cryptography

MDPI

Article

Securing Additive Manufacturing with Blockchains and Distributed Physically Unclonable Functions

Bertrand Cambou [1,*], Michael Gowanlock [1], Julie Heynssens [1], Saloni Jain [1], Christopher Philabaum [1], Duane Booher [1], Ian Burke [1], Jack Garrard [1], Donald Telesca [2] and Laurent Njilla [2]

[1] School of Informatics, Computing, and Cyber Systems, Northern Arizona University, Flagstaff, AZ 86011, USA; michael.gowanlock@nau.edu (M.G.); julie.heynssens@nau.edu (J.H.); sj779@nau.edu (S.J.); cp723@nau.edu (C.P.); Duane.booher@nau.edu (D.B.); imb29@nau.edu (I.B.); jg2562@nau.edu (J.G.)

[2] Air Force Research Laboratory, Rome, NY 13441, USA; Donald.telesca@us.af.mil (D.T.); Laurent.njilla@us.af.mil (L.N.)

* Correspondence: Bertrand.cambou@nau.edu; Tel.: +1-928-523-7824

Received: 28 April 2020; Accepted: 15 June 2020; Published: 18 June 2020

Abstract: Blockchain technology is a game-changing, enhancing security for the supply chain of smart additive manufacturing. Blockchain enables the tracking and recording of the history of each transaction in a ledger stored in the cloud that cannot be altered, and when blockchain is combined with digital signatures, it verifies the identity of the participants with its non-repudiation capabilities. One of the weaknesses of blockchain is the difficulty of preventing malicious participants from gaining access to public–private key pairs. Groups of opponents often interact freely with the network, and this is a security concern when cloud-based methods manage the key pairs. Therefore, we are proposing end-to-end security schemes by both inserting tamper-resistant devices in the hardware of the peripheral devices and using ternary cryptography. The tamper-resistant devices, which are designed with nanomaterials, act as Physical Unclonable Functions to generate secret cryptographic keys. One-time use public–private key pairs are generated for each transaction. In addition, the cryptographic scheme incorporates a third logic state to mitigate man-in-the-middle attacks. The generation of these public–private key pairs is compatible with post quantum cryptography. The third scheme we are proposing is the use of noise injection techniques used with high-performance computing to increase the security of the system. We present prototypes to demonstrate the feasibility of these schemes and to quantify the relevant parameters. We conclude by presenting the value of blockchains to secure the logistics of additive manufacturing operations.

Keywords: blockchain; digital signatures; key distribution; additive manufacturing; ternary cryptography; physical unclonable functions; high-performance computing

1. Introduction and Objectives

The objective of the work presented in this paper is to enhance the level of security of "Additive Manufacturing", which we define as the set of manufacturing operations that add value to a product through a distributed production process that relies on a network of subcontractors and suppliers interacting through the cloud and open internet communications. Mainstream manufacturing operations are increasingly specialized and have to learn how to efficiently incorporate Additive Manufacturing operations. Some of the inherent problems associated with Additive Manufacturing operations are growing in importance due to the difficulty of controlling the performance of the subcontractors and suppliers and the exposure to well-organized cyber criminals. Examples of

threats include defective parts, discontinuation of supply, counterfeits, hardware and software Trojans, loss of intellectual property, and abuse of confidential information. Blockchain technology can secure smart manufacturing through non-alterable ledgers and non-repudiable transactions. Blockchain technology is based on one-way cryptographic functions such as hash functions, which are considered extremely safe. One of the weakest links of the technology is the distribution, storage, and use of the public–private key pairs. For example, private keys can be stolen through side channel analysis during the digital signature generation. Illegitimate users that have access to stolen cryptographic keys can participate in and thus damage the integrity of Additive Manufacturing operations. An additional worry is the long-term weakness of the mathematical algorithms behind digital signatures, such as Elliptic Curve Cryptography (ECC). The threats are coming from the rapid progress in the fabrication of quantum computers, which will eventually be powerful enough to break some mathematical algorithms. For this reason, several governments are already banning the use of asymmetrical cryptographic algorithms such as ECC.605. The objective of the work presented in this paper is to mitigate these security limitations by adding an end-to-end cryptographic system utilizing both hardware and software that also uses elements of ternary computing, and a hardware network of Physical Unclonable Functions (PUFs) inserted in distributed client devices. The research question at the core of this paper is to demonstrate that it is possible to generate a new public key for each transaction with acceptable latencies. Therefore, the leakage of the keys is less critical considering that the same key is never used twice. We propose a migration path to replace ECC by Post Quantum Cryptographic (PQC) methods, while continuing to generate private keys with random streams extracted directly from the PUFs.

At the server level, the algorithms are based on ternary cryptographic schemes in which the third state includes both unstable cells and randomly selected states to confuse the opponents. We use mainstream "binary hardware" to be able to use off-the-shelf systems. Ternary computing with binary infrastructure can be valuable for cryptography [1]. The third state of ternary logic is especially important in our cryptographic protocol and offers additional layers of security and increases the level of entropy, i.e., randomness. This scheme exploits the fuzzy (unstable) cells of PUFs, allows additional noise injection into the cryptographic keys, and also enables the random masking of stable cells. For the opponent, the added complexity needed to handle this third state is high in terms of memory sizes and latencies and will offer a significant barrier against brute force attacks trying to uncover unknown ternary states. Servers with High-Performance Computing (HPC) can protect networks of client devices using only small microcontrollers [2]. The software used in this work for "ternary cryptography" is the following:

(i) The Ternary Addressable Public Key Infrastructure protocol (TAPKI) [3];
(ii) The Response-Based Cryptographic method (RBC) [2];
(iii) SHA-256 hash functions; and
(iv) EC-DSA for digital signatures.

At the client level, the objective is to minimize complexity in the following ways:

(i) The software is purely binary,
(ii) The PUFs are read only once at each handshake to generate key pairs from responses, no error correcting schemes are needed, and
(iii) The cryptography selected, written in C, is mainstream for low-cost embedded secure microcontrollers. As part of the anticipated optimization of this work, the cryptography used by the client devices (hash functions, public key generation, digital signature) will be directly executable in hardware with low-cost cryptoprocessors.

In future work, we are considering designing hardware handling native ternary logic to reduce both memory sizes and latencies. The instruction set of a native ternary computer is not universal and can be changed on demand as part of the ternary cryptographic protocol.

This multi-faceted paper serves as a pathfinder for securing smart manufacturing using blockchain technology, with the following contributions

- [Section 3] We describe the motivation for an architecture that secures additive manufacturing. To obviate the security risks of client devices using a single private key, we employ Ternary Addressable Public Key Infrastructure (TAPKI) that generates public/private key pairs using a PUF each time a client authenticates a blockchain transaction with a secure server. The migration of the mathematical algorithm of the digital signatures from ECC to PQC is presented, while the PUFs are still used to generate the private keys.
- [Section 4] To authenticate a client device, the server must perform a search over the key space using the initial response recorded from each client's PUF. To implement this system, we describe several technical challenges that need to be addressed. This culminates in an end-to-end exploratory prototype using two client devices equipped with Static Random Access Memory (SRAM)-based PUFs using a server for authentication. We examine important parameters in the protocol, such as tolerable latencies and error rates. This prototype demonstrates the practicality of the proposed architecture; we can secure blockchain using off-the-shelf-components. To the best of our knowledge, no other protocols have been published that generate public–private key pairs on demand from PUFs to secure the Digital Signature Algorithms (DSA) of blockchain technology.
- [Section 5] Using the exploratory prototype described above, we outline potential security weaknesses that shoul
- [Section 6] The server must authenticate the client's public–private key pairs each time the client wishes to update the blockchain. This requires an extensive search over the key space to correct the errors in the key generated by a client's PUF. We prototyped a response-based cryptography scheme using ECC that leverages high-performance computing resources. The proposed massively parallel search enables low-latency authentications.
- [Sections 7 and 8] We conclude the paper. This pathfinding work outlines several substantial future projects that in aggregate will enable a robust ecosystem of technologies that secure the DSA for blockchain for additive manufacturing.

2. Background and Related Work

In this section, we describe relevant background information and define the blockchain technology, additive manufacturing, and Physical Unclonable Functions (PUFs).

2.1. Blockchain Technology

In 2008, the paper published under the name Satoshi Nakamoto "Bitcoin: a peer-to-peer electronic cash system" [4] created a revolution for the financial world. Two relatively mature technologies, the hash functions with Merkle trees and Digital Signature Algorithms (DSA) [5–12] were successfully integrated in the architecture to track all transactions in a virtual public leger that is non-alterable and non-repudiable. The innovation behind Bitcoin included a trust management scheme using aspects of game theory that allowed the generation, i.e., mining, of additional cryptocurrencies. The "peer-to-peer" aspect of the scheme can be controversial for some, due to its inability to prevent suspicious users from participating. However, many believe that the blockchain technology also has the potential to evolve and gain as much importance as the existing internet technology.

Blockchain technology with a hashing function such as the Standard Hash Algorithm (SHA) SHA-2 can protect the data flow needed to track transactions for applications such as (see Figure 1) personal information, finance, transportation, logistic, and smart manufacturing. The digital signature used to secure bitcoins is based on Elliptic Curve Cryptography (ECC) [13,14] and has low power capabilities, which is desirable for the Internet of Things (IoT) infrastructure. The underlying assumption is that the entire infrastructure of IoT is homogeneous, with each node being protected by a cryptoprocessor handling the hashing and having a secure non-volatile memory to store the cryptographic keys. The DSA can be based on, but not limited to, an extended finite field ECC (also called Galois

Fields ECC: GF-ECC), which operates at lower power than the older finite field ECC, and Rivest Shamir Adelman (RSA) [15]. Malicious side channel attacks, and the physical hijacking of IoT nodes, can expose the private keys, thereby compromising the security of the infrastructure. The distribution of public–private key pairs in such an environment can be risky. Without the reliable protection of private keys, the DSA of the blockchain is vulnerable, and the technology loses its security value. ECC is also not quantum computing-resistant [16] and should be replaced by alternate DSA methods such as the ones offered by hash-based, lattice, code, and multivariate cryptography [17–19].

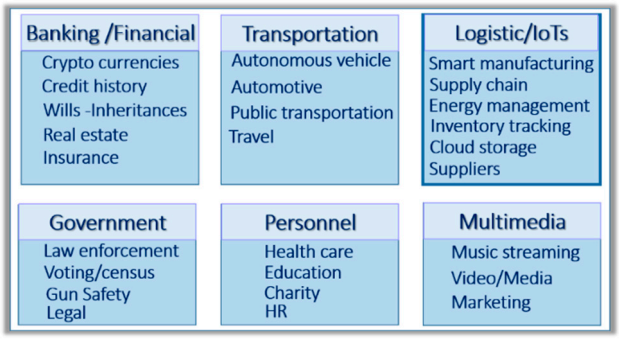

Figure 1. Example of fields that could benefit from the blockchain technology.

2.2. Additive Manufacturing

Manufacturing operations are outsourcing an increasingly large proportion of the supply chain through networks of interconnected multi-echelon subcontractors that interact remotely with the worldwide web [20–23]. The risks of cyberattacks are rapidly accelerating, including the emergence of counterfeit products, poor quality elements of unknown sources, and the insertion of hardware Trojans, malwares, worms, and viruses in electronic components. The traditional centralized mechanism cannot control the full supply chain due to conflicting interests with certain suppliers, and the vulnerability of heterogeneous Information Technology Systems (ITS) to malicious entities.

Blockchain technology with DSA brings transparency [24–26], with traceable, non-alterable, and non-repudiable transactions, and decentralized governance in support of multiple layers of suppliers and subcontractors. In order to avoid unwelcomed suppliers, the management of the public keys of the DSA can be tracked by Certificate Authorities (CA) that are trusted by the manufacturing entity [27], as shown in Figure 2. The list of the valid public keys is available in the cloud. Therefore, all trusted suppliers and subcontractors can share and directly verify the transactions transmitted by their peers. The DSA by the suppliers follows the following steps: hash the transaction, signature with private key, and post the information in the cloud. The tracking can incorporate information that is required by the manufacturing operation such as origin, quantity, quality, proof of sustainability [28,29], intellectual property [30], copyrights [31], and a list of subcontractors. It has been suggested that the blockchain technology can also track the identification of the purchased parts with RFIDs and tokens [32]. The use of blockchain to secure traditional manufacturing that actually manufactures all of their product is off topic in this paper. The readers interested in securing integrated circuit manufacturing are invited to read the summary paper presented by Guin and DiMase [33].

