what it calls a "Tangle" instead of a blockchain structure. A Tangle is a directed acyclic graph (DAG), and in this case where each node points to exactly 2 predecessor nodes. The DAG nodes represent transactions and the edges represent validations, thus each transaction validates 2 previous transactions before it is published and recorded. So theoretically, this parallelizes the transaction processing for infinite scalability. However IOTA is unable to support smart contracts, especially where order transaction order finality is required, and has centralization risk in the coordinator node. The coordinator node is also closed source [10]. Thus IOTA seems to have some serious hurdles to overcome to become an alternative to blockchain for implementing a trustless IoT network.

B. Hardware Engines

Direct approaches to building blockchain enabling hardware architectures appears to be new, or at least is rare in the literature. While the success of ASIC mining hardware shows that meaningful acceleration of PoW algorithms is straightforward, the design of limited node hardware architectures and mining use cases have completely different and largely non-overlapping requirements [4]. Trusted Platform Modules (TPM) are a more appropriate paradigm for IoT applications, but closed source designs dominate in actual implementations. Although counterintuitive, long-lived, robust security is rarely achieved through obscurity, and correcting this is one of the primary goals of this work. Case in point is the the vulnerability discovered in widely used Infineon TPM chips affecting a large number of devices manufactured between approximately 2012-2017 [11]. Furthermore, TPMs are passive devices and provide no active runtime execution security [12]. Although TPMs exist in many devices, especially PCs, they are often not used by the OS or system software, and cannot control or enforce the boot process [12].

Interesting hardware engines or accelerators for related purposes either exist with open implementations or with sufficient documentation in the literature to provide a basis for architecture design. Maene et al [13] provide a good starting point into the literature, and so we will only review briefly some applicable hardware approaches here.

i. Sancus

The Sancus 2.0 [14] system provides both hardware and middleware that takes a minimalist approach to providing trusted compute services in a limited sensor node environment. Sancus predates blockchain implementations on the IoT. Notably, the Sancus architecture does not provide acceleration for asymmetric cryptography in its base platform, and therefore depends upon at least some level of trust in other participants in the network environment, as opposed to the trustless principles normally assumed in blockchain system designs.

In spite of the limitations imposed by the assumptions around the size of system Sancus directly targets, it none the less proposes a useful if minimal set of architectural features which can be added to a processor architecture to provide guarantees for isolation of software modules, and attestation of software module integrity. The architecture of Sancus is a mixture of middleware primitives, and hardware elements, both of which are applicable to the construction of a partitioned security engine for blockchain IoT.

ii. Netronome Systems offload engine

An obviously highly successful innovation in blockchain networks is the smart contract [15] concept, by far the most successful of which is the Ethereum Virtual Machine. Virtual Machines are required to ensure portability of smart contract code to all nodes regardless of hardware. While VMs are nothing new, providing proofs of correctness and security guarantees for arbitrary VM code is an active area of research. One such VM which attempts to provide certain well defined guarantees is the

Linux kernel extended Berkeley Packet Filter [16], [17]. The eBPF subsystem uses a validator to analyze all possible execution paths through eBPF binary code to provide such guarantees. eBPF then may be translated using a JIT compiler to native machine code.

In order to not slow down time required for consensus, offloading VM code execution to hardware is desirable. Netronome provides a direct binary translation of eBPF code to run on its hardware offload engines [18]. The Linux Kernel provides a framework for JIT translation to architectures which Linux runs on, but translation of eBPF to a dedicated hardware engine appears new, at least when combined with the VM code security analysis. The utility of being able to move execution of eBPF code to either the host CPU or one of a number of parallel hardware acceleration engines will be immediately obvious. eBPF VM code can be emitted by the LLVM compiler suite, which opens the possibility of writing or porting code in a large selection of mature and rich programming languages. While the purpose of the Netronome offload engine is acceleration of packet processing, if the hardware offload engine provides security guarantees for sensitive objects such as key material, this in combination with the analysis of VM code for execution behavior is beneficial for safe and deterministic execution of e.g. smart contract code.

iii. ARM TrustZone and Intel TXT/SGX

Although they not open implementations, it is important to mention these platforms because of their ubiquity and widespread, de facto acceptance.

