Crypto Agility Definitions for Space Systems

1st Jannik Mähn
Cryptographic Systems
OHB System AG
Bremen, Germany
jannik.maehn@ohb.de

2nd Matthias Müller
Cryptographic Systems
OHB System AG
Bremen, Germany
matthias.mueller@ohb.de

3rd Karin Zielinski
Cryptographic Systems
OHB System AG
Bremen, Germany
karin.zielinski@ohb.de

***Abstract*—This paper explores the concept of crypto agility with the aim to establish clear definitions and terminology for future discussions and implementations. For that, relevant threats are analyzed and existing literature is reviewed. The main contribution of this work is a clear set of definitions of crypto agility for the context of space systems using certifiable crypto systems implementing opinionated cryptographic protocols. In addition, a secure update initialization mechanism is introduced.**

***Index Terms*—Crypto Agility, Post-Quantum Cryptography, Reprogramming**

I. Introduction

With the ever growing threat posed by quantum computers to classical cryptographic systems, cryptography is undergoing a transition towards Post-Quantum Cryptography (PQC) schemes. However, as the security foundations of PQC are still evolving, even protocols such as ML-KEM [1] and ML-DSA [2], recently standardized by the National Institute of Standards and Technology (NIST), remain potentially vulnerable to unforeseen attacks. As a response, security agencies including the French National Cybersecurity Agency (ANSSI) [4], the German Federal Office for Information Security (BSI) [6], [7], and NIST [8] recommend the implementation of crypto agility in IT-security systems. Thereby, crypto agility refers to the ability of updating the security system in case that any novel threat to the system is discovered.

Despite its growing importance, the term ”crypto agility” lacks a single, universally accepted definition in the literature. Several interpretations and conceptual frameworks exist, such as [9]–[11], [13], [16]. The objective of this work is to define crypto agility and its relevant aspects in the context of space systems. This involves three steps: identifying threats and risks to crypto agility, conducting a detailed literature review of existing definitions, and defining crypto agility wrt. to space systems.

An initial and important distinction must be made between opinionated and negotiation-based protocols. Negotiation protocols allow the selection of cryptographic primitives and algorithms during runtime, offering flexibility in configuration. In contrast, opinionated protocols enforce a fixed set of primitives and algorithms that must be adhered to. Although negotiation-based protocols offer more run-time flexibility, as soon as new threats occur, they also need to be updated via crypto agility measures. Thus, for simplicity, this work focuses its discussion to the agility of opinionated protocols.

The paper is structured as follows: Section 2 outlines threats to crypto agility, including active and passive attacks, as well as system failures - such as those caused by environmental conditions or update errors. The literature review, in Section 3, draws on the works of Ott and Peikert [9], Alnahawi et al. [10], Näther et al. [11], and IETF RFC 7696 [12], each offering distinct perspectives on crypto agility. Section 4 examines their relevance to space systems, where physical access to cryptographic units in flight is not possible and updates must be performed and validated remotely. This requires authenticated, integrity-protected updates and mechanisms for attack prevention and rollback, detailed in sections 5 and 6. A minimal hardware architecture implementing crypto agility is also introduced there. Section 7 concludes the work.

II. Threats to Crypto Agility

In this section, the threats to the crypto agility mechanism are discussed. The main threats are active attacks, passive attacks and update failures. Active adversaries try to interfere with the security unit to create an advantage actively, for instance through jamming or fake updates. Passive adversaries on the other side operate as curious bystanders, that are trying to gain information, but are not actively interacting with the unit. Passive attacks also include misuse, such as the operation of the security unit by unauthorized and/or unqualified personnel that unintentionally introduces security flaws. The third class of threats are failures: failures that occur during the update process, and failures that occur due to random events of the environment, a general threat to space systems. Update failures can cause unusability of hardware components, or the incompatibility of the unit wrt. to its interfaces.

These threats lead to two important aspects of the crypto agility mechanism: a secure update initialization process on one side, and an authenticated, integrity protecting fallback mechanism on the other side. These two aspects are discussed in detail in sections 5 and 6.

III. Literature Review

In this section, the results of the literature review are discussed. However, only the most relevant aspects are covered, since there are several, very different references of crypto agility in the literature. For instance, the Transport Layer

Protocol (TLS) [13], the Secure Shell Connection Protocol (SSH) [14] and the Internet Key Exchange Protocol Version 2 (IKEv2) [15] are implemented with 'crypto agility' in mind. However, the meaning of 'crypto agility' in these works is limited to cipher negotiation protocols, as defined above. As discussed, when the security agencies demand crypto agility in critical systems, such as space systems, their intention is the possibility to make changes to the security unit through reprogramming of soft- and/ or hardware: Hence, in this work, we focus on the reprogramming aspect of crypto agility.