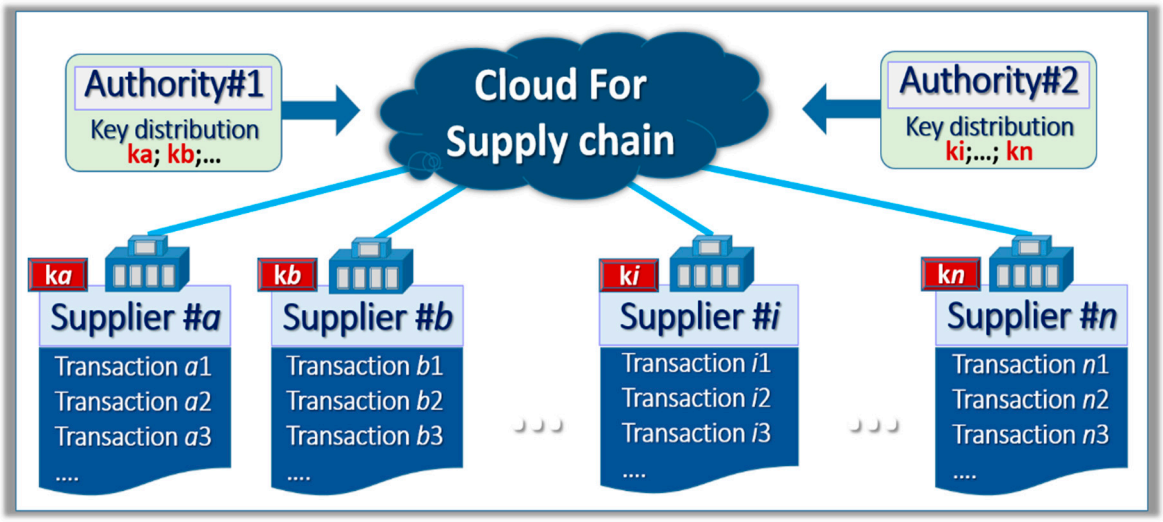

Figure 2. Example of certificate authorities distributing valid public keys to a constellation of suppliers.

2.3. Ternary Physical Unclonable Functions

PUF technology exploits the variations created during fabrication to differentiate each device from all other devices, acting as a hardware "fingerprint" [34–36]. Solutions based on PUFs embedded in the hardware of each supplier node can mitigate the risk of an opponent reading the keys stored in non-volatile memory. Instead, the keys for the DSA of the blockchains are generated on demand. Authentication protocols based on PUFs, embedded in each IoT node, are effective with (1) intra-PUF stability, (2) inter-PUF randomness, and (3) small enough drifts of the PUF characteristics over time. Memory structures with nanotechnology [37], SRAM [38], DRAM [39], non-volatile memories and Flash [40,41], ReRAM [42], and MRAM [43] are suitable to generate strong PUFs. In the protocols selected in this work, the initial readings of the PUFs, also called the "initial responses", are the result of computations and statistical analysis to sort out the cells that are solidly identified as a logical "0" or "1" and the unstable fuzzy cells that are identified with an additional third state "X". During the enrollment cycle of PUFs, the three potential states (0,1,X) of the PUF's cells are stored in look-up tables in secure servers; enrollment has to be done only once in a secure environment. The PUF "responses" refer to the data generated during the life of the client devices. The client devices operate with protocols that are as simple as possible; this includes the use of binary states (0,1), not ternary, and low-complexity read cycles. During authentication cycles in which the PUFs are "challenged", the "initial responses", which are stored in the server, are compared with the "responses" read from the client device. This results in matching "Challenge–Response Pairs" (CRP) when the "responses" are the same than the "initial responses".

The PUFs can age, and they are subject to environmental drift, electromagnetic interference, and aging. When the CRP error rates are below 10%, the false rejection rate (FRR) and false acceptance rate (FAR) are usually acceptable, and the PUFs can be used as part of authentication protocols to protect cyber physical systems. The use of PUFs to generate cryptographic keys from the responses, a focus of this work, is more challenging than generating responses for authentication. A single-bit mismatch in a cryptographic key is not acceptable for most encryption protocols. Therefore, the use of error correcting methods [44], helper data [45–47], and fuzzy extractors [48,49] is needed to achieve the zero error level required. Error correcting schemes burden client devices, as they consume additional computing power to run fuzzy extraction and error correcting codes. Such error correcting protocols also increase the vulnerability to differential power analysis that leak information to opponents. With ternary PUFs [1,50], when the fuzzy "X" states are blanked, CRP error rates are typically reduced by two orders of magnitudes to the 10^{-3} range, which greatly simplifies the entire error correcting

protocol. Conversely, when the "X" values are selected, CRP error rates are higher, which can be used as a feature when HPC is used to handle the erratic keys as presented below.

3. Description of the Architecture that Enhances the Security of Additive Manufacturing

3.1. Overview

The blockchain architecture presented in Section 2.2 has the potential to enhance the security of Additive Manufacturing; however, it is vulnerable when the public–private key pairs of suppliers are compromised. As shown in Table 1, the security of blockchains requires several technologies [51–53].

- The first layer of blockchain technology security is from hash pointers and Merkle trees that generate non-alterable public ledgers. As existing hash algorithms such as SHA-2 and SHA-3 are considered safe, no further improvements are suggested.
- The second layer of Table 1, digital signatures and the storing and handling of the public–private key pairs can become a major liability for Additive Manufacturing. The prime objective of the work presented in this paper is to enhance security in this area.
- The third layer of security shown in Table 1 is based on trust mechanisms relying on peer-based groups and is a vibrant field of research [54–60]. For example, Distributed Ledger Technology (DLT) registers transactions in multiple locations simultaneously without a central administration or certificate authority. Such transactions are fast and could be less vulnerable to certain cyberattacks. Different DLTs are available such as Ethereum, International Occultation Timing Association (IOTA), and others. However, the manufacturing of strategic assets such as weapons, planes, and satellites cannot rely solely on peer-based trust mechanisms. The prototype developed to generate key pairs to secure the digital signatures does require a strong certificate authority. This does not preclude the use of DLT in combination of the proposed architecture to expand its capabilities; the development of such combination is outside the scope of the paper.

Table 1. Layers of security offered by the blockchain technology.

1- Hash Pointers and Merkel Trees → blockchains Chain of messages with non-alterable public ledgers
2- Digital signatures with public/private key pairs Identification of the users and non-repudiation
3- Trust mechanisms Majority rules against small participants Maximize revenues by following the rules

The architecture shown in Figure 3 includes the following protections:

- Ternary Addressable Public Key Infrastructure (TAPKI) for the generation of private keys from the PUF. During enrollment, the image of the PUF and the initial responses are stored in a look-up table of the CA. New keys are generated for each transaction by the TAPKI.
- The generation of public key pairs and a path toward post quantum cryptography;
- Response-Based Cryptographic (RBC) scheme to verify that the public keys generated from the private keys and the TAPKI are valid with acceptable error rates;
- A cryptographic scheme that uses noise injection in the PUF and HPC to mitigate attacks from opponents that do not have access to similar computing power.
- RBC verifies the validity of the public keys and posts them in a public ledger.

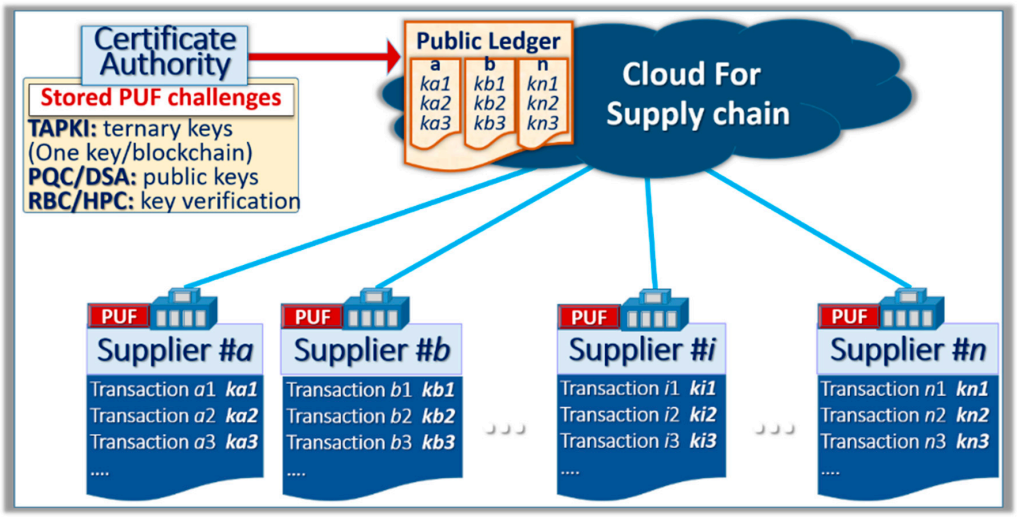

Figure 3. Physical Unclonable Functions (PUFs) distribution, and ternary cryptography, to generate new keys at each transaction.

3.2. Ternary Addressable Public Key Infrastructure (TAPKI)

The term Public Key Infrastructure (PKI) has been used to describe an environment where each communicating party is equipped with two keys: the private key that is secret and the public key that is openly available. With asymmetrical cryptographic schemes, the messages are encrypted with one of the two keys and decrypted with the second key. In the case of the DSA for blockchains, the author of a blockchain encrypts the signature with the private key in such a way that anyone can verify the signature with the public key. These private keys can be stolen by various methods including during the generation, distribution, and storage of the keys, as well as during encryption/decryption cycles. The objective of the TAPKI [4], see Figure 4, is to provide additional security to PKI by generating a new private key for each blockchain transaction from distributed ternary PUFs. To sign a new blockchain, the server transmits the information needed by the supplier to generate a new private key from the Ternary PUF. The information is shared with a communication channel that is assumed to be insecure. The random number $\boldsymbol{T}$ generated at each transaction by the TAPKI concurrently feeds two hashing elements at the server and the supplier levels. The number $\boldsymbol{T}$ is concatenated with the password $\boldsymbol{PW}$ and additional multifactor schemes to generate the message digest $\boldsymbol{A_i}$. The message digest is turned into a particular address $\{\boldsymbol{X_i}, \boldsymbol{Y_j}\}$ of the PUF array; see Figure 4. For example, if the PUF array contains 1024×1024 cells for a one mega-bit of memory, 10 digits of the message digest are used for $\boldsymbol{X_i}$ and 10 digits are used for $\boldsymbol{Y_i}$.

Only the server with the appropriate look-up table and the client device with its PUF can independently generate the same private key for the TAPKI protocol; a third party without the same look-up table cannot find the same address $\{\boldsymbol{X_i}, \boldsymbol{Y_j}\}$. At this address, the server extracts a ternary stream $\boldsymbol{C_i}$ from the initial responses stored in the look-up table, and the supplier reads a binary stream $\boldsymbol{C_i'}$ from the PUF. Both streams should be similar, with some errors present in $\mathbf{C_i'}$ due to the natural physical variations of the PUF. The server generates a mask $\boldsymbol{M_i}$ to blank the ternary cells and then generates a key $\boldsymbol{K}$ that contains only the solid cells with "0"s and "1"s. The mask is XORed with the message digest $\boldsymbol{A_i}$ to generate the stream $\boldsymbol{S}$, which is communicated to the supplier as part of the handshake. This XOR operation encrypts the mask $\boldsymbol{M_i}$; this encryption method is also called a one-time pad because both the mask and the message digest are only used once during each handshake. The knowledge of S will not disclose either $\boldsymbol{M_i}$ or $\boldsymbol{A_i}$. The supplier can again XOR the stream $\boldsymbol{S}$ with the message digest $\boldsymbol{A_i}$ to recover the mask. Both the server with the look-up table and the client device with the PUF will explore the same portion of the array with $\boldsymbol{A_i}$ and mask the same cells with $\boldsymbol{M_i}$ to independently generate the keys $\boldsymbol{K}$ (server) and $\boldsymbol{K'}$, which are similar when the error rates are low.

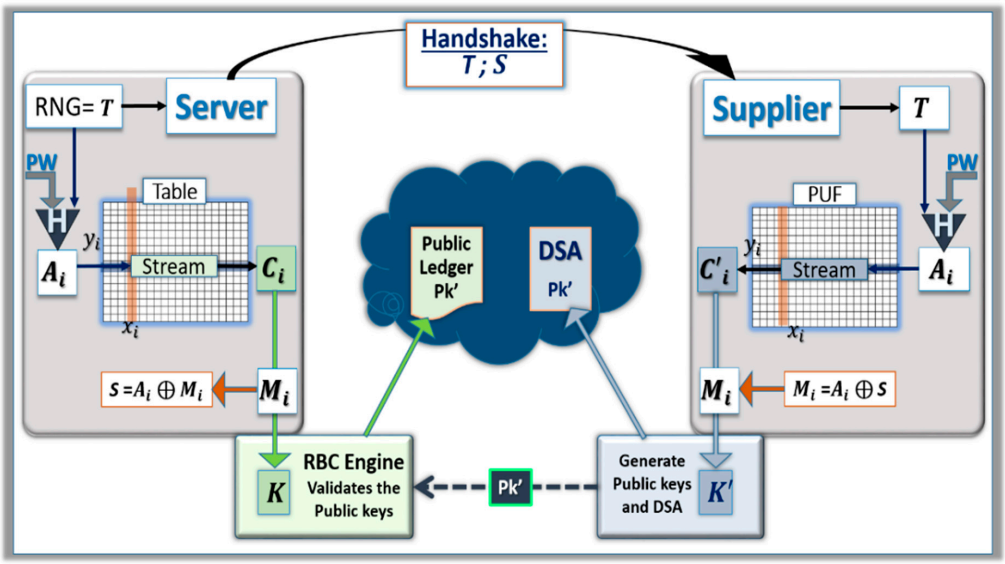

Figure 4. Generation/verification of public keys: The protocol with multifactor authentication finds the addresses to generate PUF responses. The private keys are generated by masking the fuzzy cells, the public keys are generated with Elliptic Curve Cryptography (ECC). The Response-Based Cryptographic method (RBC) engine finds the public keys ***PK'***. The Digital Signature Algorithms (DSAs)/public keys are posted.