ARM TrustZone was introduced in 2003 and provides hardware support for two execution environments, one trusted and one non-trusted on the same core and constructs a security perimeter around each.

Intel Trusted Execution Technology (TXT) was introduced first in 2007 and supports hardware root of trust (with the secure element provided by a 3rd party manufacturer), boot time integrity measurement, and attestation. Intel Software Guard Extensions (SGX) was introduced in 2015 and are a set of hardware-assisted CPU instructions that are intended to provide isolation (code & data) between processes.

Yet despite the undoubtedly significant engineering efforts put into these designs, in 2017 two security vulnerabilities, codenamed Meltdown and Spectre affecting virtually every ARM, Intel and IBM POWER processor since 1995 were discovered [19],[20]. Qualcomm and AMD have also confirmed that some of their processors are susceptible but the number of chip designs are significantly fewer than ARM and Intel. This reinforces the position of many security specialists that all closed source designs even by leading manufacturers are not infallible. Security by obscurity only works so long before it is broken by someone. Both the Meltdown and Spectre vulnerabilities are to side channel attacks in which a process is able to snoop data from another process, that is, process isolation is violated.

III. Proposed System Architecture

One imagines a scenario where small chains of computing power limited, internet connected IoT devices are organized into geographically or logically related groups. Individual devices can have separate owners, and may be multivendor. These IoT devices attach to a local blockchain, and there may be one of more such blockchains. This leads naturally to the following assumptions around the participating IoT devices themselves:

- The users of such devices need not have any knowledge of the hardware, or network architecture.
- Devices need not inherently trust either other nodes on the blockchain, including for example, any service provider(s) or relays. That is, the blockchain is a trustless environment.
- The user and vendor expect the device itself to provide the basic security guarantees, defending itself against subversion from outside attack.
- Transactions (of any kind, e.g. payments, configuration, upgrade, provision of service) will be protected by blockchain consensus from known common attacks. As such, the blockchain will provide immutable data to devices and service providers.
- Some number of devices may be subverted by an attacker or malicious user.
- The middleware on the device shall be able to verify the state of e.g. cryptographic key material and transactions locally, by checking the integrity of the software modules performing these actions on the private blockchain.
- Layers of security, with hardware support for different privileges, are required. Hardware isolation and attestation of system software, and secure execution of smart contract VM code shall be guaranteed by the hardware itself to contain the effects of inevitable software vulnerabilities being exploited.

Devices for which these assumptions are true will provide the owner of such devices with assurances that the device itself does not perform malicious actions on the users behalf (say, transfer all your Bitcoin to me, although I am happy to send you an in address).

A blockchain network of nodes composed of such IoT devices in turn has certain properties:

- Limited hashing power, necessitating a consensus algorithm other than PoW.
- Necessarily separate from public blockchains. That is, these devices are not directly nodes on the ABC IoT coin network.
- Support for a flexible and verifiable smart contract VM, to allow the creation and deployment of distributed applications between users of devices built on the platform.
- The ability to perform transactions on a public blockchain, for any purpose such as service registration, payment transactions, etc. on behalf of the private blockchain itself, and its users is provided by a hierarchical set of parachains and vendor chains.

i. Scope

The architecture proposed here envisions a minimum specification necessary to meet the stated assumptions.

- A CPU architecture with security primitives and memory protection units necessary to provide isolation, measurement and attestation.
- An SoC platform that provides for cryptographic operations. This engine shall allow for both trusted and untrusted use of the underlying cryptographic

acceleration primitives, without compromise of security in either mode.

- A blockchain middleware built around a core, lightweight but flexible consensus architecture. Selection of e.g. validator nodes, etc is left to the individual private blockchains implemented on top of the middleware (with an example provided).
- Unique hardware device IDs, issued by smart contract code running on a public blockchain, not under the control of any single entity. The hardware device ID is used to form the basis of the hardware root of trust.
- Relay blockchains built by vendors or service providers connect devices to services. These relay chains can be built on top of Ethereum, QTUM tokens for example, or they can be completely private blockchain networks.
- A public relay chain is provided, both as a reference implementation and production environment, running on the Ethereum network. It is implemented as an ERC20 or QRC20 contract [21] that acts as the gateway of information between the vendor/ application chains and the public blockchain.