Näther et. al. [11], give a canonical definition of crypto agility: *Cryptographic Agility "is a theoretical or practical approach, objective, or property which provides capabilities for setting up, identifying, and modifying encryption methods and keying material in a flexible and efficient way while preserving business continuity."*

The definition of "Cryptographic Agility" will be used as the baseline in this paper. However, in the following, several refinements of this definition will be made. In all these refinements, we refer to the definition of crypto agility from above, i.e., the term hardware agility reduces the term agility to hardware components, but in all facets described in the definition.

The above definition grasps several crucial ideas. Firstly, agility may be a fixed mechanism, but it is also a more general approach to the design of the crypto unit. Secondly, crypto agility includes the identification of novel threats, such as attacks to a specific implementation, or that entire mathematical foundations of schemes are broken. Then, based on the specific threat, crypto agility is flexible enough to adapt to these individual threats according, e.g., through reprogramming of hardware, or by upgrading the software. Lastly, without business continuation, the notion of crypto agility is pointless.

Historically, crypto agility meant the ability to replace certain components of algorithms within an implementation. Ott et al. [RD09] ask this "minor implementation challenge to be recast as a major design challenge" to better comprehend the larger picture of agility challenges. In their work, the authors define among others the following modalities of crypto agility:

1) Implementation Agility: "Application interfaces and policy configuration frameworks facilitate migration across implementations."
2) Compliance Agility: "Cryptographic infrastructure can be reconfigured to address compliance requirements for varying international regulations and frameworks, or to minimize a trusted computing base."
3) Security Strength Agility: "Many PQC algorithms require different implementations for different security strengths. Algorithms that dynamically scale security strength based on configuration provide better agility."
4) Migration Agility: "The ability to move automatically from one scheme to another - including conversion. Requires better use of cryptographic metadata at the level of application data."
5) Platform Agility: "The ability to use assured cryptographic algorithms across different platform types."

Here, several novel points are made. Firstly, the terms implementation, compliance, security strength and migration agility can all be achieved by reprogramming of soft- and/ or hardware. However, it is important to distinguish the purpose of the reprogramming. In particular, the complexity of the update can be very different, depending on if only the parameters are changed to achieve a higher level of security. Or, if an entire protocol, including hardware design, needs to be exchanged.

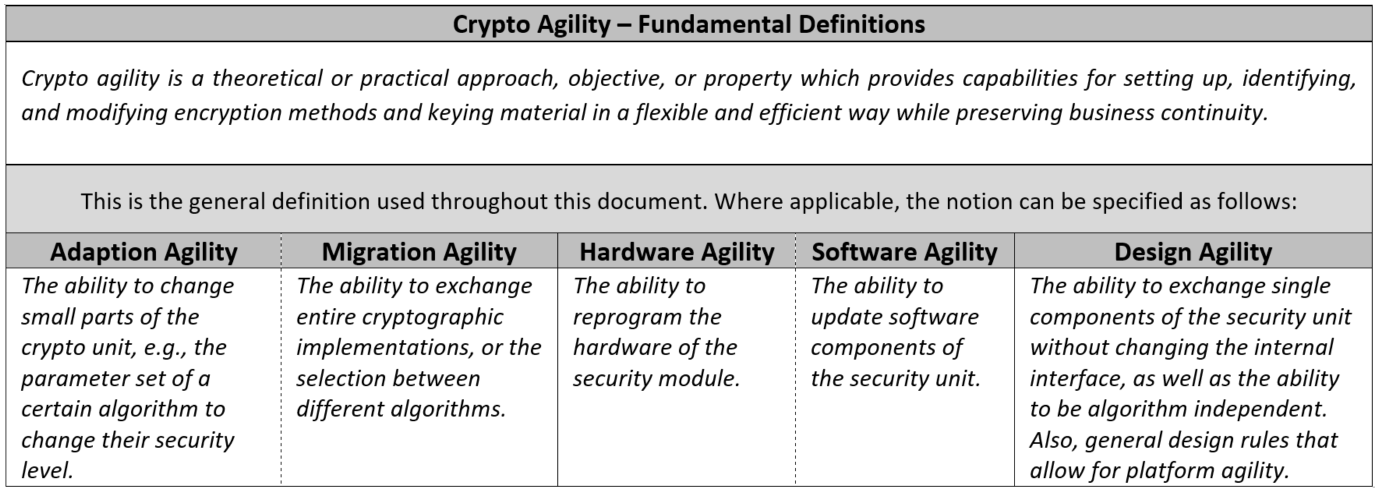
Furthermore, the need for modularity in cryptography is stressed through the definition of Platform Agility. So far, the point of view has been the threat of novel attacks breaking or reducing the security of cryptographic algorithms. However, not the entire crypto unit must be affected. Instead, fixing the threat can, in many cases, be fixed by exchanging/ updating single components such as the key derivation, the hash-function, or by updating the specific implementation, or by changing the parameter set of the protocols in use. In particular, this modularity of cryptographic system supports the reuse of components across different platforms.

