ETSI GR SAI 006 V1.1.1 (2022-03)

Securing Artificial Intelligence (SAI); The role of hardware in security of AI

Disclaimer

The present document has been produced and approved by the Secure AI (SAI) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference
DGR/SAI-006

Keywords
artificial intelligence, cybersecurity

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM®** and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Group Report (GR) has been produced by ETSI Industry Specification Group (ISG) Secure AI (SAI).

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document identifies the role of hardware, both specialized and general-purpose, in the security of AI. It addresses the mitigations available in hardware to prevent attacks (as identified in ETSI GR SAI 005 [i.9]) and also addresses the general requirements on hardware to support SAI (expanding from ETSI GR SAI 004 [i.8] and ETSI GR SAI 002 [i.7]). In addition, the present document reviews possible strategies for using AI for protection of hardware. The present document also provides a summary of academic and industrial experience in hardware security for AI. In addition, it addresses vulnerabilities and weaknesses introduced by hardware that can amplify attack vectors on AI.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] US NIST Glossary.

NOTE: Available at https://csrc.nist.gov/glossary.

[i.2] Recommendation ITU-T X.1252: "Baseline identity management terms and definitions".

NOTE: Available at https://www.itu.int/rec/T-REC-X.1252/en.

[i.3] Recommendation ITU-T X.1254: "Entity authentication assurance framework".

NOTE: Available at https://www.itu.int/rec/T-REC-X.1254/en.

[i.4] ISO/IEC 24760-1:2019: "IT Security and Privacy -- A framework for identity management -- Part 1: Terminology and concepts".

NOTE: Available at https://www.iso.org/standard/77582.html.

[i.5] ISO/IEC 24760-2:2015: "Information technology -- Security techniques -- A framework for identity management -- Part 2: Reference architecture and requirements".

NOTE: Available at https://www.iso.org/standard/57915.html.

[i.6] ISO/IEC 24760-3:2016: "Information technology -- Security techniques -- A framework for identity management -- Part 3: Practice".

NOTE: Available at https://www.iso.org/standard/57916.html.

[i.7] ETSI GR SAI 002: "Securing Artificial Intelligence (SAI); Data Supply Chain Security".

[i.8] ETSI GR SAI 004: "Securing Artificial Intelligence (SAI); Problem Statement".

[i.9] ETSI GR SAI 005: "Securing Artificial Intelligence (SAI); Mitigation Strategy Report".

[i.10] Florian Tramèr, Dan Boneh: "Slalom: Fast, Verifiable and private execution of neural networks in trusted hardware", Proc. ICLR 2019. February 2019.

NOTE: Available at https://arxiv.org/abs/1806.03287.

[i.11] Nick Hynes, Raymond Cheng, Dawn Song: "Efficient Deep Learning on Multi-Source Private Data", July 2018.

NOTE: Available at https://arxiv.org/abs/1807.06689.

[i.12] ETSI GS NFV-SEC 009: "Network Functions Virtualisation (NFV); NFV Security; Report on use cases and technical approaches for multi-layer host administration".

NOTE: Available at https://www.etsi.org/deliver/etsi_gs/NFV-SEC/001_099/009/01.01.01_60/gs_nfv-sec009v010101p.pdf.

[i.13] US NIST: "Cybersecurity White Paper: Hardware-Enabled Security for Server Platforms" (draft).

NOTE: Available at https://csrc.nist.gov/News/2021/hardware-enabled-security-draft-nistir-8320.

[i.14] US NIST SP800-155: "BIOS Integrity Measurement Guidelines (draft)".

NOTE: Available at https://csrc.nist.gov/CSRC/media/Publications/sp/800-155/draft/documents/draft-SP800-155_Dec2011.pdf.

[i.15] TCG Glossary.

NOTE: Available at https://trustedcomputinggroup.org/resource/tcg-glossary/.

[i.16] GlobalPlatform GPD-SPE-009: "TEE System Architecture".

NOTE: Available at https://globalplatform.org/wp-content/uploads/2017/01/GPD_TEE_SystemArch_v1.2_PublicRelease.pdf.

[i.17] GlobalPlatform GP-REQ-025: "Root of Trust Definitions and Requirements v1.1".