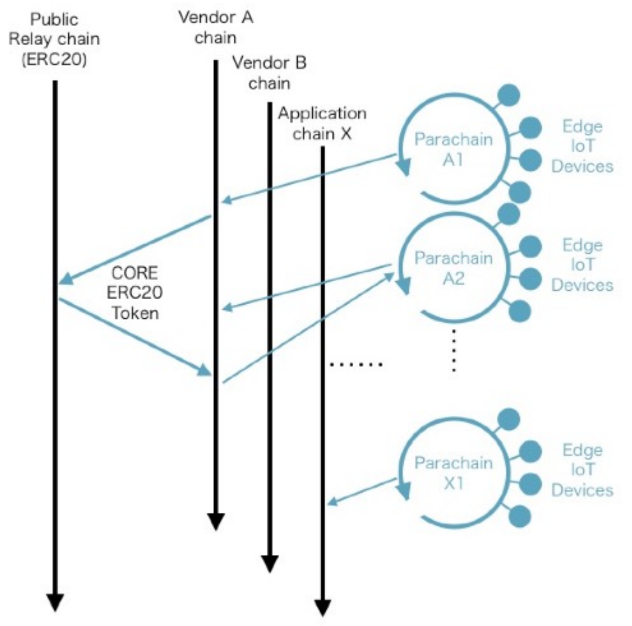

Figure 1. CORE Blockchain Architecture

An overview of the CORE ecosystem blockchain architecture is shown in figure 1. The public relay chain, using the CORE ERC20 (or QRC20) token, supports the open and scalable attestation of device and software for e.g. provisioning, registration services, firmware upgrade and app store, as well as traditional cryptocurrency functionality such as payment transactions. Note that essential functions such as provisioning, registration and key management are implemented as decentralized applications in order to eliminate centralization risk, such as that identified in the Sancus design. CORE serves as a marketplace currency for X-as-a-Service to connected devices. Device sensor data and other off-chain data is not dictated to be stored in a specific location. Most IoT offerings use cloud servers, but this is only one of several possibilities in the CORE architecture. To be cyberattack resistant, decentralized storage on the vendor/application chains, especially using DHTs is a good option.

ii. Consensus In Blockchains of Small Devices

Scalability, specifically transaction throughput, is perhaps the main area of research around the applicability of blockchain based general purpose cryptocurrency systems. The consensus algorithm problem space of practical and secure blockchain systems for networks of connected devices will necessarily have different constraints from general purpose cryptocurrencies, and therefore different solutions are necessary. Since the computational complexity of hashing problems addressable by a network of small embedded devices (even if hardware hashing accelerators are used) will always be smaller than a well resourced attacker would be able to amass, simple PoW consensus cannot provide security from such attacks, and we must look for alternative algorithms. It is perhaps useful to look to expected and typical use cases here to determine what properties a successful algorithm might have. At one end of the spectrum, when each node (in this case, each device) is basically an equal stakeholder in the network, PoS reduces to round robin of available nodes, many of which will be network-connected much of the time. Such use cases have relatively benign threat models. At the other end, mobile devices of differing capability may consume e.g. media from different content providers, a use case where a large percentage of the device nodes on the blockchain will be transient, and perhaps actively attempt to subvert the network in order to achieve theft of service or similar malicious activities.

A class of consensus algorithms using the properties of Byzantine Fault Tolerant systems seems a promising approach. Kwon [22] shows that such a class of algorithms with well defined properties can be built which all rely upon the construction of the BFT problem, providing security guarantees, if an appropriate selection of nodes designated as validators can be made and made securely. In the typical case of industrial IoT devices, the set of validators in e.g. a factory setting might simply be all participating nodes, or a fixed set of nodes selected by the process engineer, implementing effectively a static Proof of Authority consensus algorithm. In another example case of a power grid network, the selection of validators might more appropriately be dynamic. Validator nodes might be chosen by a deterministic algorithm taking recent historical power flows on the electricity grid, recorded in the blockchain, as inputs. In all cases, the selection of validator set must be deterministic if consensus is to be reproducible, and in the case of blockchains where Byzantine behavior (as defined by the BFT problem) is expected from possibly malicious actors, difficult to predict in advance, influence or subvert.