Most facets of crypto agility will be achieved either through hardware reprogramming, or the upgrade of software components. Hardware agility is defined in [10], as "The ability of the hardware to support design agility through general design-agility, redesign/ repurposing of hardware and accelerator agility in mind." To add on this, we define software agility as the ability to update software components of the security unit, such as the object code.

Furthermore, the authors emphasize the importance of a fallback mechanism in case of any malfunctioning. A malfunctioning is detected by a collection of (online) tests with the purpose of validating the functioning of the security unit during operation and the correctness of hard- and software reprogramming. This includes the process of testing and validating the functionality of updated cryptographic mechanisms. This testing and validation of the updated implementation of protocols and algorithms is highly important to prevent misbehavior, or even failure, of the unit. These tests could for instance be combined with autonomous self-tests. Those are necessary for testing functionality after reboots.

Additionally, redundancy concepts that contain the necessary hardware for crypto agility, must be tailored to the specific mission, taking into account both reliability and cost. Furthermore, in certified or regulated projects, any update may require re-certification and official approval.

Lastly, we refer to the RFC 7696 [12] Best Current Practice-document "Guidelines for Cryptographic Algorithm Agility". Here, one aspect of crypto agility is of particular interest: Opportunistic Security, which describes a mechanism that allows the communication link to use algorithms that are considered weak in case the communicating parties do not have strong algorithms in common. Using out-of-date schemes or implementations is considered more secure than using no cryptographic algorithms at all. However, for each application,



Crypto Agility – Fundamental Definitions

Crypto agility is a theoretical or practical approach, objective, or property which provides capabilities for setting up, identifying, and modifying encryption methods and keying material in a flexible and efficient way while preserving business continuity.

This is the general definition used throughout this document. Where applicable, the notion can be specified as follows:

Adaption Agility	Migration Agility	Hardware Agility	Software Agility	Design Agility
The ability to change small parts of the crypto unit, e.g., the parameter set of a certain algorithm to change their security level.	*The ability to exchange entire cryptographic implementations, or the selection between different algorithms.*	*The ability to reprogram the hardware of the security module.*	*The ability to update software components of the security unit.*	*The ability to exchange single components of the security unit without changing the internal interface, as well as the ability to be algorithm independent. Also, general design rules that allow for platform agility.*

Fig. 1. Crypto Agility Definitions Overview for Space Systems

a careful analysis if opportunistic security is applicable must be carried out. In particular, it must be decided if the disabling of the service is preferred over the use of weak cryptography.

IV. Crypto Agility Definition for Space Systems

The literature review shows that there are many definitions of, and aspects to crypto agility. However, some of them are too general in the space context. Therefore, this section condenses the above notions to the needs for space systems.

Find an overview of the crypto agility definitions for space systems in Fig. 1. Those include the general crypto agility definition from [11], which combines all relevant aspects of crypto agility in one definition, also in the space context. However, as in the general case, refinements of this general definition must be made. The aim of that refinement is to provide notions for different special cases of crypto agility. This allows both researchers and developers to communicate more precisely.

There are three categories of refinements: one is concerned with the choice of algorithms, another deals with different forms of reprogramming, and lastly the design agility. The first category includes adaption agility (for instance, parameter strength agility, and updating key material or certificates) and migration agility (migrating from one algorithm to an entirely different one). This differentiation is very important, since adaption agility does not require the reprogramming of the entire algorithms. In particular, adaption agility can be achieved, depending on the design, simply by exchanging a configuration section in a non-volatile memory. This makes a great difference in the update: no software or hardware reprogramming is required, which minimizes risk, testing effort and time where the crypto unit is out of operation. Thus, even though both adaption and migration agility have the same aim - exchanging a possibly insecure algorithm by one that is considered secure - the implications are very different.