NOTE: Available at https://globalplatform.wpengine.com/wp-content/uploads/2018/07/GP_RoT_Definitions_and_Requirements_v1.1_PublicRelease-2018-06-28.pdf.

[i.18] IETF RFC 8392: "CBOR Web Token (CWT)".

NOTE: Available at https://tools.ietf.org/html/rfc8392.

[i.19] IETF RFC 7519: "JSON Web Token (JWT)".

NOTE: Available at https://tools.ietf.org/html/rfc7519.

[i.20] IETF draft-ietf-rats-architecture-14: "Remote Attestation Procedures Architecture".

NOTE: Available at https://datatracker.ietf.org/doc/draft-ietf-rats-architecture/.

[i.21] IETF draft-ietf-rats-eat-11: "The Entity Attestation Token (EAT)".

NOTE: Available at https://datatracker.ietf.org/doc/draft-ietf-rats-eat/.

[i.22] IETF draft-birkholz-rats-tuda-06: "Time-Based Uni-Directional Attestation".

NOTE: Available at https://datatracker.ietf.org/doc/draft-birkholz-rats-tuda/.

[i.23] IETF draft-ietf-rats-tpm-based-network-device-attest-11: "TPM-based Network Device Remote Integrity Verification".

NOTE: Available at https://datatracker.ietf.org/doc/draft-ietf-rats-tpm-based-network-device-attest/.

[i.24] IETF draft-ietf-rats-yang-tpm-charra-13: "A YANG Data Model for Challenge-Response-based Remote Attestation Procedures using TPMs".

NOTE: Available at https://datatracker.ietf.org/doc/draft-ietf-rats-yang-tpm-charra/.

[i.25] IETF draft-ietf-sacm-coswid-20: "Concise Software Identification Tags".

NOTE: Available at https://datatracker.ietf.org/doc/draft-ietf-sacm-coswid/.

[i.26] IETF draft-ietf-sacm-epcp-01: "Endpoint Posture Collection Profile".

NOTE: Available at https://datatracker.ietf.org/doc/draft-ietf-sacm-epcp/.

[i.27] IETF RFC 8248: "Security Automation and Continuous Monitoring (SACM) Requirements".

NOTE: Available at https://tools.ietf.org/html/rfc8248.

[i.28] IETF RFC 8412: "Software Inventory Message and Attributes (SWIMA) for PA-TNC".

NOTE: Available at https://tools.ietf.org/html/rfc8412.

[i.29] TCG: "Runtime Integrity Preservation for Mobile Devices".

NOTE: Available at https://trustedcomputinggroup.org/resource/tcg-runtime-integrity-preservation-in-mobile-devices/.

[i.30] TCG: "Trusted Platform Module 2.0 Library".

NOTE: Available at https://trustedcomputinggroup.org/resource/tpm-library-specification/.

[i.31] TCG: "Trusted Attestation Protocol (TAP) Information Model for TPM Families 1.2 / 2.0 and DICE Family 1.0".

NOTE: Available at https://trustedcomputinggroup.org/resource/tcg-tap-information-model/.

[i.32] IETF RFC 9019: "A Firmware Update Architecture for Internet of Things" (IETF Last Call).

NOTE: Available at https://datatracker.ietf.org/doc/draft-ietf-suit-architecture/.

[i.33] IETF RFC 9124: "A Manifest Information Model for Firmware Updates in Internet of Things (IoT) Devices".

NOTE: Available at https://datatracker.ietf.org/doc/draft-ietf-suit-information-model/

[i.34] IETF draft-ietf-suit-manifest-16: "A Concise Binary Object Representation (CBOR)-based Serialization Format for the Software Updates for Internet of Things (SUIT) Manifest".

NOTE: Available at https://datatracker.ietf.org/doc/draft-ietf-suit-manifest/.

[i.35] IETF draft-ietf-teep-architecture-15: "Trusted Execution Environment Provisioning (TEEP) Architecture".

NOTE: Available at https://datatracker.ietf.org/doc/draft-ietf-teep-architecture/.

[i.36] MITRE: "Hardware Assurance and Weakness Collaboration and Sharing (HAWCS)" Trust & Assurance Cyber Technologies Dept., Cyber Solutions Technical Center.

NOTE: Available at https://csrc.nist.gov/CSRC/media/Projects/cyber-supply-chain-risk-management/documents/SSCA/Fall_2019/WedPM2.2_Robert_Martin.pdf.