It is clear that any system which provides either no solution to the selection of validators, or limits the choice of validator selection algorithms based on a fixed set of design assumptions will fail to address many conceivable use cases. Instead, we propose to use the relay chain to provide generic authority and/or staking information to the parachains. The relay chain attests to the right of individual parachain nodes to participate on the chain and the validator set. The devices themselves provide attestation to their unsubverted state in hardware. Consensus regarding the selection of validator set at any given block height can be implemented in node software, and is therefore adaptable to various use cases, supported by the hardware, attestation of software state and immutable data provided by the relay chain as an oracle.

iii. Threat Model

The threats for such a system can be simplified into the sum of threats common to all BFT consensus algorithms (chiefly that more than one-third of nodes are Byzantine), plus the threats to the Trust Engine underlying the devices participating as validators to causing them to be Byzantine and threats to the selection of the validators. Table 1 gives a summary of the threats and the primary defense mechanism(s) of the design.

Threat Type	**In Scope**	**Primary Defense Mechanism(s)**
Man-in-the-middle	Yes	PKI signatures & encryption (secure I/O)
Device spoofing	Yes	Hardware Root of Trust, non-exportable identity keys, PKI authentication
Code reuse attack	Yes	Controlled entry & exit
Privilege escalation	Yes	Hardware enforced protection
Chip reverse engineering	Partial	Can be addressed post-synthesis in layout
Physical attack (e.g. bus probing/injection, reset, memory, JTAG)	Yes	SoC integrated TCE, Off chip memory encryption, measurement and attestation, secure JTAG
Side channel attack (non-physical)	Yes	Hardware enforced isolation
Physical side channel attack (e.g. timing, power analysis)	No	Careful code implementation can mitigate
Fault injection (e.g. Rowhammer)	Yes	Run-time integrity verification
Fault injection (e.g. power supply, clock)	No	Can be addressed by system level design
Sybil attack	Yes	Signed transactions using hardware Root of Trust derived key
DoS/DDoS attack	Yes	Decentralized, P2P blockchain architecture
Collusion attack, double spending attack	Yes	BFT Consensus
Malicious smart contracts	Yes	VM accelerator with isolation
Attack on CA (e.g. for public key certificates)	Yes	Decentralized CA application

Table 1. Threat Model

iv. Trust Engine Model

The usual requirements apply:

- Validation of integrity at load time
- Attestation of continued integrity at run time
- Isolation of software modules
- Secure storage of private data e.g. key material
- Enforced single points of entry and exit
- A secure mechanism to switch contexts
- Cryptographic primitives and services

A hardware architecture which provides these services is shown in Figure 2. A general purpose CPU supported by instruction set additions is connected to hardware accelerators. The SHA256 accelerator computes the module block hash upon loading to verify integrity before it is allowed to execute. The module block hash is also computed immediately after exit, and stored to be used to verify integrity next time, thereby providing run time integrity. The Enforcement Engine enforces isolation of software modules, and single points of code entry and exit.

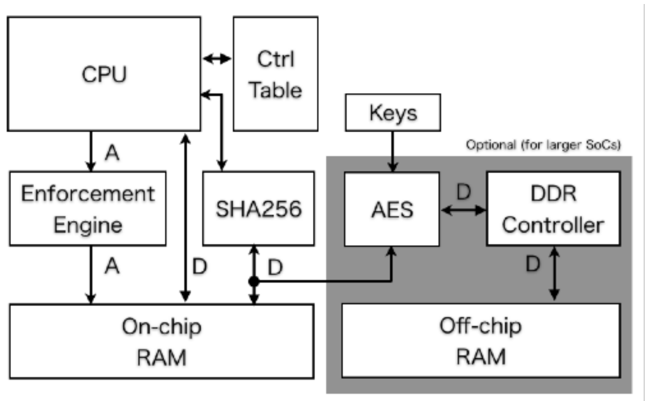

Figure 2. Trusted Execution Engine Hardware

Caches also need to be cleared between context switches. For design scalability, modules can be AES encrypted when stored in external, off-chip RAM. For high performance applications, run-time attestation may be forgone or performed less often at lower security levels. Remote attestation is facilitated by the blockchain, to avoid centralization of trust in any single entity. This solves the purported Achilles heel in Intel's SGX implementation [13],[23]. The associated reference implementation software stack is shown in figure 3.