Figure 5 shows an example of the sequential scheme used to generate a new public key ***k4j*** for the blockchain ***Block4j***, with a random number ***RN4j***, and ***Mask4j***. This figure simplifies the protocol and does not include the protection of the mask with the XOR presented in Figure 4. The fuzzy cells of the ternary PUF and associated ternary states offer a protection against man-in-the-middle attacks sending their own handshakes. When an opponent sends random streams T_f and S_f to the supplier, the errors of the key K' generated from data stream C_i' will contain high error rates, because an invalid mask does not blank the fuzzy cells. The client needs to have the right password ***PW*** to retrieve the right message digest A_i from a random number T_f. The man-in-the middle does not know the mask, ***PW***, and $\underline{\mathbf{A_i}}$, so the stream S_f will be random, and $S_f \oplus \mathbf{A_i}$ will be an invalid mask. Therefore, the erratic keys generated by the supplier under such a handshake will not be recognizable by the server.

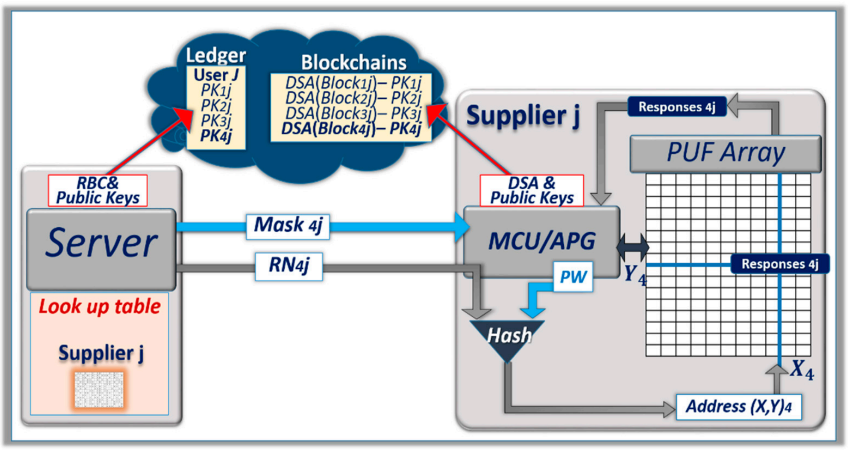

Figure 5. Architecture generating new public keys for each blockchain. The circuitry around the PUF drives the Ternary Addressable Public Key Infrastructure (TAPKI) protocol. The cells at address $\mathbf{(X,Y)_4}$ are used to generate the 512 bit-long ***Response 4j***. With ***Mask 4j***, 256-bit long key pairs are generated. The public key PK_{4j} and the DSA of $Block_{4j}$ are posted.

3.3. Generation of the Public Keys—PQC Considerations

The DSA protecting blockchains uses the private keys, which are natural numbers, typically 256 bits long, while the verification algorithms use the public keys. With ECC, the public keys are computed by multiplying the primitive element of the cyclic group by the private keys. The reverse computation, finding private keys from public keys, requires enormous processing power; this protects the encryption method. In the proposed protocol, TAPKI is acting as a key exchange mechanism for the private keys using the ternary PUFs, while the public keys are generated by an asymmetrical cryptographic scheme such as ECC. TAPKI is a generic method; we employ ECC for evaluation purposes to design the prototype described in Section 4 and to validate the overall architecture. With ECC, a single bit mismatch between the private key $\boldsymbol{K'}$ generated from the PUF and the private key $\boldsymbol{K}$ generated from the look-up table will result in entirely different public keys. The RBC scheme mitigates this problem (described in Section 3.4). The natural mismatch of the private keys $\boldsymbol{K'}$ and $\boldsymbol{K}$ is considered a feature in this work (see Section 3.5), which is leveraged to enhance the security of the network of suppliers for Smart Manufacturing.

It is now anticipated that Quantum Computers (QC) will be able to break ECC when the technology to design enough quantum nodes becomes available. N. Koblitz and A. J. Menezes in their paper "A Riddle Wrapped in an Enigma" suggested that the ban of ECC by the National Security Agency is unavoidable, the risk of QC being only one element of the problem [61]. Plans to replace ECC by PQC schemes have been developed for the DSA using blockchain technology, even if the timeline for the availability of powerful QC is highly speculative [62–65]. The efforts required to implement the blockchain technology for Smart Manufacturing is such that a plan to prepare the migration to PQC-DSA is needed, even if non-PQC schemes are used at first. The project driven by the National Institute of Standards and Technology (NIST) has pre-selected nine potential PQC-DSA candidates during the round 2 phase of the program [66]: SPHINCS and PICNIC with hash-based cryptography [67–71]; CRYSTALS, FALCON, and qTESLA with lattice cryptography [67,72–74]; and GeMSS, LUOV, MQDSS, and Rainbow with multivariate cryptography [75–78]. The software developed for in this work for blockchain, in particular TAPKI and RBC, should be applicable to these PQC-DSA schemes to replace Elliptic Curve DSA (EC-DSA). Therefore, we analyzed the possibility to generate the private keys from ternary PUFs for PQC-DSA schemes and then to generate the public keys from these private keys to sign and verify messages. NIST has encouraged all candidates for the PQC program to post online the codes and supporting documentation. The summary of our analysis is presented below.

3.3.1. Hash-Based PQC-DSA (SPHINCS, PICNIC)

The PQC-DSA algorithm SPHINCS+ relies on well-known hash-based signature schemes, such as Winternitz One Time Signature (WOTS), Forest of Random Subsets (FORS), and a set of Merkle trees called hyper-trees [67]. The size of these hyper-trees is such that an almost infinite number of signatures can be generated with the same tree. The sizes of the keys are relatively small (256 to 512 bits); however, architectures such as hyper-trees need multiple layers of keys, which could be heavy to manage. PICNIC uses zero-knowledge algorithms [68]. The disadvantage of hash-based cryptography is the high latencies to sign and verify; due to the need to perform large quantities of hashing, the size of the signatures could be quite large. The use of TAPKI and PUFs to generate the key pairs for PQC hash algorithms is also challenging because even small levels of defectivity in the PUFs can be prohibitive. One way to simplify the scheme is to use the TAPKI and PUF to generate the seeds needed in the hash-based PQC, rather than generating the private keys from the PUFs. The seeds are much smaller than the resulting private keys. Therefore, we propose the following protocol: use the PUFs to get the random numbers needed to generate the seeds; then, generate the private keys from the seeds, and finally, generate the public keys from the private keys with PQC algorithms rather than ECC. The authors successfully tested such protocol with SPHINCS and SRAM PUFs, using the C-codes available online. The preliminary results were encouraging in terms of latencies, and we intend to publish the final results at a later date after comprehensive characterization.

At the client/supplier device side, the latencies can be reduced with hardware implementation of the hashing functions. At the server level, which can have access to parallel computing architectures and graphic processor units, the latencies can be mitigated to an acceptable level.

3.3.2. Lattice-Based PQC-DSA (CRYSTAL, qTESLA, and FALCON)

Lattice-based algorithms exploit hardness to resolve problems such as the Closest Vector Problem (CVP) and Learning With Error (LWE) algorithms and share some similarities with the knapsack cryptographic problem.

- The public–private key pair generation of CRYSTAL is based on polynomial computation in a lattice ring [68]. During key generation, a matrix $\boldsymbol{A}$ is generated with random numbers, and the two vectors $\mathbf{s_1}$ and $\mathbf{s_2}$ are generated with relatively small numbers. With these elements, the vector $\mathbf{t}$ is computed as $\boldsymbol{t} = \boldsymbol{As_1} + \boldsymbol{s_2}$; both $\mathbf{A}$ and $\mathbf{t}$ become the public key, while $\mathbf{s_1}$ and $\mathbf{s_2}$ become the private key. One method to implement CRYSTAL with the TAPKI protocol is to have the handshake pointing at three set of addresses in the PUF to generate $\boldsymbol{A}$, $\mathbf{s_1}$, and $\mathbf{s_2}$, then compute $\mathbf{t}$. The DSAs are signed using the private keys and verified using the public keys. An alternate method to implement CRYSTAL is to use the handshake to send $\boldsymbol{A}$ and the addresses to find $\mathbf{s_1}$ and $\mathbf{s_2}$, and then compute $\mathbf{t}$. This second method is not as secure, but it does not have to mitigate potential errors due to the PUFs in the generation of matrix $\boldsymbol{A}$.
- The key generation of qTESLA, which is also based on polynomial computation in a lattice ring, uses a $\boldsymbol{seed_a}$ to generate the matrix A, $\boldsymbol{seed_s}$ to generate the vector $\mathbf{s}$ and $\boldsymbol{seed_{e_1,\ldots,\ e_k}}$ for the vector of error. The vector $\mathbf{t}$ is computed from $\boldsymbol{A}$, $\boldsymbol{s}$, and the vector of error [73]. An additional $\boldsymbol{seed_y}$ is needed at each signature cycle to generate a vector $\boldsymbol{y}$ that signs the next message. In the preliminary implementation, we generated a pre-seed from the SRAM-based PUF, which we used to generate the seeds $\boldsymbol{seed_a}$, $\boldsymbol{seed_s}$, and $\boldsymbol{seed_{e_1,\ldots,\ e_k}}$, thereby generating the private–public key pairs. The preliminary results were also encouraging in terms of latencies, comparable to EC-DSA, and we intend to publish the final results at a later date after characterization.
- FALCON, which uses NTRU (***Nth*** degree of **TRU**ncated polynomial ring) arithmetic [72], is based on methods to generate public–private key pairs that can be implemented with TAPKI schemes. The PUFs can replace random number generators to find private keys; however, the resulting polynomial elements are not always usable, as they are subject to some pre-conditions. The server will need to try several possible TAPKI handshakes and select the ones giving acceptable private keys. The generation of the public keys from the private keys is based on inverse modulo computation.

3.3.3. Multivariate PQC-DSA (GeMSS, LUOV, MQDSS, and Rainbow)

The private keys for multivariate-based PQC-DSA algorithms are generated with numbers forming invertible matrix and polynomials. The TAPKI and ternary PUFs can replace the random number generators. The public keys are derived from the private keys; the signature of the DSA uses the private keys; and the verification uses the public keys. These multivariate methods have been known for a long time, and the size of their signature can be small. However, it is still unknown if the performance and size of the public keys will be competitive with other methods.

The list of recommended PQC-DSA should be reduced by NIST in the next two years, and the final recommendations are expected to be announced in the 2023–2025 window. TAPKI will be easier to implement with PQC-DSA algorithms based on relatively small key pairs; we anticipate that NIST will select DSA algorithms with small keys. The framework presented in this paper is using EC-DSA as a transitional technology toward PQC-DSA, using PUFs as sources of random number generators for the various seeds needed for PQC. Once NIST finalizes DSAs, we will implement the TAPKI protocol with one of these approved algorithms.

3.4. Response-Based Cryptography for Public Key Verification

In the scheme described in Figure 4, the secret key ***K'*** generated by the client device with TAPKI is slightly different from the key ***K*** generated by the server due to the errors caused by the drift of the physical parameters of the PUF. RBC is the important scheme needed to validate the public key ***PK'*** used in the DSA scheme of the blockchain [2]. RBC is a search engine that finds the uncorrected responses of the PUF, i.e., the private key ***K'***. As shown in Figure 6, the starting point of the search is the reference key ***K*** stored in the look-up table of the server.