[i.37] MITRE: data definitions.

NOTE: Available at https://cwe.mitre.org/data/definitions/1194.html.

[i.38] Overview of MITRE CWE.

NOTE: Available at https://cwe.mitre.org/about/index.html.

[i.39] Wenye Liu at al.: "Two Sides of the Same Coin: Boons and Banes of Machine Learning in Hardware Security", EEE Journal on Emerging and Selected Topics in Circuits and Systems, VOL. 11, No. 2, June 2021.

[i.40] Y.-H. Chen, et al.: "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks" IEEE J. Solid-State Circuits, vol. 52, no. 1, pp. 127-138, January 2017.

[i.41] B. Moons and M. Verhelst: "An energy-efficient precision-scalable ConvNet processor in 40-nm CMOS", IEEE J. Solid-State Circuits, vol. 52, no. 4, April 2017.

[i.42] Y. Ma, Y. Cao, S. Vrudhula and J.-S. Seo: "Optimizing the convolution operation to accelerate deep neural networks on FPGA", IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 26, no. 7, Jul. 2018.

[i.43] S. Han et al.: "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding".

NOTE: Available at http://arxiv.org/abs/1510.00149.

[i.44] S. Han et al.: "EIE: Efficient inference engine on compressed deep neural network" in Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA), June 2016.

[i.45] P. Gysel, J. Pimentel, M. Motamedi and S. Ghiasi: "Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks", IEEE Trans. Neural Netw. Learn. Syst., vol. 29, no. 11, November 2018.

[i.46] B. Moons and M. Verhelst: "A 0.3-2.6 TOPS/W precision-scalable processor for real-time large-scale ConvNets", IEEE Symp. VLSI Circuits (VLSI-Circuits), June 2016.

[i.47] P. N. Whatmough et al.: "14.3 A 28 nm SoC with a 1.2 GHz 568 nJ/prediction sparse deep-neural-network engine with >0.1 timing error rate tolerance for IoT applications", IEEE Int. Solid-State Circuits Conf. (ISSCC), February 2017.

[i.48] DPU for Convolutional Neural Network V3.0, 2019.

[i.49] A. Tang et al.: "CLKSCREW: Exposing the perils of security-oblivious energy management", in Proc. 26th USENIX Security Symp., August 2017.

[i.50] G. Li et al.: "Understanding error propagation in deep learning neural network (DNN) accelerators and applications", Int. Conf. for High Perform. Comput., Netw., Storage, November 2017.

[i.51] A. Vakil et al.: "LASCA: Learning assisted side channel delay analysis for hardware Trojan detection", 21st Int. Symp. Qual. Electron. Design (ISQED), March 2020.

[i.52] M. Tehranipoor and F. Koushanfar: "A survey of hardware Trojan taxonomy and detection", IEEE Des. Test. Comput., vol. 27, no. 1, January 2010.

[i.53] R. Karri et al.: "Trustworthy hardware: Identifying and classifying hardware Trojans", Computer, vol. 43, no. 10, October 2010.

[i.54] M. Tehranipoor and C. Wang: "Introduction to Hardware Security and Trust", Springer, 2011.

[i.55] S. T. King et al.: "Designing and implementing malicious hardware," 1st Usenix Workshop Large-Scale Exploits Emergent Threats (LEET), 2008.

[i.56] K. Basu et al.: "CAD-base: An attack vector into the electronics supply chain", ACM Trans. Design Autom. Electron. Syst., vol. 24, no. 4, July 2019.

[i.57] C. Pilato, K. Basu, F. Regazzoni and R. Karri: "Black-hat high-level synthesis: Myth or reality?" IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 27, no. 4, April 2019.

[i.58] C. Dunbar and G. Qu: "Designing trusted embedded systems from finite state machines," ACM Trans. Embedded Comput. Syst., vol. 13, December 2014.

[i.59] X. Zhang and M. Tehranipoor: "Case study: Detecting hardware Trojans in third-party digital IP cores", in Proc. IEEE Int. Symp. Hardware Oriented Security and Trust, June 2011.

[i.60] T. Iwase et al.: "Detection technique for hardware Trojans using machine learning in frequency domain", IEEE 4th Global Conf. Consum. Electron. (GCCE), October 2015.