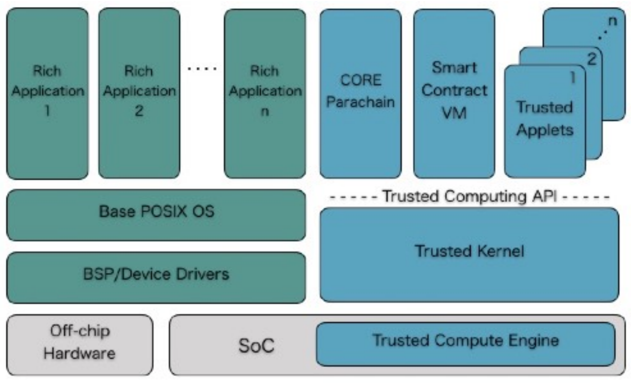

Figure 3. Software Stack

Note that the platform can support execution of closed source trusted applications, with encrypted binaries which are decrypted upon loading, should a vendor desire to do so.

v. Relay Chain Token Economics

A key aspect of blockchain systems is the use of a cryptocurrency token, not only as a way to exchange value, but to incentivize participants, including participants with ill intentions, to act in ways that are beneficial to the whole. The CORE token is a utility token, burned to authenticate new devices onto the blockchain.

IV. Conclusions

We have described a trust engine architecture, that provides device security rooted in hardware and has the potential to serve as the core of all IoT devices. We have further described a scalable, blockchain network which can provide very robust network level security. We assert that the 3 principles of this architecture, namely:

1. A trusted compute engine with basic security guaranteed inherently by hardware
2. A decentralized blockchain network with no potential single point of failure
3. Independently verifiable, open source hardware and software

are complementary and necessary. The absence of any one of these characteristics compromises the security and reliability of the whole, exposing an Achilles heel exploitable by malicious actors. Ongoing future research will focus on detailed implementation of unique aspects of this architecture, alignment with existing and emerging IoT standards, toolchains to automate hardware and software development based on the platform, as well as application to key IoT applications.

V. Glossary

AES	Advanced Encryption Standard
ASIC	Application Specific Integrated Circuit
BFT	Byzantine Fault Tolerant
CA	Certificate Authority
DAG	Directed Acyclic Graph
DHT	Distributed Hash Table
eBPF	extended Berkley Packet Filter
ERC20	Ethereum Request for Comments #20: The technical standard used for smart contracts on the Ethereum blockchain for implementing tokens.
Hdac	Hyundai Digital Asset Currency
IoT	Internet of Things
JIT	Just in Time
LLVM	Low Level Virtual Machine
parachain	parallelizable chain
PKI	Public Key Infrastructure
PoS	Proof of Stake
PoW	Proof of Work
QTUM	UTXO-based smart contract system with a PoS consensus model
relay chain	A blockchain that can validate and read events and/or state in other blockchains
SHA256	Secure Hash Algorithm - 256 bit
SoC	System on a Chip
TCE	Trusted Compute Engine
TPM	Trusted Platform Module
VM	Virtual Machine