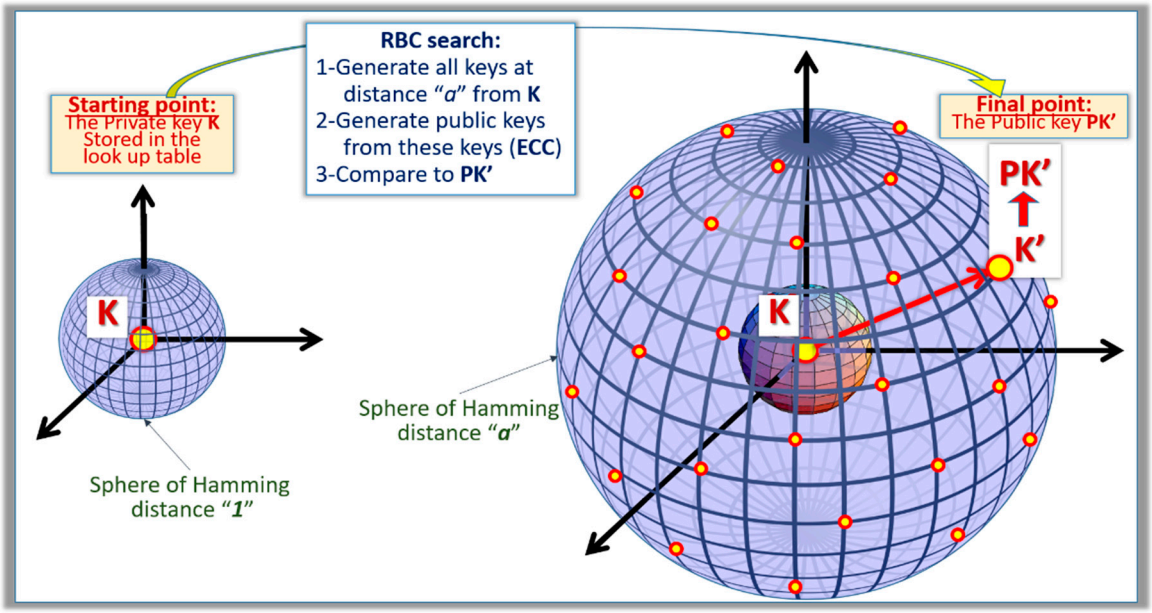

Figure 6. Graphical representation of the RBC. The search starts with ***K***, and the public key is ***PK'***. All possible keys at a Hamming distance "***a***" are located in the sphere shown in the right, including ***K'***.

The objective of the search is to find ***K'*** that is a stream with "***a***" errors; i.e., the Hamming distance between both streams is "***a***". The search algorithm is an iterative process:

- Step 0: A public key ***PK*** is generated from ***K*** and compared with ***PK'***, which is known. If they are equal, the search stops;
- Step 1: All keys at a Hamming distance of one from ***K*** are generated with their associated public keys. If one public key matches ***PK'***, the search stops;
- Step ***a***: All keys at a Hamming distance of "***a***" from ***K*** are generated with their associated public keys. If one public key matches ***PK'***, the search stops;
- Step ***a***+1: When the RBC search is positive, ***PK'*** is posted in the public ledger as valid.

The RBC method is effective when the error rate is sufficiently low. If the error rates are high, the latencies are prohibitive. For example, with 256-bit keys, RBC can find keys having 3 to 4 errors, which correspond to an error rate of 1.5%. Ternary PUFs characterized in the experimental section have error rates below 0.1%, which is well within the search capabilities of the RBC scheme. Conversely, the typical error rates of the PUFs without ternary states and the blanking of fuzzy states are in the 5% to 10% range. The average latency $\mathbf{A_{(\lambda,N)}}$ of the RBC search for N-bit long PUFs with an average number of erratic bits $\boldsymbol{\lambda}$ is given by:

$$\mathbf{A_{(\lambda,N)}} = \boldsymbol{\tau_o} \sum_{X=0}^{X=N} \mathbf{P_\lambda(X)} \left[\sum_{i=0}^{i=X} \binom{N}{i} - \frac{1}{2} \binom{N}{X}\right] \approx \boldsymbol{\tau_o} \frac{1}{2} \binom{N}{L}. \quad (1)$$

- $\mathbf{P_\lambda(X)}$ is the probability of having **X** erratic bits in the **N**-bit long keys, with $\boldsymbol{\lambda}$ erratic bits;

- τ_o is the average latency to generate a public key from a private key and to compare it to ***PK'***;
- **L** is the integer number greater than λ: **L-1**< λ ≤ **L** (the approximation is correct when λ is large).

The use of PQC-DSA schemes with slower key pair generation could result in high latencies with lower efficiencies of the RBC search, which could be a limiting factor of the scheme. In order to be able to use PUFs with higher rate of errors and PQC-DSA with higher latency, we recommend implementing a scheme with key fragmentation. The general concept behind this operation is summarized in Figure 7. The keys are fragmented into **k** segments, and padding is used to keep the resulting sub-keys at the same length. In the development described in the experimental section, the error-free padding information is shared as part of the handshake. For a fragmentation by four, we used the 512-bit long stream **S**, which is defined in Section 3.2 as $\mathbf{Mi} \oplus \mathbf{A_i}$. The first 192 bits of **S** are used to pad the first key, the next 192 bits are used to pad the second key. The last 128 bits of **S** are combined with the first 64 bits of **S** to pad the third key. Finally, the bits 65 to 256 are used to pad the fourth key. Public keys are generated from the **k** sub-keys feeding the RBC search engine.

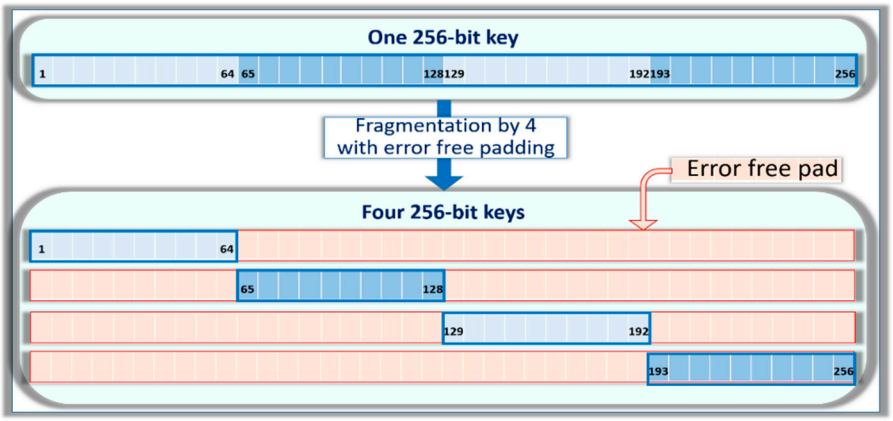

Figure 7. Example of fragmentation by 4 of a 256-bit long key cut into four pieces (1–64), (65–128), (129–192), and (193–256). Error-free padding is used to form four sub-keys, which are analyzed by the RBC engine.

When **k** is an integer number dividing **N**, **N/k** must be an integer number as well. The average latency $\mathbf{A_{k(\lambda,N)}}$ of the RBC search with fragmentation by **k** is given by:

$$\mathbf{A_{k(\lambda,N)}} = \mathbf{k} \times \mathbf{A_{(\lambda/k,N/k)}} \tag{2}$$

$$\mathbf{A_{k(\lambda,N)}} = \mathbf{k}\,\tau_o \sum_{X=0}^{X=N} \mathbf{P_{\lambda/k}(X)} \left[\sum_{i=0}^{i=X} \binom{\mathbf{N/k}}{\mathbf{i}} - \frac{1}{2}\binom{\mathbf{N/k}}{\mathbf{X}}\right] \approx \mathbf{k}\,\tau_o \frac{1}{2}\binom{\mathbf{N/k}}{\mathbf{L/k}}]. \tag{3}$$

- $\mathbf{A_{(\lambda/k,N/k)}}$ is the average latency of the search with **N/k** bit long keys and **λ/k** average erratic bits;
- **L/k** is the integer greater than **λ/k**: **(L/k)**-1< **λ/k**≤ **L/k** (approximation correct when **λ** is large);

With fragmentation, the RBC search latencies are greatly reduced. For example, when **N** = 256, **λ** = 16, **k** = 4 the ratio between the latencies without and with fragmentation is:

$$\mathbf{A_{(16,256)}}/\mathbf{A_{4(16,256)}} \approx 1/4\binom{256}{16}/\binom{64}{4} = 1/4(1.0 \times 10^{25})/(6.4 \times 10^{5}) = 3.9 \times 10^{18}. \tag{4}$$

It is desirable to minimize the fragmentation levels to reduce electronic power at the supplier level. During the experimental work, based on a 200 MHz MIPS RISC microcontroller, we measured that one cycle of public key generation with ECC took less than 100 μs; fragmentation by 8 can be done well within 1 ms. This latency is reduced by two orders of magnitude when the supplier operates with a commercial 4GHz quad core PC, which is mainstream in Smart Manufacturing. A PQC-DSA

technology that operates with public key generation that is 100,000 times slower than ECC will be still acceptable on a PC with latencies around one second.

3.5. Noise Injection and HPC

The validity verification of public keys by the CA is critical for a network of suppliers involved in Smart Manufacturing. Conversely, the CA could become a target for the opponent. The computing cluster used in this work has 2500 effective cores, and the error density of the ternary PUFs can be adjusted from 0.01% to 10% by changing the fuzzy cell masking. As presented above, when the error rates of the PUFs are approximately lower than 1.0%, the computing power of commercially available PCs is enough for the RBC search to quickly verify a public key. The concept presented is noise injection in the PUF to generate highly noisy keys, so that only CAs equipped with HPC resources can be effective in the public key generation, thereby restricting access to opponents with inferior computing power. An example of a sequence that was developed based on Equation (3) is shown in Table 2. The ternary PUF using commercially available SRAMs was set up so that the challenge–response pair error rates averaged 0.05%. This was done by submitting the SRAMs to 100 repetitive power off–on cycles and only keeping the cells awaking as solid "0" or "1" states. About 20% of the SRAM cells were blanked, and the resulting mapping was stored in the look-up table of the server. The noise is injected in 256-bit long keys by randomly flipping 36 bits, representing an approximate 14% error rate. Our models are showing that with a fragmentation by 4, the HPC can verify a public key in 1.2 s, while the expected latency of the same search with a commercial PC is approximately 1.4 days. Thereby, if the maximum acceptable time to verify a public key by the CA is set around 5.0 s, only powerful HPCs can reduce FRR.

Table 2. Example of sequence to protect a network with High-Performance Computing (HPC).

1- Ternary PUF	Error rates ≈ 0.05%
2- Noise injection in the PUF	14% in 256-bit long keys
3- Use fragmentation by 4	
4- Use HPC for RBC search	2000 cores
5- Average RBC latency with HPC	1.2 s
6- Average RBC latency with PC	1.4 days
7- Maximum acceptable latency	5.0 s

3.5.1. Importance of Eliminating Fuzzy Cells.

The use of ternary PUFs that mask fuzzy cells enhances the stability of the scheme. With ternary PUF error rates in the 0.05% range and a normal distribution, the natural variations are such that the probability to have three bad bits or more on 256-bit long key is 3.18×10^{-4}, which makes the minimization of the false rejection rates (FRR) of the HPC search relatively easy. Conversely, a PUF having 4% error rates will face the natural variations shown in Figure 8, from 2 to 20 errors, which makes the protocol with HPC described in Table 2 hard to implement. Assuming that the noise injector adds 10% bad bits in a 256-bit long key, the HPC will not be able to find the erratic keys when the PUF errors are on the high end of the distribution, thereby resulting in FRR. When the errors are at the low end of the normal distribution, a regular PC is anticipated to be able to find the erratic keys, which defeats the purpose of the scheme. In summary, the injection of noise to discriminate HPC versus PC is only effective when the PUFs have low error rates.

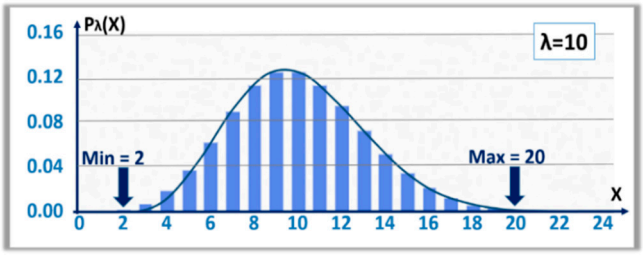

Figure 8. Normal distribution of the number of errors X for a 256-bit long key and λ = 10.

3.5.2. Fragmentation to Widen the Window of Operation

As presented in Section 3.3, key fragmentation allows the use of PUFs with higher error rates. This fragmentation method can also widen the scheme's window of operation using noise injection and HPC; see Figure 9. Without fragmentation, an injection of approximately 1.5% bad bits into 256-bit long keys differentiates the use of HPC from regular PC. The PCs are not powerful enough for RBC. However, the addition of few bad bits would increase FRR to non-acceptable levels, even with an HPC-based search. With key fragmentation by 4, the injection of 7% to 15% bad bits into 256-bit long keys differentiates the use of HPC from PC. This represents a wide window of operation in which the FRR of HPC could be set extremely low. The sequence proposed in Table 2 is set at the high end of the window, i.e., 14%, to prevent the effectiveness of more powerful PCs.