However, not all changes can be achieved through parameter change alone. Thus, this second category of notions also contains software and hardware agility. Software updates are already established in space systems to cope with necessary modifications or enhancements after launch. However, depending on the location of the crypto system in the communication chain and the certification/ approval level of the crypto system, software agility is only implemented if necessary. Many cryptographic systems are not implemented completely in software. The implementation of some parts in hardware allows to improve the performance of the cryptographic computations or are required for certification and approval of the crypto system. Depending on the FPGA technology different aspects need to be considered as outlined in Section 6.

The main motive of the third category, the notion design agility, is modularity. Designing the security system in a modular way is beneficial to agility in different ways. The above example, which can be achieved through a well-chosen hardware/ software co-design, can allow the developer to make smaller changes to the algorithm by updating the software, without changing the hardware implementation. Furthermore, making the modular components algorithm independent (by operating through fixed, implementation independent interfaces) should be considered for space systems. Hence, if the interfaces are preserved, the algorithm can easily be exchanged. Note that design agility does not describe a specific aspect of agility, it is more a general design rule that supports agility. Furthermore, design agility aims at designing components of the crypto unit in a possibly platform independent way, such that they can be reused/ connected across platforms.

V. Update Initialization Process

The above discussion shows that in the space system context, crypto agility in most cases refers to soft- and/or hardware reprogramming. However, fake-updates pose a severe threat to the security of the system. Thus, the update file received by the satellite must be authentic and its integrity must be intact. This demands a specific update initialization process, which is discussed in this section.

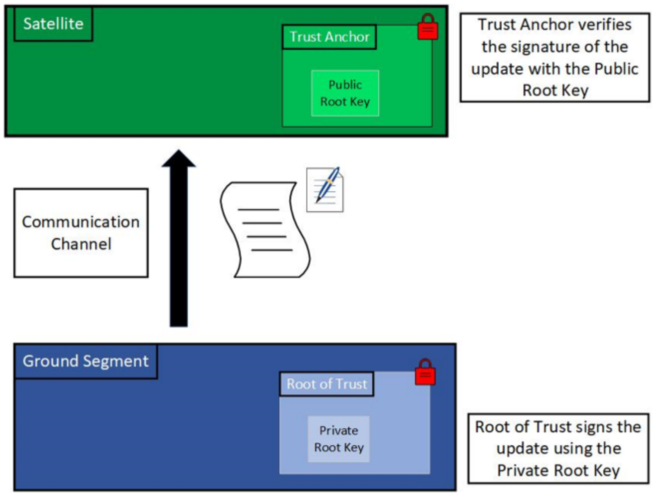

Satellite
Trust Anchor
Public Root Key
Trust Anchor verifies the signature of the update with the Public Root Key
Communication Channel
Ground Segment
Root of Trust
Private Root Key
Root of Trust signs the update using the Private Root Key

Fig. 2. Authentication Procedure for the Update Initialization

Before the process itself is defined, some general terms need to be defined.

1) Certification Authority: In this context, the Certification Authority (CA) is the ground unit that is communicating with other units, in particular with the satellite, and that is authorized to initiate all kinds of updates and upgrades of the on-board security unit.
2) Root Key: The Root Key (RK) is the public-private key pair of the signature scheme that is used to authenticate the CA. The private-part of the RK is stored in the RoT (see below) of the CA, whereas the public-part of the RK pair is stored in the Trust Anchor (see below) in the satellite. Note that the public part of a RK may be stored in different units.
3) Root of Trust: A Root of Trust (RoT) is a security module in the ground unit that comes with at least three functionalities: storing the private part of the root key such that it is protected from reading and manipulation, the ability to sign messages with that private Root Key, and to provide a anti-replay protection.
4) Trust Anchor: The Trust Anchor (TA) is a security module in the satellite that comes with at least the following three functionalities: storing the public part of the Root Key in an authenticity-protected manner, the ability to verify signatures using the public Root Key, and the ability to verify the anti-replay protection. The first functionality includes integrity and that the public key cannot be replaced by any other key.

Please note that the term Certification Authority combines entities that must usually be considered separately. On one side, it functions as the Ground Segment, and at the same time it holds the authenticity to initiate the update process. This is a simplification of the real-world setup, but suffices to illustrate the ideas.