[i.61] Y. Liu et al.: "Silicon demonstration of hardware Trojan design and detection in wireless cryptographic ICs", IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 4.

[i.62] A. Kulkarni et al.: "SVM-based real-time hardware Trojan detection for many-core platform," in Proc. 17th Int. Symp. Qual. Electron. Design (ISQED), 2016.

[i.63] V. R. Carvalho and W. W. Cohen: "Single-pass online learning: Performance, voting schemes and online feature selection", in Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD), August 2006.

[i.64] Y. Jin, D. Maliuk and Y. Makris: "Post-deployment trust evaluation in wireless cryptographic ICs", in Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE, March 2012).

[i.65] J. Li, L. Ni et al.: "A novel hardware Trojan detection based on BP neural network", in Proc. 2nd IEEE Int. Conf. Comput. Commun. (ICCC), Chengdu, China, October 2016.

[i.66] K. Hasegawa et al.: "Hardware Trojans classification for gate-level netlists using multi-layer neural networks", in Proc. IEEE 23rd Int. Symp. Line Test. Robust Syst. Design (IOLTS), July 2017.

[i.67] A. Kulkarni et al.: "Adaptive real-time Trojan detection framework through machine learning", in Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST), May 2016.

[i.68] K. Hasegawa et al.: "Hardware Trojans classification for gate-level netlists based on machine learning", in Proc. IEEE 22nd Int. Symp. Line Test. Robust Syst. Design (IOLTS), July 2016.

[i.69] X. Chen et al.: "A general framework for hardware Trojan detection in digital circuits by statistical learning algorithms," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 36, no. 10.

[i.70] R. Elnaggar et al.: "Run-time hardware Trojan detection using performance counters", in Proc. IEEE Int. Test Conf. (ITC), October 2017.

[i.71] H. Salmani: "COTD: Reference-free hardware Trojan detection and recovery based on controllability and observability in gate-level netlist", IEEE Trans. Inf. Forensics Security, vol. 12, no. 2.

[i.72] B. Çakır and S. Malik: "Hardware Trojan detection for gate-level ICs using signal correlation based clustering", in Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE), March 2015.

[i.73] U. Guin, D. DiMase and M. Tehranipoor: "Counterfeit integrated circuits: Detection, avoidance, and the challenges ahead", J. Electron. Test., vol. 30, no. 1, February 2014.

[i.74] US Congress: "Ike Skelton national defense authorization act for fiscal year 2011", U.S. Government Printing Office, Tech. Rep., 2011.

[i.75] K. Mahmoodet al.: "Real-time automated counterfeit integrated circuit detection using X-ray microscopy", Appl. Opt., vol. 54, no. 13, 2015.

[i.76] N. Asadizanjani et al.: "Counterfeit electronics detection using image processing and machine learning", J. Phys., Conf. Ser., vol. 787.

[i.77] A. Ahmadi et al.: "A machine learning approach to fab-of-origin attestation", Proc. 35th Int. Conf. Comput.-Aided Design, November 2016.

[i.78] B. Ahmadi et al.: "Automated detection of counterfeit ICs using machine learning", Microelectron. Rel., vols. 88-90, September 2018.

[i.79] X. Zhang et al.: "Path-delay fingerprinting for identification of recovered ICs", Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT), October 2012.

[i.80] M. M. Alam et al.: "Recycled FPGA detection using exhaustive LUT path delay characterization", Proc. IEEE Int. Test Conf. (ITC), November 2016.

[i.81] A. Stern et al.: "EMFORCED: EM-based fingerprinting framework for counterfeit detection with demonstration on remarked and cloned ICs", Proc. IEEE Int. Test Conf. (ITC), October 2018.

[i.82] K. Huang et al.: "Parametric counterfeit IC detection via support vector machines", Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT), October 2012.

[i.83] H. Dogan et al.: "Aging analysis for recycled FPGA detection", Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT), October 2014.

[i.84] K. Huanget al.: "Recycled IC detection based on statistical methods", IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 34, no. 6, June 2015.

[i.86] S. Sharma and K. Chen: "Confidential machine learning on untrusted platforms: a survey", Cybersecurity 4, 30 (2021).

NOTE: Available at https://doi.org/10.1186/s42400-021-00092-8.