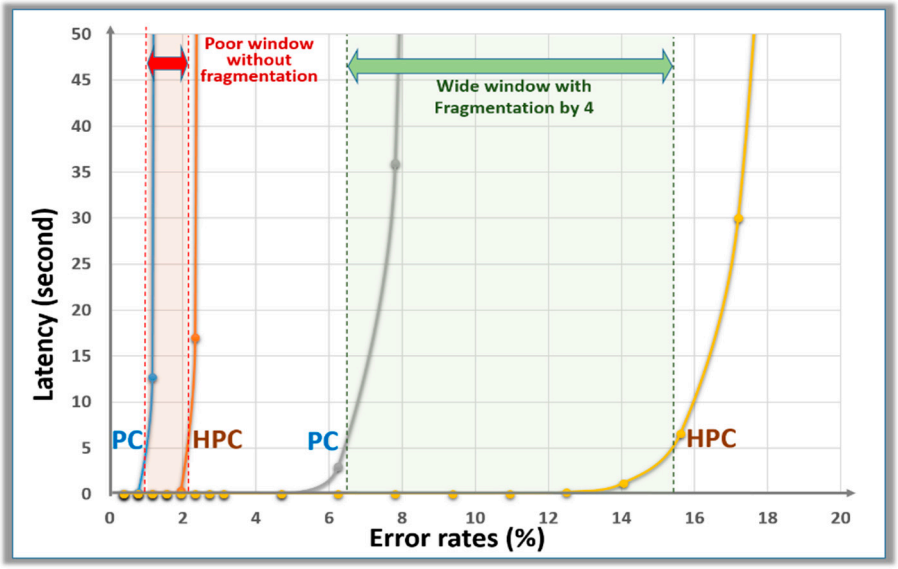

Figure 9. Modeling the latencies of RBC searches for 256-bit long keys. With a fragmentation by 4, this window increases to 9%: PCs can only operate below 6.5%, whereas HPC work up to 15.5%.

4. End-to-End Exploratory Prototype

As stated in Section 2, additive manufacturing refers to the decentralized nature of production. Manufacturing is no longer centralized and relies on multiple subcontractors participating. This change yields new security vulnerabilities that did not exist in the centralized production model. Hence, technologies are needed to ensure traceability during production such that malicious actors are unable to influence the production process. To eliminate risk, each time a supplier adds value to a product in the production process, the supplier updates the blockchain using a low-powered client device equipped with a PUF that generates private keys on demand. The transaction is authenticated by a server that verifies the public–private key pair supplied by the client. Only products that are validated by each supplier in the production chain are considered secure. Therefore, malicious actors are isolated

from the production process, and any subcontractors not conforming to the protocol can be easily detected by tracing the historical information stored in the blockchain.

Using the architecture described in Section 3, we implemented an exploratory end-to-end prototype that uses off-the-shelf components to enable secure additive manufacturing. The goal of this prototype is to characterize system performance and explore security vulnerabilities. We begin by outlining the problem statement below.

Problem statement: Consider the large design space for PUF-based DSA for securing blockchain technology for additive manufacturing. Such a system requires client devices that can initiate a transaction with a secure server. The server authenticates client transactions through the response-based cryptography protocol to validate that the client's PUF-generated public–private key pair is authentic based on each client device's initially recorded response. While there are many technologies that can be used for each system component, and each variation in system architecture leads to different security vulnerabilities, we realized one such end-to-end system. By realizing a prototype system, we can explore security vulnerabilities germane to the selected architecture and its variants. As a guide for our design decisions, we elected to use exclusively off-the-shelf components. For example, we used SRAM-based PUF technology that is potentially sensitive to side channel analysis and key leakage. However, it is an excellent technology in terms of entropy, with relatively low CRP error rates and stability. In short, we built a system with SRAM-based PUF technology, ECC key exchange, WiFire microcontrollers and chipkits, a laptop, and tablets.

We summarized the key vulnerabilities found in the prototype and propose solutions to these problems in Section 5. While we recognize the exploratory nature of this prototype, to our knowledge, no other systems have been published that secure the DSA for blockchain using PUF-generated public–private key pairs.

4.1. Description of the Prototype

Commercially available components, such as SRAMs, SHA-512, and ECC have been selected to validate the protocol securing Smart Manufacturing with blockchains. The ternary PUFs were designed with SRAM and private key generation using TAPKI schemes. One of the challenges of this development was the public key-matching algorithm with RBC, which allows the server to independently recognize the public keys generated by the ternary PUFs of each client device. On the client device, the objective was to implement the ECC key exchange and the DSA protocol as part of TAPKI in a microcontroller environment with relatively low computing power. On the server side, the objective was to implement ECC key exchange as part of the RBC search algorithm, which is executable on both PCs. One of the complexities of the overall project was to develop a software stack working in such a heterogeneous computing environment, from low-end microcontrollers to Windows-based PCs, and to HPC. The designed for this work is summarized in Figure 10.

The WiFire microcontrollers fabricated by Digilent drive the two client devices. The custom daughter cards handle the SRAM PUFs and the wireless connectivity. The tablet PCs are used to enter messages and to display the message digests, digital signatures, and public keys. The protocol developed includes the following steps:

- Step-1: Alice enters the message in the first tablet PC in plain text;
- Step-2: Generation of the private keys with TAPKI and the handshake between the CA (i.e., the PC) and the microcontroller board;
- Step-3: The microcontroller hashes the message, signs it with the private key, and generates a public key with ECC. The resulting information is displayed on the screen of the tablet PC;
- Step-4: The same information is transmitted to Bob's microcontroller board;
- Step-5: Bob's microcontroller board verifies with the CA that the public key is valid, verifies that the signature is valid, and displays the information to the screen of the second tablet PC. The RBC search is performed on the PC with key fragmentations by four to validate the public key.

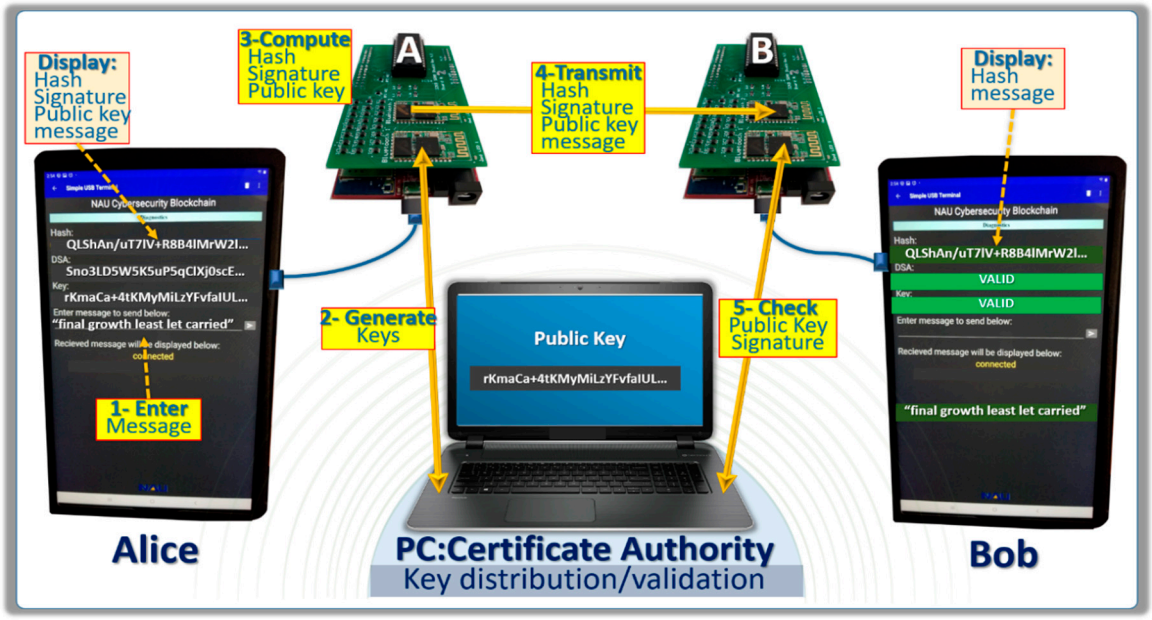

Figure 10. Overall representation of the prototype developed to demonstrate the architecture. The TAPKI handshake is used to concurrently generate keys. The public key generation, DSA, and verifications are based on ECC. The RBC search validates the public keys. The results of the transaction are posted on Bob's tablet PC.

4.2. Design of the Client Devices

To interface commercially available SRAM with the ChipKit WiFire microcontrollers, a custom daughter card or shield is needed. This is a custom PCB that allows additional hardware components to be placed on top of the ChipKit microcontroller. Before using the shields, breadboards and jumper wires were used to connect to the SRAM. The breadboard setup worked slowly and was less reliable. One of the biggest issues faced was figuring out how to power down the SRAM for responses without completely powering down the entire microcontroller and project setup. To avoid these issues and create a smaller, more compact hardware package, the design of a custom PCB was implemented. With the shield PCB design being the next step in the project, research was done on which components to use to manage the SRAM's power and IO. We also needed a way to incorporate wireless hardware peer-to-peer communication. After much prototyping, 26 analog switches for SRAM I/O management were used to quickly power off the devices, and two HC-06 Bluetooth modules were used for wireless communication. For most of the prototyping phase, desktop workstations or laptops were used to interface with the microcontrollers. The interfacing entailed a simple way to read out diagnostics, message data, and verifications through a computer terminal. Moving forward using a desktop or even multiple laptops for a portable demonstration is not ideal, so to make the demonstration more portable, we moved to Android tablets. We chose Android tablets because we could easily implement through Android Studio along with using open source apps to assist in data management. The Samsung 10.1 inch Tab A tablet was chosen for this. It meets the power standards for providing power to the microcontroller and shield components, along with being able to handle serial communication for interfacing. Figure 11 shows the app layout that displays required diagnostics, verification checks, and messages. An example of sequence demonstrated in the prototype is shown below:

- Step-1: Message randomly generated by Alice:
  - ***'final growth least let carried'***
    (0x66696e616c2067726f777468206c65617374206c6574206361727272696564)
- Step-2: Key generated by the TAPKI and Alice's WireFire Chipkit:

- Random number exchanged during the handshake:
  - ***39b5b15badd904619ea98424a5545e49bd725ffa9d959bcd604d3232a2f471945a69699 4ce9 8a2568b49dfeec698cb001daff100c629fc46090456a292c4b1e7***
- Private key:
  - ***e2e4e4dbf34cba3177425cd7df5d21b20ae0c2660316cd396f0608e5e7a1fc7b296893dbbb3***
  - ***a369a9839a64063aee6606dfcbedf496a4bdfbdee123cd2a0472***
- Public key generated with ECC:
  - ***rKmaCa+4tKMyMiLzYFvfaIULIdemLQBBvPocyMi6iaxRV7NNbyvyR9Wb2sb TaBoG5ayHIS1sQHFYvtPn1s1gHA==***

- Step-3: Computing by Alice's WiFire Chipkit
  - Hashing of the message with SHA512:
    - ***QLShAn/uT7lV+R8B4lMrW2XClETs8/tlzxaPmDAs1hiv1dYSOhxs7JduzUMuZ zrZpUWBHhjKuW0Gx7skfEAe7g==***
  - Digital signature of the message digest with the private key and ECC:
    - ***Sno3LD5W5K5uP5qClXj0scEuCH+6bFyCqsT4MQbcwQ4tZF08raCHHMJ51p dvecBTmTns7ZqGz9/DNsGGupSsgg==***
- Step-4: Information transmitted to Bob's Chipkit: message, message digest, signature, and
  - Public key. This information is posted on the screen of both tablets;
- Step-5: Verification by Bob's WiFire Chipkit:
  - The PC verifies the validity of the public key with RBC;
  - Bob's Chipkit hashes Alice's message with SHA-512 to check the message digest;
  - Bob's Chipkit verifies the validity of the signature with the public key and ECC.

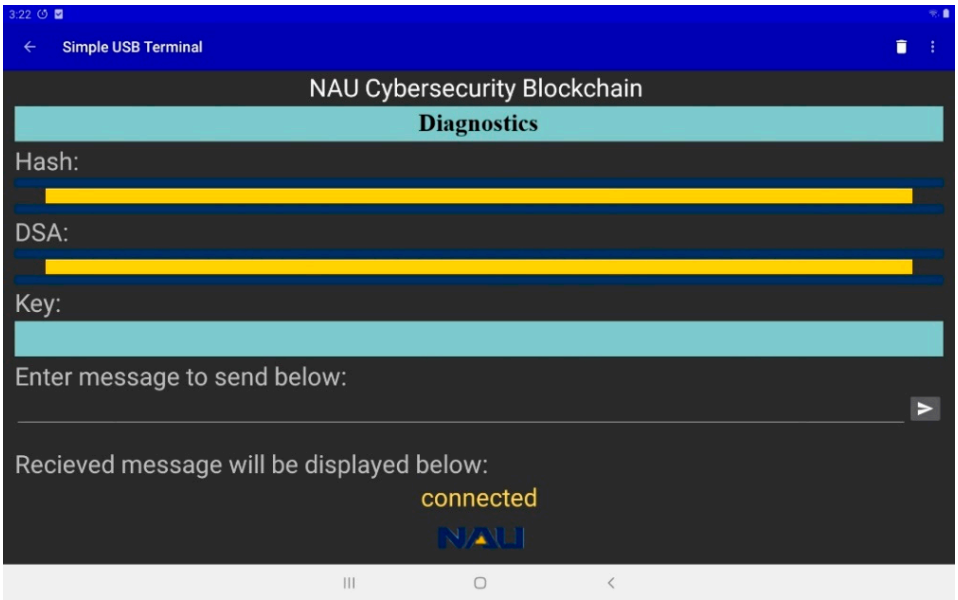

Figure 11. App screenshot of the tablet to display information.