Find an overview of the update initialization process in Fig. 2. In case of a partly successful attack, the satellite's Trust

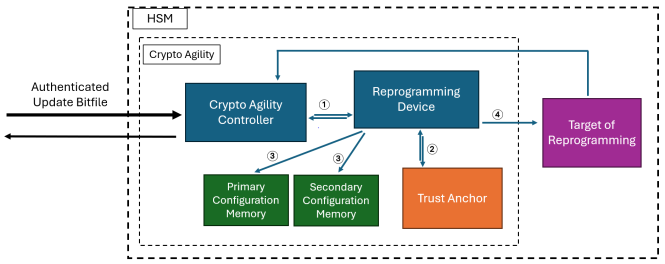

HSM
Crypto Agility
Authenticated Update Bitfile
Crypto Agility Controller
Reprogramming Device
Target of Reprogramming
Primary Configuration Memory
Secondary Configuration Memory
Trust Anchor
①
②
③
③
④

Fig. 3. Minimal Hardware Configuration for Crypto Agility

Anchor shall protect the recovery process. As a main point, the Trust Anchor ensures that only a legitimate originator can command recovery steps. Thus, the combination of a Trust Anchor and a Root-of-Trust protects the update process against misuse. This is one crucial step in securing the crypto agility mechanism, but it should further be supported by an appropriate security design. Since the goal of RoT and Trust Anchor is to authenticate the update code, the Root Key is an asymmetric signature key. In particular, the security agencies such as ANSSI [4] and the BSI [5], [7] recommend stateful hash-based signatures for this update authentication.

VI. Exemplary Hardware Configuration for Crypto Agility

In this section, we introduce an exemplary hardware configuration that is able to execute the update process in a failure-resistant way. Hence, a rollback mechanism is essential, to ensure secure and reliable on-the-fly updates of the security unit. In case of errors during reprogramming, the system must revert to the previously validated configuration. This mechanism is closely tied to the minimal hardware requirements for reprogramming.

A possible example is a dual-FPGA architecture, as depicted in Figure 3, consisting of:

- Target of Reprogramming (ToR):
  The application to be updated, such as an FPGA, microcontroller, firmware, or software.
- Reprogramming Device:
  A dedicated hardware unit (e.g., FPGA or microprocessor) that performs the update.
- Primary Non-Volatile Memory:
  Stores the currently running and validated bitstream.
- Secondary Non-Volatile Memory:
  Stores the newly uploaded update. After successful reprogramming, the roles of the memories may switch.
- Crypto Agility Controller
  A hardware unit (e.g., FPGA) that manages update commands and error handling via a finite state machine.

This hardware configuration executes the reprogramming by performing the following reconfiguration sequence: First, the update is loaded into the secondary memory upon command from mission control. Then, the update is verified using a Trust

Anchor to ensure authenticity and integrity. The reprogramming device installs the new bitstream or object code onto the ToR, and initial tests are executed by the reprogramming device. Finally, validation is confirmed via handshake between the ToR and mission control. This validation creates two possible cases:

- Validation Successful:
  The update is finalized and persisted. The memory roles are switched. Mission control must explicitly command this step.
- Incomplete Validation:
  If tests fail or the handshake is not completed within a predefined time, the rollback mechanism is triggered automatically by the reprogramming device.

This entire sequence is managed and monitored by the crypto agility controller. It includes a finite state machine, which autonomously operates the rollback mechanism, especially when the ToR is disconnected from mission control. Adjacent systems must be notified after both successful updates and rollbacks. This crypto agility mechanism ensures secure, reliable, and flexible reprogramming, with authentication handled by the Root-of-Trust and Trust Anchor.

This previously defined reprogramming architecture can, for instance, be realized using both SRAM-based and Flash-based FPGAs. When considering reprogramming of these FPGAs, it is important to distinguish between them, as each requires a different approach to updating and reliability. The realization differs due to the configuration memory characteristics of each FPGA type:

SRAM-Based FPGA Realization:
The reprogramming device loads the bitstream from either the primary or secondary non-volatile memory into the volatile configuration memory of the ToR. Rollback is performed by reloading the previous bitstream from the primary memory. Bitstream switching is fast and supports dynamic reconfiguration. Authenticity and integrity checks are performed before loading. Here, external memory access and secure boot logic are required. It is to some extend comparable to a software update due to the configuration from the non-volatile memory into the volatile FPGA configuration RAM after each power cycle.