VI. References

1. G. Heiser, "For Safety's Sake: We Need a New Hardware-Software Contract!," IEEE Design & Test, vol. 35, no. 2, pp. 27-30, April 2018
2. G. Wood, "Polkadot Whitepaper," *parity.io*, 2016
3. M. Castro, and B Liskov, "Practical Byzantine Fault Tolerance," *Proceedings of the Third Symposium on Operating System Design and Implementation,* 1999
4. T. Graham, "Tendermint: Byzantine Fault Tolerance in the Age of Blockchains," MAsc, School of Eng., Univ. of Guelph, 2016
5. Hdac Technology AG. "Hdac: Transaction Innovation - IoT Contract & M2M Transaction Platform based on Blockchain", GitHub, 2017
6. S. Nakamoto, H. Finny, J. Garzik et. al., "Bitcoin Core", Sources, bitcoin.org/en/bitcoin-core, 2011-2018
7. S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," bitcoin.org/bitcoin.pdf
8. P. Dai, N. Mahi, J. Earls, and A. Norta, "Smart-Contract Value-Transfer Protocols on a Distributed Mobile Application Platform," Qtum Foundation, Singapore, 2016
9. J. Young, South Korean Conglomerate Hyundai's Cryptocurrency Mining Pool Hacked, Withdrawals Disallowed, CCN, May 26, 2018
10. B. Breier, Technical Analysis of the Tangle in the IOTA-Environment, Technical University of Munich, Nov. 6, 2017
11. National Cyber Security Centre, ROCA: Infineon TPM and Secure Element RSA Vulnerability Guidance, Oct. 20, 2017
12. A. Segall, Trusted Platform Modules, The Institution of Engineering and Technology, 2016
13. P. Maene, J. Götzfried, and R. de Clercq, "Hardware-Based Trusted Computing Architectures for Isolation and Attestation", *IEEE Transactions on Computers,* vol. 67, issue 3, 2018
14. J. Noorman, J. Van Bulck, J. T. Mühlberg, et al., "Sancus 2.0: A Low-Cost Security Architecture for IoT devices," *ACM Transactions on Privacy and Security*, 2017
15. N. Szabo, "Smart Contracts: Formalizing and Securing Relationships on Public Networks," *First Monday*, vol. 2, no. 9, 1997
16. S. McCanne, and V. Jacobson, "The BSD Packet Filter: A New Architecture for User-level Packet Capture," *Proceedings, USENIX 93, Winter,* 1993
17. A. Starovoitov et al., Linux Socket Filtering aka Berkeley Packet Filter (BPF), *Linux Kernel Documentation*.
18. J. Kicinski, N. Viljoen, "*eBPF Hardware Offload to SmartNICs: cls_bpf and XDP.,*" Netronome Systems
19. P. Kocher, J. Horn, A. Fogh et al., *Spectre Attacks: Exploiting Speculative Execution*, 40th IEEE Symposium on Security and Privacy, 2019
20. M. Lipp, M. Schwarz, D. Gruss et al., *Meltdown: Reading Kernel Memory from User Space*, 27th {USENIX} Security Symposium, 2018
21. ERC20 Token Standard, Ethereum Foundation.
22. J. Kwon, "Tendermint: Consensus without Mining," Cornell, 2014
23. V. Costan, S. Devadasi, Intel SGX Explained, *IACR Cryptology ePrint Archive*, 2016

VII. Biographies

D. Jeff Dionne is the CEO and founder of Core Semiconductor. Prior to that he was the CEO/CTO of Smart Energy Instruments, and also CEO of Arcturus Networks, and CTO of Lineo. His research interests include open source software and hardware, semiconductor design, signal processing, networking, network security, embedded operating systems, and analog design. He is perhaps most well known as the co-creator of the µClinux OS, which was the original foundation for all variants of embedded Linux.

Jen McCann is the VP of Engineering at Core Semiconductor. Prior to that she was the VP of Engineering of Smart Energy Instruments and also held engineering positions at Texas Instruments, Novell and CIBC. Her research interests include software testing, quality and validation. She holds both a Bachelor of Engineering and a Masters of Management Science degrees from the University of Waterloo.

Kenneth Tang is currently the Marketing Manager at Core Semiconductor. Prior to that he was the Marketing Manager at Fast Access Blockchain, Director of Technology Partnerships at Smart Energy Instruments and Quality Assurance Manager at ALT Software. His research interests are in the areas of embedded systems, blockchain, digital signal processing, software quality, and smart grid. He holds a Bachelor’s degree in Electrical Engineering from the University of Toronto.

Jeff Garzik is the co-founder and CEO of Bloq and Metronome, both companies in the blockchain space, and a director of The Linux Foundation. Previously he was one of the 3 original Bitcoin core developers, the CEO of Dunvegan Space Systems, the CEO of exMULTI, and a Principal Software Engineer at Red Hat. Jeff serves on the board of Coin Center, and the advisory board of BitFury, BitPay, Chain.com, Netki and WayPaver Labs. He holds a degree in computer science from the Georgia Institute of Technology.