After the manual entry of plaintext in the first tablet PC having variable lengths, the latencies of the entire protocol lasted less than one second; the second tablet displays the plain text, its message digest, and the validation of the DSA within 500 ms. The generation of the 256-bit long public keys from the private keys with ECC takes 10,000 clock cycles; the generation of the 256-bit long private keys from the PUF takes 800 clock cycles. For reference, other PKI protocols such as RSA are much slower. The key pair generation with RSA takes 500,000 clock cycles for 1500-bit long keys, which have the same cryptographic strength as the 256-bit long keys for ECC. In both cases, ECC or RSA, the latency of the generation of the private keys from the PUFs is negligible. The CRP error rates of the SRAM-based PUFs with unstable cell masking were in the sub 10^{-4} range. No false rejects of the RBC search were observed over thousands of cycles and several months of repetitive testing. We also tested the protocol with SRAM PUFs without masking unstable cells, showing CRP error rates in the 5% range. With fragmentation by 8, we were able to get similar results: latencies around 500 ms and no observable false rejects. To the best of our knowledge, no other protocols have been published that generate one-time use public–private keys pairs from PUFs to secure the DSA of blockchain technology with such latencies and no observable FRR of the keys.

5. Security Considerations of the End-to-End Prototype

In this architecture, the tablets, the communication between the tablets and the WiFire Chipkits, the wireless communication between the two Chipkits, and the communication from Chipkits to PC are all assumed to be vulnerable and non-secure. The purpose of the tablets is to display non-secure publicly available information: messages, message digests, digital signatures, and public keys. This publicly available information is freely transmitted from the tablet PC to Chipkit, and from Chipkit to Chipkit. The TAPKI handshake from the PC to the ChipKit is also publicly available information, which is protected by multifactor authentications of the Chipkit such as passwords, pin codes, biometric prints, and PUF CRPs. The most vulnerable link of the architecture is the Chipkit and the daughterboard with the SRAM PUF. Examples of vulnerabilities include:

- Loss of the Chipkits to the opponents, who will directly attack the SRAM PUFs, read the mapping of the responses, and generate a look-up table, similar to the one stored by the CA, or a clone of the client device to fool the CA;
- Side channel analysis to extract the private keys during generation from the PUF, or during the public key generation from the private keys, and during the digital signature cycles also from the private keys. Examples of side channel analysis include Differential Power Analysis (DPA), fault injections, and the use of sensing elements of the electromagnetic radiation generated by the Chipkit;
- Generic software attacks between the client device and the CA such as fake client devices to generate malicious blockchains from unauthorized users. Then, the users could interface with fake CA, pretending to be legitimate;
- Neutralization of certain client devices with malware injection, Denial of Service (DoS) attacks, confusion of the PUF with thermal or Electro Magnetic Interference (EMI) attacks;
- Attacks directed at the CA to steal the look-up tables of the PUFs and develop fake CAs handling the constellation of client devices.

The design of the WiFire Chipkit uses generic components, which are by definition non-secure. The implementation of the proposed scheme will require a set of improvements such as the following:

- Replacement of the SRAM by tamper-resistant components. When lost to the opponent, the responses of the SRAM PUFs are relatively easy to extract. Advanced memory devices such as Resistive RAM and Magnetic RAM can be used to design lower power PUFs, which are more difficult to break [42,43];
- Use of encryption and protection schemes to generate PUF responses that prevent wide leakages of the content of the PUF;

- Design of a custom secure microcontroller chip integrating the PUF, the cryptoprocessor, and an Reduced Instruction Set Instruction (RISC) processor, with hardware implementation of the cryptographic protocols. Commercial SIM and banking cards are currently leveraging powerful secure microcontroller chips with wireless connectivity that could replace the WiFire Chipkit and interact directly with a tablet or another terminal device. Commercial secure microcontrollers are equipped with counter measures against side channel analysis, DPA, and physical attacks;
- Implement multifactor authentication of the CA to mitigate man-in-the-middle attacks and the entry of malicious CAs. The look-up table of the CA, which stores the PUF challenges, and the initial responses can provide one of these factors.

In an additive manufacturing environment, the entities managing their suppliers usually have a stringent process to qualify their suppliers. The delivery of a secure microcontroller with a PUF for each is rather simple from a logistical standpoint. Before delivery, the managing entities will capture the image of the PUF and store it in a look-up table on their server, which can be handled in a highly secure environment. The responsibility of the managing entities will be to implement a process to protect their servers from the opponent and to act as a CA for the constellation of supplier. The manufacturers of strategic assets usually have access to powerful servers and HPC resources.

6. High-Performance Computing for RBC

6.1. Description of the Schemes Driving the HPC

We implemented the response-based cryptography protocol described in Sections 3.4 and 3.5. Our implementation is written in C and is parallelized using Message Passing Interface (MPI) [78] and Multi-Threaded Programming (Pthreads) [79]. Let **KS(a, k)** be the total key space, containing all of the **N** = 256-bit keys that need to be searched using the starting key **K** (known by the server), with a hamming distance of **a**, using fragmentation **k**. Since *on average*, a key will be found halfway through the search at hamming distance **a,** the total number of keys searched, with fragmentation **k**, is as follows:

$$\left|\mathbf{KS}(\mathbf{a},\ \mathbf{k})\right| = \sum_{\mathbf{i}=0}^{\mathbf{a}-1} \binom{\frac{256}{\mathbf{k}}}{\mathbf{i}} + \frac{1}{2}\binom{\frac{256}{\mathbf{k}}}{\mathbf{a}}. \tag{5}$$

Given the total key space, we assign $\left|\mathbf{KS}(\mathbf{a},\ \mathbf{k})\right|/\mathbf{p}$ keys to search for each MPI process rank, where there are **p** physical cores on our platform**.** Without the loss of generality, we assume that **p** evenly divides $\left|\mathbf{KS}(\mathbf{a},\ \mathbf{k})\right|$. When one rank finds the correct key, ***PK'***, the search needs to terminate. There are several methods that could be employed to terminate the search; however, some methods lead to unacceptable overhead. We briefly describe our search termination procedure as follows. Each MPI rank creates two threads (implemented using Pthreads).

One thread performs the search for the correct key, while the other thread performs communication between ranks. If a rank finds the correct key, then this information is sent to all other process ranks, and their respective communication threads terminate the search at each rank. To ensure that each communication thread consumes few computational resources, which would otherwise be used by the search thread, we use the Iprobe functionality in MPI that performs a non-blocking check for the message that indicates that the search needs to be terminated. We adjust a parameter that determines how often we check for this message to reach a trade-off between the message checking overhead and the number of wasted searches, where wasted searches refer to those searches that are performed after the key has been found.

6.2. Statistical Analysis with HPC

Our code, which is written in C, has been posted in GitHub; in all experiments, we use 256-bit keys and average response times over 10 trials: https://github.com/GiantDarth/hamming_validator. The code is compiled using the O3 compiler optimization flag and is compiled using the GNU compiler

v.6.2.0. As described in Section 3.4, a PUF will have an error rate that follows a distribution. In our experiments, we select a single Hamming distance that does not vary as a function of distribution (e.g., the distribution in Figure 8). Since the algorithm response time will be impacted based on when the key is found within the search space, we elect to fix the key found in the middle of the key space at Hamming distance **a**, such that we achieve the *average case* response time. This average case is outlined in Section 3.4 and 3.5. All experiments are carried out on the Monsoon cluster at Northern Arizona University (NAU). In our experiments, we use two dedicated computer nodes. Each node has 2×2.6 GHz Intel Xeon Gold 6132 processors with $2 \times 14 = 28$ physical cores. All experiments were performed on 64 physical cores across nodes. Regarding the experimental results, we note the following caveat: our implementation may contain remaining errors. Therefore, the response times reported in this section may not be accurate in absolute terms. The experiments in this section are thereby preliminary and may change in future implementations. We will be conducting a detailed performance evaluation of RBC for ECC. Despite this caveat, we find that the reported measurements are in general agreement with the expected latencies derived by the model. Figure 12 plots the measured response time versus Hamming distance for $\mathbf{k} = 1$, no fragmentation; $\mathbf{k} = 4$; and $\mathbf{k} = 8$.

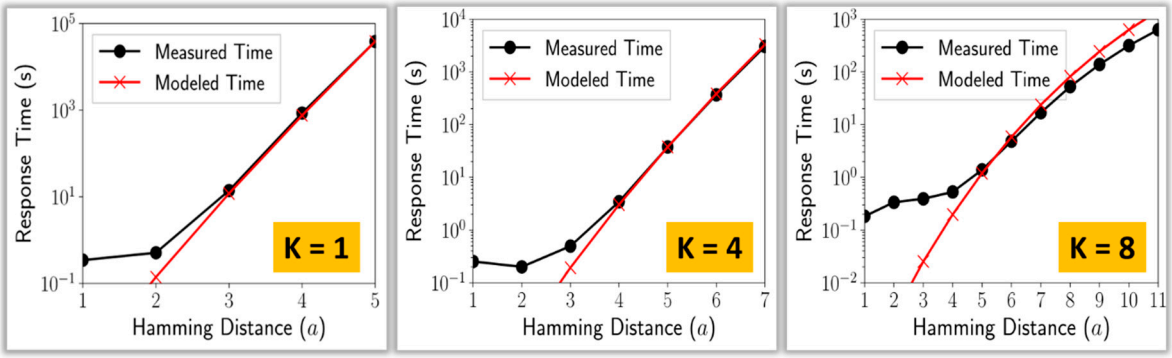

Figure 12. Measured and modeled latencies for various levels of fragmentations.

In all experiments, we use $\mathbf{p} = 64$ physical cores. Since increasing the Hamming distance exponentially increases the search space, we plot the response time on a log scale. In Figure 12 left with $\mathbf{k} = 1$ (no fragmentation), we find that at Hamming distance $\mathbf{a} = 3$–5, and using $\mathbf{p} = 64$ cores, we cannot find the key in a reasonable amount of time, where only $\mathbf{a} < 3$ is practical for the search. For example, at $\mathbf{a} = 5$, the key is found in 38,570 s. At the other extreme, Figure 12 right plots $\mathbf{k} = 8$, where the key can be found within 1.39 s at $\mathbf{a} = 5$. This shows that the use of fragmentation increases the range of practical Hamming distances, **a.** Consequently, when implementing RBC in practice, the values of **k** and **a** can be carefully selected based on **p** to achieve the desired key authentication throughput. Although we limited $\mathbf{p} = 64$ in this evaluation, our implementation is expected to achieve good scalability on larger core counts. We model the response time of the search to determine whether the expected performance is impacted by any of the search parameters. We first measure the constant $\boldsymbol{\tau_o}$, which is the time to perform one ECC calculation. Since $\boldsymbol{\tau_o}$ is implementation-dependent, it must be experimentally derived. We find that $\boldsymbol{\tau_o} = 8.417 \times 10^{-6}$ s on our platform using $\mathbf{p} = 64$ cores. Using the number of keys, $\left|\mathbf{KS}(\mathbf{a},\ \mathbf{k})\right|$, as a function of the Hamming distance, **a**, and fragmentation **k**, and the value of $\boldsymbol{\tau_o}$, our model is simply $\boldsymbol{\tau_o}\left|\mathbf{KS}(\mathbf{a},\ \mathbf{k})\right|$. Figure 12 compares the measured and modeled algorithm response time. On the smaller workloads (low Hamming distance), we find that the model underestimates the total response time. This is because there are overheads associated with the implementation that are amortized on the larger workloads but are not amortized on the smaller workloads. Overall, we find that our model can capture the performance behavior of the search.

7. Implementation and Implications for Additive Manufacturing

Additive Manufacturing (AM) creates an object by adding layers of material from three-dimensional data. By comparison, traditional, or subtractive, manufacturing processes are where the product is created by cutting away material from a larger piece [80]. Due to the numerous technical and economic advantages, AM is expected to become a dominant manufacturing technology in both industrial and home settings. The United States (US) National Defense Authorization Act for the Fiscal Year 2017 US Senate Report "strongly encouraged" the US Department of Defense (DoD) to more aggressively pursue AM capabilities to improve readiness and enable the Military Services to be more self-sustainable. The Office of the Deputy Assistant Secretary of Defense for Manufacturing and Industrial Base Policy is the US DoD AM lead that oversees the implementation of AM and reports to the Under Secretary of Defense for Research and Engineering [81]. The growing penetration of AM at manufacturers across the world and the dependence of this technology on computerization have already raised security concerns, some of which have been proven experimentally [81].