Flash-Based FPGA Realization:
The reprogramming device writes the new bitstream to the secondary non-volatile memory, whereas the primary non-volatile memory is loaded into the FPGA. Rollback is realized by reverting to the previous flash image. Integrity and authenticity are verified before reprogramming. Reconfiguration is slower and less flexible but benefits from built-in secure boot and instant-on capability.

VII. Conclusion

This paper provides a structured overview of crypto agility, beginning with a threat analysis, including active and passive attacks as well as system failures. A comprehensive literature review is conducted, discussing different definitions and conceptual frameworks. These are then condensed in the specific context of space systems, with its unique prerequisites and requirements. The space-system specific definitions are, however, purposefully left general, such that they can be adapted to the many different possible requirements of different space missions. Furthermore, the mechanism for secure update initialization is defined to ensure integrity and authenticity in cryptographic updates. Lastly, a minimal hardware configuration that allows for failure-resistant updates is introduced. The overall objective of this work is to establish clear and consistent definitions and terminology for crypto agility and for future discussion, evaluation, and implementation for space systems in the context of post-quantum cryptography migration.

Acknowledgment

This research was carried out in the context of the ESA project “Agile Post-Quantum Space Data Link Security Protocol Hardware Module” (Activity No. 1000038710), whose support is gratefully acknowledged.

References

[1] National Institute of Standards and Technology (NIST), *FIPS 203: ML-KEM – Module-Lattice-Based Key Encapsulation Mechanism*, Final Publication, 2024.

[2] National Institute of Standards and Technology (NIST), *FIPS 204: ML-DSA – Module-Lattice-Based Digital Signature Algorithm*, Final Publication, 2024.

[3] National Institute of Standards and Technology (NIST), *FIPS 205: SLH-DSA – Stateless Hash-Based Digital Signature Algorithm*, Final Publication, 2024.

[4] ANSSI (French National Cybersecurity Agency), “ANSSI Views on the Post-Quantum Cryptography Transition (2023 Follow-Up),” *Position Paper*, Dec. 21, 2023.

[5] Federal Office for Information Security (BSI), “Cryptographic Mechanisms: Recommendations and Key Lengths (BSI-TR-02102-1),” Technical Guideline, Version 2025-01, Mar. 4, 2025.

[6] Federal Office for Information Security (BSI), “Quantum-safe cryptography – fundamentals, current developments and recommendations” Guideline, May 18, 2022.

[7] Federal Office for Information Security (BSI), “Migration to Post-Quantum Cryptography,” Technical Guideline/position paper, August, 2020.

[8] National Institute of Standards and Technology (NIST), “Status Report on the Fourth Round of the NIST Post-Quantum Cryptography Standardization Process,” NIST Interagency/Internal Report 8545, Mar. 11, 2025.

[9] David Ott, Christopher Peikert, et al., “Identifying Research Challenges in Post Quantum Cryptography Migration and Cryptographic Agility,” *CoRR*, vol. abs/1909.07353, 2019.

[10] Alnahawi, N., Schmitt, N., Wiesmaier, A., Heinemann, A., & Grasmeyer, T. (2023). On the state of crypto-agility. Cryptology ePrint Archive.

[11] Christian Näther, Daniel Herzinger, Jan-Philipp Steghöfer, Stefan-Lukas Gazdag, Eduard Hirsch, and Daniel Loebenberger, “Toward a Common Understanding of Cryptographic Agility – A Systematic Review,” *arXiv preprint*, arXiv:2411.08781, 2025.

[12] Russ Housley, “Guidelines for Cryptographic Algorithm Agility and Selecting Mandatory-to-Implement Algorithms,” *RFC: 7696*, RFC Editor, Nov. 2015.

[13] Eric Rescorla and Tim Dierks, “The Transport Layer Security (TLS) Protocol Version 1.2,” *RFC: 5246*, RFC Editor, Aug. 2008.

[14] Tatu Ylönen and Tim Wright, “The Secure Shell (SSH) Protocol Architecture,” *RFC: 4251*, RFC Editor, Jan. 2006.

[15] Pasi Eronen, Yoav Nir, Paul E. Hoffman, and Charlie Kaufman, “Internet Key Exchange Protocol Version 2 (IKEv2),” *RFC: 5996*, RFC Editor, Sep. 2010.

[16] Jason A. Donenfeld, “WireGuard: Next Generation Kernel Network Tunnel,” *RFC: 8998*, RFC Editor, Mar. 2021.