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

attack surface: total number of attack vectors that an attacker can use to manipulate a network or computer system or extract data

attack vector: method or way an attacker can gain unauthorized access to a network or computer system

authentication: verifying the identity of a user, process, or device, often as a prerequisite to allowing access to resources in an information system

NOTE: From US NIST Glossary [i.1].

authorization: access privileges granted to a user, program, or process or the act of granting those privileges

NOTE: From US NIST Glossary [i.1].

availability: ensuring timely and reliable access to and use of information

NOTE: From US NIST Glossary [i.1].

Client Application (CA): application running outside of the Trusted Execution Environment making use of the TEE Client API to access facilities provided by Trusted Applications inside the Trusted Execution Environment

NOTE: From Global Platform GPD-SPE-009 [i.16].

confidentiality: preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information

NOTE: From US NIST Glossary [i.1].

context: environment with defined boundary conditions in which entities exist and interact

NOTE: From Recommendation ITU-T X.1252 [i.2].

cybersecurity: prevention of damage to, protection of, and restoration of computers, electronic communications systems, electronic communications services, wire communication, and electronic communication, including information contained therein, to ensure its availability, integrity, authentication, confidentiality, and nonrepudiation

NOTE: From US NIST Glossary [i.1].

entity: something that has separate and distinct existence and that can be identified in a context

NOTE: From Recommendation ITU-T X.1252 [i.2].

Entity Authentication Assurance (EAA): degree of confidence reached in the authentication process that the entity is what it is, or is expected to be

NOTE 1: This definition is based on the 'authentication assurance' definition given in Recommendation ITU-T X.1252 [i.2].

NOTE 2: The confidence is based on the degree of confidence in the binding between the entity and the identity that is presented. Recommendation ITU-T X.1254 [i.3].

firmware image: binary that can contain the complete software of a device or a subset of it

NOTE 1: The firmware image can consist of multiple images, if the device contains more than one microcontroller. Often it is also a compressed archive that contains code, configuration data, and even the entire file system. The image can consist of a differential update for performance reasons.

NOTE 2: From IETF RFC 9019 [i.32].

Hardware-Mediated Execution Enclave (HMEE): area of process space and memory within a system environment within a computer host which delivers confidentiality and integrity of instructions and data associated with that enclave and which is protected from eavesdropping, replay, and alteration attacks as the programs within the enclave are executed

NOTE: Derived from ETSI GS NFV-SEC 009 [i.12].

identity: set of attributes related to an entity

NOTE: Within a particular context, an identity can have one or more identifiers to allow an entity to be uniquely recognized within that context. (from ISO/IEC 24760 [i.4] to [i.6]).

integrity: property that data or information have not been altered or destroyed in an unauthorized manner

NOTE: From US NIST Glossary [i.1].

manifest: meta-data about the firmware image which is protected against modification and provides information about the author

NOTE: Derived from IETF RFC 9019 [i.32].

mutual authentication: authentication of identities of entities which provides both entities with assurance of each other's identity

NOTE: Derived from Recommendation ITU-T X.1254 [i.3].

non-repudiation: ability to protect against denial by one of the entities involved in an action of having participated in all or part of the action

NOTE: Derived from Recommendation ITU-T X.1252 [i.2].

platform: any computing device (regardless of its architecture or operating system)

platform integrity: verifiable state of complete assurance that, under all conditions, a computing system has a correct and reliable operating system, logically complete hardware, firmware, and software, and system isolation and protection mechanisms (e.g. memory access control, process isolation, data integrity, etc.)

Relying Party (RP): actor that relies on an identity assertion or claim

NOTE: Derived from Recommendation ITU-T X.1254 [i.3].