The parts themselves have now become new targets for cyber criminals. More specifically, the parts' "digital twin", the digital file that contains the parts' specifications and manufacturing instructions, now becomes a vulnerability. This is due to the dependency of the effectiveness of AM almost entirely on the integrity of digital files to instruct the 3D printing mechanism [82]. To ensure the integrity and traceability of digital files and assure their secure delivery at each stage in the supply chain, ranging from the file developer all the way to the end user, more companies are turning to blockchain. Blockchain functions as a distributed database that maintains a continuously growing list of ordered records. Blockchain works by storing information, in this case design files, across each phase of the digital supply chain. The phases would include design, distribution, manufacturing, and in-field use on any participating nodes. If an additive manufacturing supply chain implemented blockchain at these transactional node levels, it has the potential to assure that all assets were traceable and their provenance known. Users would have the capability to see and trace the full lifecycle of the part.

Having secure blockchain architecture becomes a cornerstone toward securing AM capabilities. The current weakness of blockchain technology is the protection of the private keys. When stored in the non-volatile memory or when they are too weak, this becomes a vulnerability. A paper presented by Independent Security Evaluators (ISE) discovered that funds from weak key addresses are being pilfered and sent to a destination address belonging to an individual or group that is running active campaigns. On January 13, 2018, this "blockchainbandit" held a balance of 37,926 ETH valued at $54,343,407 [83]. The work presented in this paper demonstrates a solution based on PUFs embedded in the hardware of each supplier node as an effective mitigation to the private key weakness of blockchain technology. This will increase the reliability and resilience of the AM process.

8. Conclusion and Future Work

The authors recognize that one of the most impressive aspects of the technology behind Bitcoin, the elimination of a central authority in favor of a peer-to-peer trust mechanism, is not included in the proposed architecture. We argue that the Smart Manufacturing of strategic assets with networks of suppliers can benefit from certificate authorities restricting the list of suppliers, monitoring public key infrastructures, and the validity of the digital signatures. Such a restrictive environment can still benefit from non-alterable, non-repudiable ledgers, resulting from hash functions and digital signatures. The prototype developed in this research work demonstrates that commercially available SRAM-based PUFs with ternary cryptographic schemes can generate highly reliable one-time use public–private key pairs for the digital signature of each blockchain. We experimentally verified that the latencies to generate keys, hash the messages, and sign them are in the 500 ms range; the FRR, due to erratic ternary PUF responses, is extremely low. The commercially available WiFire Chipkits with custom daughter cards are relatively low power, and it is expected that they can be replaced by custom secure integrated circuits with complexity similar to mainstream SIM cards. Adding HPC and noise injection

to the private key generation is going one step further in the direction of establishing strong CAs that monitor the key distribution to known suppliers. The very preliminary data generated experimentally by our HPC seem to validate the models proposed to optimize RBC search latencies. For example, with masked SRAM-based PUFs having low error rates, the injection of about 14% bad bits into 256-bit long keys, and RBC search using fragmentation by four, our HPC can verify the validity of public keys within seconds, while regular PCs are not powerful enough to perform such verification. The scheme is anticipated to increase the cost to break the supplier-based smart manufacturing environment using the blockchain technology.

The future work envisioned by the authors includes:

- **Replacement of the SRAM-based PUF by tamper-resistant components**. Two memory technologies are considered: Resistive RAM and Magnetic RAM. The architecture suggested in this paper is agnostic on the type of PUF selected, as long as the defect density is low enough. Note that that the masking methodology proposed is effective to reduce the defect density of the SRAM PUFs from 5% to 10^{-5}. The effort needed to get similar results with the ReRAMs and the MRAMs is not under-estimated;
- **Replacement of the Elliptic Curve Digital Signature by quantum-resistant** *DSA*. Both hash and lattice-based cryptographic schemes that are currently under consideration by the NIST-driven PQC program are excellent candidates. We intend to take an early look at SPHINCS, CRYSTAL, and qTESLA, which are compatible with PUF-based private key generation. The main figure of merits of the novel PQC-DSAs that will be characterized are the latencies for the generation of public keys from the private keys and erratic public key generation. The RBC search involves large quantities of public key generation; therefore, excessive latencies will be prohibitive. We will investigate if HPC/GPU technology can reduce these latencies. The second important figure of merit is the size of the private keys. Long keys are statistically more sensitive to erratic bits generated from the PUF responses;
- **Optimization of the HPC/GPU**. The work presented in this paper is preliminary and will require a lengthy investigation. Several parameters of the RBC search can be optimized to enhance the efficiency of HPC and GPU, namely the level of fragmentation, the type of noise injection, the use of DSA algorithms, and the ways to concurrently assign tasks to processors;
- **Enhancing the levels of security of the architecture.** We mentioned in Section 5 potential attacks against the proposed architecture; remedies are needed to mitigate these vulnerabilities. We also intend to involve third parties to highlight additional potential weaknesses.

In conclusion, the proposed architecture, which uses distributed PUFs and ternary cryptographic schemes, has the potential to enhance security of the blockchain technology when applied to the logistics of Smart Manufacturing. The prototype developed is encouraging; however, the implementation will require significant additional resources and third-party assessment.

Author Contributions: Conceptualization, all; methodology, all; software, C.P., D.B., and M.G.; validation, I.B., J.G., J.H.; formal analysis, S.J., M.G.; data curation, M.G.; writing—original draft preparation, all; writing—review and editing, B.C., M.G., D.T., L.N.; supervision, B.C., D.T., L.N.; project administration, B.C.; funding acquisition, B.C. All authors have read and agreed to the published version of the manuscript.

Funding: Contractor acknowledges Government's support in the publication of this paper. This material is partially based upon the work funded by the Information Directorate, under the AFRL grant number FA8750-19-2-0503. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of AFRL.

Acknowledgments: The authors are thanking the staff, students and faculty from Northern Arizona University (NAU), in particular Christopher Coffey who is managing NAU's HPC, Vince Rodriguez, Nan Duan, Mohammed Mohammadi, and Sareh Assiri. We are also thanking the professionals of the Air Force Research laboratory (AFRL) of Rome, New York (US), who supported this effort.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Cambou, B.; Flikkema, P.; Palmer, J.; Telesca, D.; Philabaum, C. Can Ternary Computing Improve Information Assurance? *Cryptography* **2018**, *2*, 6. [CrossRef]
2. Cambou, B.; Philabaum, C.; Booher, D.; Telesca, D. *Response-Based Cryptographic Methods with Ternary Physical Unclonable Functions*; Springer Science and Business Media LLC: Cham, Switzerland, 2019.
3. Cambou, B.; Telesca, D. Ternary Computing to Strengthen Information Assurance, Development of Ternary State based public key exchange. In Proceedings of the Computing Conference, London, UK, 10–12 July 2018.
4. Nakamoto, S. Bitcoin: A Peer to Peer Electronic Cash System. 2008. Available online: www.biticoin.org (accessed on 1 June 2020).
5. Croman, K.; Decker, C.; Eyal, I.; Gencer, A.E.; Juels, A.; Kosba, A.; Miller, A. On scaling decentralized blockchains. In Proceedings of the 3rd Workshop on Bitcoin and Blockchain Research, Bridgetown, Barbados, 26 February 2016.
6. Luu, L.; Narayanan, V.; Zheng, C.; Baweja, K.; Gilbert, S.; Saxena, P. A secure sharing protocol for open blockchains. In Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016.
7. Eyal, I.; Gencer, A.E.; Sirer, E.G.; Renesse, R.V. Bitcoin-NG: A Scalable Blockchain Protocol. In Proceedings of the 13th Symposium on Networked Systems Design and Implementation, Santa Clara, CA, USA, 16–18 March 2016.
8. Dorri, A.; Kanhere, S.S.; Jurdak, R. Blockchain in internet of things: Challenges and solutions. *MathJax* **2016**. Available online: https://arxiv.org/ftp/arxiv/papers/1608/1608.05187.pdf (accessed on 1 June 2020).
9. Gervais, A.; Karame, G.O.; Wüst, K.; Glykantzis, V.; Ritzdorf, H.; Capkun, S. On the security and performance of proof of work blockchains. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016.
10. Zheng, Z.; Xie, S.; Dai, H.-N.; Wang, H. Blockchain challenges and opportunities: A survey. *Int. J. Web Grid Serv.* **2016**, *14*, 1–25. [CrossRef]
11. Dua, S.; Du, X. *Data Mining and Machine Learning in Cybersecurity*; CRC Press of Taylor & Francis Group: Boca Raton, FL, USA, 2016.
12. Buczak, A.; Guven, E. A Survey of Data Mining and Machine Learning Methods for Cybersecurity Intrusion Detection. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 1153–1176. [CrossRef]
13. Paar, C.; Pezl, J. *Understanding Cryptography*; Springer: New York, NY, USA, 2011.
14. Pfleeger, C.P.; Pfleeger, S.L.; Margulies, J. *Security in Computing*, 5th ed.; Prentice Hall: Upper Saddle River, NJ, USA, 2015.
15. Rivest, R.; Shamir, A.; Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **1978**, *21*, 120–126. [CrossRef]
16. Shor, P. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *J. Soc. Ind. Appl. Math.* **1999**, *41*, 303–332. [CrossRef]
17. Alagic, G.; Alperin-Sheriff, J.; Apon, D.; Cooper, D.; Dang, Q.; Liu, Y.-K.; Miller, C.; Moody, D.; Peralta, R.; Perlner, R.; et al. *Status Report on the First Round of the NIST Post-Quantum Cryptography Standardization Process*; NIST Interagency/Internal Report (NISTIR), NIST: Gaithersburg, MD, USA, 2019. [CrossRef]
18. Tera, H. Introduction to Post-Quantum Cryptography in Scope of NIST's Post-Quantum Competition. Bachelor's Thesis, University of Tartu, Tartu, Estonia, 2019.
19. Nejatollahi, H.; Dutt, N.; Ray, S.; Regazzoni, F.; Banerjee, I.; Cammarota, R. Post-quantum Lattice-based Cryptography Implementations: A Survey. *ACM* **2018**, *51*, 1–41. [CrossRef]
20. Chen, S.; Shi, R.; Ren, Z.; Yan, J.; Shi, Y.; Zhang, J. A Blockchain-based Supply Chain Quality Management Framework. In Proceedings of the 14th International Conference on e-Business Engineering, Shanghai, China, 4–6 November 2017.
21. Banerjee, A. Blockchain Technology: Supply Chain Insights from ERP. *Adv. Comput.* **2018**, *111*, 69–98.
22. Zhang, Y.; Xu, X.; Liu, A.; Lu, Q.; Xu, L.; Tao, F. Blockchain-Based Trust Mechanism for IoT-Based Smart Manufacturing System. *IEEE Trans. Comput. Soc. Syst.* **2019**, *6*, 1386–1394. [CrossRef]
23. Bahga, A.; Madisetti, V. Blockchain Platform for Industrial Internet of Things. *J. Softw. Eng. Appl.* **2016**, *9*, 533–546. [CrossRef]

24. Francisco, K.; Swanson, D. The Supply Chain Has No Clothes: Technology Adoption of Blockchain for Supply Chain Transparency. *Logistics* **2018**, *2*, 2. [CrossRef]
25. Joseph, K. Engineering and Manufacturing on the Blockchain: A Systematic Review. *IEEE Eng. Manag. Rev.* **2020**, *48*, 31–47.
26. Barenji, V.; Li, Z.; Wang, W.; Huang, G.; Guerra-Zubiaga, D. Blockchain-based ubiquitous manufacturing: A secure and reliable cyber-physical system. *Int. J. Prod. Res.* **2020**, *58*, 2200–2221. [CrossRef]
27. Abeyratne, S.; Monfared, R. Blockchain Ready Manufacturing Supply Chain Using Distributed Ledger. *Int. J. Eng. Technol.* **2016**, *5*, 1–10.
28. Saberi, S.; Kouhizadeh, M.; Sarkis, J.; Shen, L. Blockchain technology and its relationships to sustainable supply chain management. *Int. J. Prod. Res.* **2018**, *57*, 2117–2135. [CrossRef]
29. Tijan, E.; Aksentijevic, S.; Ivanic, K.; Jardas, M. Blockchain Technology Implementation in Logistics. *Sustainability* **2019**, *11*, 1185. [CrossRef]
30. Holland, M.; Stjepandic, J.; Nigischer, C. Intellectual Property Protection of 3D Print Supply Chain with Blockchain Technology. In Proceedings of the 2018 IEEE International Conference on Engineering Technology and Innovation, Stuttgart, Germany, 17–20 June 2018.
31. Holland, M.; Nigischer, C.; Stjepandic, J. Copyright Protection in Additive Manufacturing with Blockchain Approach. *Transdiscipl. Eng. A Paradig. Shift* **2017**, *5*, 914–921. [CrossRef]
32. Westerkamp, M.; Victor, F.; Ktipper, A. Blockchain-based Supply Chain Traceability: Token Recipes model Manufacturing Processes. In Proceedings of the 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, 30 July–3 August 2018.
33. Guin, U.; DiMase, D. Counterfeit Integrated Circuits: Detection Avoidance, and the Challenges Ahead. *J. Electron. Test.* **2014**, *30*, 9–23. [CrossRef]
34. Delavar, M.; Mirzakuchaki, S.; Ameri, M.H.; Mohajeri, J. PUF based solution for secure communication in advanced metering infrastructure. *Int. J. Commun. Syst.* **2016**, *30*, e3195. [CrossRef]
35. Herder, C.; Yu, M.-D.; Koushanfar, F.; Devadas, S. Physical Unclonable Functions and Applications: A Tutorial. *Proc. IEEE* **2014**, *102*, 1126–1141. [CrossRef]
36. Jin, Y. Introduction to hardware security. *Electronics* **2015**, *4*, 763–784. [CrossRef]
37. Gao, Y.; Ranasinghe, D.C.; Al-Sarawi, S.F.; Kavehei, O.; Abbott, D. Emerging Physical Unclonable Functions with nanotechnologies. *IEEE Access* **2016**, *4*, 61–80. [CrossRef]
38. Holcomb, D.E.; Burleson, W.P.; Fu, K. Power-up SRAM state as an Identifying Fingerprint and Source of TRN. *IEEE Trans. Comp.* **2008**, *57*, 1–14.
39. Christensen, T.A.; Sheets, J.E. Implementing PUF Utilizing EDRAM Memory cell Capacitance Variation. U.S. Patent No. 8,300,450B2, 30 October 2012.
40. Prabhu, P.; Akel, A.; Grupp, L.M.; Yu, W.-K.S.; Suh, G.E.; Kan, E.; Swanson, S. Extracting Device Fingerprints from Flash Memory by Exploiting Physical Variations. In Proceedings of the 4th International Conference on Trust and Trustworthy Computing, Pittsburgh, PA, USA, 22–24 June 2011.
41. Plusquellic, J.; Swarup, B. Systems and Methods for Generating PUF's from Non-Volatile Cells. U.S. Patent 10,216,965, 26 February 2019.
42. Chen, A. Comprehensive Assessment of RRAM-based PUF for Hardware Security Applications. In Proceedings of the 2015 IEEE International Electron Devices Meeting (IEDM), Washington, DC, USA, 7–9 December 2015.
43. Vatajelu, E.I.; Di Natale, G.; Barbareschi, M.; Torres, L.; Indaco, M.; Prinetto, P. STT-MRAM-Based PUF Architecture exploiting MTJ Fabrication-Induced Variability. *ACM Trans.* **2015**, *13*, 1–21.
44. Becker, G.T.; Wild, A.; Güneysu, T. Security analysis of index-based syndrome coding for PUF-based key generation. In Proceedings of the 2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), Washington, DC, USA, 5–7 May 2015.
45. Rahman, M.T.; Rahman, F.; Forte, D.; Tehranipoor, M. An Aging-Resistant RO-PUF for Reliable Key Generation. *IEEE Trans. Emerg. Top. Comput.* **2016**, *4*, 335–348. [CrossRef]
46. Chen, T.I.B.; Willems, F.M.; Maes, R.; van Der Sluis, E.; Selimis, G. A Robust SRAM-PUF Key Generation Scheme Based on Polar Codes. In Proceedings of the GLOBECOM 2017—2017 IEEE Global Communications Conference, Singapore, 4–8 December 2017.

47. Delvaux, J.; Gu, D.; Schellekens, D.; Verbauwhede, I. Helper Data Algorithms for PUF-Based Key Generation: Overview and Analysis. *IEEE Trans. Comput. Des. Integr. Circuits Syst.* **2015**, *34*, 889–902. [CrossRef]
48. Taniguchi, M.; Shiozaki, M.; Kubo, H.; Fujino, T. A stable key generation from PUF responses with a Fuzzy Extractor for cryptographic authentications. In Proceedings of the 2013 IEEE 2nd Global Conference on Consumer Electronics (GCCE), Tokyo, Japan, 1–4 October 2013.
49. Kang, H.; Hori, Y.; Katashita, T.; Hagiwara, M.; Iwamura, K. Cryptographie key generation from PUF data using efficient fuzzy extractors. In Proceedings of the 16th International Conference on Advanced Communication Technology, Pyeongchang, Korea, 16–19 February 2014.
50. Cambou, B.; Orlowski, M. Design of PUFs with ReRAM and ternary states. In Proceedings of the Cyber and Information Security Research Conference, Oak Ridge, TN, USA, 5–7 April 2016.
51. Niranjanamurthy, M.; Nithya, B.N.; Jagannatha, S. Analysis of Blockchain technology: Pros, cons and SWOT. *Clust. Comput.* **2018**, *22*, 14743–14757. [CrossRef]
52. Zhen, Z.; Xie, S.; Dai, H.-N.; Chen, X.; Wang, H. Blockchain challenges and opportunities: A survey. *Int. J. Web Grid Serv.* **2018**, *14*, 352–375. [CrossRef]
53. Wu, J. Are blockchains immune to all malicious attacks? *Financ. Innov.* **2016**, *2*, 2. [CrossRef]
54. Suciu, G.; Nădrag, C.; Istrate, C.; Vulpe, A.; Ditu, M.-C.; Subea, O. Comparative analysis of distributed ledger technologies. In Proceedings of the 2018 Global Wireless Summit (GWS), Chiang Rai, Thailand, 25–28 November 2018; pp. 370–373.
55. Zhu, X.; Shi, J.; Huang, S.; Zhang, B. Consensus-oriented cloud manufacturing based on blockchain technology: An exploratory study. *Pervasive Mob. Comput.* **2020**, *62*, 101113. [CrossRef]
56. Herbert, J.; Lichfield, A. A Novel Method for Decentralised Peer-to-Peer Software License Validation Using Cryptocurrency Blockchain Technology. In Proceedings of the 38th Australasian Computer Science Conference, Sydney, Australia, 27–30 January 2015.
57. Harlev, M.; Yin, H.; Langenheldt, K.; Mukkamala, R.; Vatrapu, R. Breaking Bad: De-Anonymizing Entity Types on the Bitcoin Blockchain Using Supervised Machine Learning. In Proceedings of the 51st Hawaii International Conference on System Sciences, Waikoloa Village, HI, USA, 3–6 January 2018.
58. Tosh, D.; Shetty, S.; Liang, X.; Kamhoua, C.; Kwiat, K.; Njilla, L. Security Implications of Blockchain Cloud with Analysis of Block Withholding Attack. In Proceedings of the 17th International Symposium Cluster, Cloud and Grid Computing, Madrid, Spain, 14–17 May 2017.
59. Meng, W.; Tischhauser, E.; Wang, Q.; Wang, Y.; Han, J. When Intrusion Detection Meets Blockchain Technology: A Review. *IEEE Access* **2018**, *6*, 10179–10188. [CrossRef]
60. Zheng, Z.; Xie, S.; Dai, H.; Chen, X.; Wang, H. An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. In Proceedings of the IEEE 6th International Congress on Big Data, Honolulu, HI, USA, 25–30 June 2018.
61. Koblitz, N.; Menezes, A. A Riddle Wrapped in an Enigma. Available online: http://eprint.iacr.org/2015/1018 (accessed on 1 June 2020).
62. Kiktenko, E.; Pozhar, N.; Anufriev, M.; Trushechkin, A.; Yunusov, R.; Kurochkin, Y.; Lvovsky, A.; Fedorov, A. Quantum Secured Blockchains. *Quantum Sci. Technol.* **2018**, *3*, 35004. [CrossRef]
63. Semmouni, M.; Nitaj, A.; Belkasmi, M. Bitcoin Security with Post Quantum Cryptography. *Intell. Tutoring Syst.* **2019**, 281–288. Available online: https://hal-normandie-univ.archives-ouvertes.fr/hal-02320898 (accessed on 1 June 2020).
64. Kampanakisy, P.; Sikeridisz, D. Two Post-Quantum Signature Use-cases: Non-issues, Challenges and Potential Solutions. In Proceedings of the 7th ETSI/IQC Quantum Safe Cryptography Workshop, Seattle, WA, USA, 3 November 2019.
65. Campbell, R. Evaluation of Post-Quantum Distributed Ledger Cryptography. *J. Br. Blockchain Assoc.* **2019**, *2*, 1–8. [CrossRef]
66. Gaj, K. Toward Efficient and Fair Software/Hardware Codesign and Benchmarking of Candidates in Round 2 of the NIST PQC Standardization Process. *CryptArchi* **2019**, *2019*. Available online: https://bit.ly/37INTzN (accessed on 1 June 2020).
67. Andrzejzak, M. Lattice sieving acceleration in FPGAs. *CryptArchi* **2019**. Available online: https://labh-curien.univ-st-etienne.fr/cryptarchi/workshop19/abstracts/andrzejczak.pdf (accessed on 1 June 2020).

68. Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schwabe, P.; Seiler, G.; Stehlé, D. CRYSTALS-Dilithium Algorithm Specifications and Supporting Documentation. 2019. Available online: https://pq-crystals.org/dilithium (accessed on 1 June 2020).
69. Buchanan, B. Quantum Robust Hash-Based Signatures. 2018. Available online: https://medium.com/coinmonks/quantum-robust-hash-based-signatures-2c11d2739d38 (accessed on 1 June 2020).
70. Kampanakis, P.; Fluhrer, S. LMS vs XMSS: Comparison of two Hash-based Signature Standards. Available online: https://eprint.iacr.org/2017/349 (accessed on 1 June 2020).
71. Becker, G. *Merkle Signature Schemes, Merkle Trees*; Seminar Bochum University: Bochum, Germany, 2008; Available online: https://www.emsec.ruhr-uni-bochum.de/media/crypto/attachments/files/2011/04/becker_1.pdf (accessed on 1 June 2020).
72. Fouque, P.-A.; Hoffstein, J.; Kirchner, P.; Lyubashevsky, V.; Pornin, T.; Prest, T.; Ricosset, T.; Seiler, G.; Whyte, W.; Zhang, Z. Falcon: Fast-Fourier Lattice-Based Compact Signatures over NTRU. NIST PQC Project Round 2, Documentation. 2019. Available online: https://falcon-sign.info/falcon.pdf (accessed on 1 June 2020).
73. Bindel, N.; Akleylek, S.; Alkim, E.; Barreto, P.; Buchmann, J.; Eaton, E.; Gutoski, G.; Kramer, J.; Longa, P.; Polat, H.; et al. Lattice-Based Digital Signature Scheme qTESLA. NIST PQC Project Round 2, Documentation. 2019. Available online: https://eprint.iacr.org/2019/085.pdf (accessed on 1 June 2020).
74. Casanova, A.; Faugere, J.-C.; Macario-Rat, G.; Patarin, J.; Perret, L.; Ryckeghem, J. GeMSS: A Great Multivariate Short Signature. NIST PQC Project Round 2, Documentation. 2019. Available online: https://www-polsys.lip6.fr/Links/NIST/GeMSS.html (accessed on 1 June 2020).
75. Beullens, W.; Preneel, B.; Szepieniec, A.; Vercauteren, F. LUOV: Signature Scheme Proposal. NIST PQC Project Round 2, Documentation. 2019. Available online: https://limo.libis.be/primo-explore/fulldisplay?docid=LIRIAS2286367&context=L&vid=Lirias&search_scope=Lirias&tab=default_tab&lang=en_US&fromSitemap=1 (accessed on 1 June 2020).
76. Chen, M.-S.; Hulsing, A.; Rijneveld, J.; Samardjiska, S.; Schwabe, P. MQDSS Specifications. NIST PQC Project Round 2, Documentation. 2019. Available online: https://www.google.com.hk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjizuKS94rqAhV0yYsBHUFqBt8QFjABegQIAxAB&url=http%3A%2F%2Fmqdss.org%2Ffiles%2Fmqdss.pdf&usg=AOvVaw3bVOKOlRmeA39DcioZkcsh (accessed on 1 June 2020).
77. Ding, J.; Chen, M.-S.; Petzoldt, A.; Schmidt, D.; Yang, B.-Y. Rainbow. NIST PQC Project Round 2, Documentation. 2019.
78. Snir, M.; Gropp, W.; Otto, S.; Huss-Lederman, S.; Dongarra, J.; Walker, D. *MPI—The Complete Reference: The MPI Core*; MIT Press: Cambridge, MA, USA, 1998; Volume 1.
79. Nichols, B.; Buttlar, D.; Farrell, J.; Jackie, F. *Pthreads Programming: A POSIX Standard for Better Multiprocessing*; O'Reilly Media, Inc.: Sevastopol, MA, USA, 1996.
80. Graves, L.; Lubell, J.; Yampolskiy, M.; King, W. Characteristic Aspects of Additive Manufacturing Security from Security Awareness Perspectives. *IEEE Access J.* **2019**, *7*, 103833–103853. [CrossRef]
81. Audit of the DoD's Use of Additive Manufacturing for Sustainment Parts. Available online: https://media.defense.gov/2019/Oct/21/2002197659/-1/-1/1/DODIG-2020-003.PDF (accessed on 1 June 2020).
82. Ellis, D.; Schuster, F. Why additive manufacturing needs blockchain. *Supply Chain Quarterly, Quarter 1*, 22 February 2019.
83. Independent Security Evaluators, "Ethercombing: Finding Secrets in Popular Places". Available online: https://www.ise.io/casestudies/ethercombing/ (accessed on 23 April 2019).

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).