Rich Execution Environment (REE): execution environment comprising at least one device OS or Rich OS and all other components of the device (SoCs, other discrete components, firmware, and software) which execute, host, and support the Rich OS (excluding any TEEs and SEs included in the device)

NOTE: WARNING: In a previous version of Global Platform GPD-SPE-009 [i.16] the REE was considered to be everything outside of the TEE under consideration. In the new definition other entities are acknowledged. Contrast *Trusted Execution Environment*. Global Platform GPD-SPE-009 [i.16].

risk: negative system impact considering:

1) the probability that a particular threat-source will exercise (accidentally trigger or intentionally exploit) a particular information system vulnerability; and

2) the resulting impact if this would occur

Secure Element (SE): tamper-resistant secure hardware component which is used in a device to provide the security, confidentiality, and multiple application environment required to support various business models

NOTE: Can exist in any form factor, such as embedded or integrated SE, SIM/UICC, smart card, smart microSD, etc. Derived from Global Platform GPD-SPE-009 [i.16].

security goals: integrity, availability, confidentiality, accountability, and assurance

security vulnerability: flaw or weakness in an information system, system security procedures, design, implementation, or internal controls that could be exercised (accidentally triggered or intentionally exploited by a threat) and result in a security breach or a violation of the system's security policy

NOTE: A hardware vulnerability is an exploitable weakness in a computer system that enables attack through remote or physical access to system hardware

Tensor Processing Unit (TPU): Application-Specific Integrated Circuits (ASICs) used to accelerate machine learning workloads

threat: potential for a threat-source to exercise (accidentally trigger or intentionally exploit) a specific vulnerability

trust framework: set of requirements and enforcement mechanisms for parties exchanging identity information (from Recommendation ITU-T X.1254 [i.3])

Trust Provisioning Authority (TPA): distribution subsystem for trust anchors and authorization policies to devices and various stakeholders

NOTE: The TPA can also delegate rights to stakeholders. Typically, the Original Equipment Manufacturer (OEM) or Original Design Manufacturer (ODM) will act as a TPA, however complex supply chains can require a different design. In some cases, the TPA can decide to remain in full control over the firmware update process of their products. IETF RFC 9019 [i.32].

Trusted Application (TA): application running inside the Trusted Execution Environment (TEE) that provides security related functionality to Client Applications outside of the TEE or to other Trusted Applications inside the TEE

NOTE: Derived from Global Platform GPD-SPE-009 [i.16].

Trusted Execution Environment (TEE): execution environment that runs alongside but isolated from an REE

NOTE: A TEE has security capabilities and meets certain security-related requirements: It protects TEE assets from general software attacks, defines rigid safeguards as to data and functions that a program can access, and resists a set of defined threats. There are multiple technologies that can be used to implement a TEE, and the level of security achieved varies accordingly. Contrast *Rich Execution Environment*. Global Platform GPD-SPE-009 [i.16].

trusted platform: platform that has verifiable assurance of the integrity of the underlying system configuration, including all hardware, firmware, and software

NOTE: The Platform Integrity can be verified prior to the instantiation of application services (e.g. virtual functions). TCG has recently published TCG Runtime Integrity Preservation in Mobile Devices (RIP) to address the continued assurance of Trusted Platform integrity for highly available system.

Trusted Third Party (TTP): authority or its agent, trusted by other actors with respect to specified activities (e.g. security-related activities)

NOTE: A trusted third party is trusted by an entity and/or a verifier for the purposes of authentication. (from Recommendation ITU-T X.1254 [i.3]).

trustworthiness: attribute based on assurance that a person or system will behave predictably, even under stress

NOTE: The key properties of trustworthiness include: Trustworthiness is based on experience and/or evidence; Trustworthiness is based on fundamental properties (such as identity and integrity); Trustworthiness is easily lost and difficult to regain.

verifier: actor that corroborates identity information

NOTE: The verifier can participate in multiple phases of the Entity Authentication Assurance Framework (EAAF) and can perform credential verification and/or identity information verification. (from Recommendation ITU-T X.1254 [i.3]).

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
BP	Back-Propagation
CA	Client Application
CBOR	Concise Binary Object Representation
CoT	Chain of Trust
COTD	Controllability and Observability for hardware Trojan Detection
CPU	Central Processing Unit
CVE	Common Vulnerabilities and Exposures
CWE	Common Weakness Enumeration
DNN	Distributed Neural Network
DP	Data Processing
DVFS	Dynamic Voltage Frequency Scaling
EAAF	Entity Authentication Assurance Framework
EDA	Exploratory Data Analysis
EFR	Early Failure Rate
EM	Electro-Magnetic
FIDO	Fast IDentity Online
FPGA	Field-Programmable Gate Array
GP	Global Platform
GPU	Graphics Processing Unit
HMEE	Hardware-Mediated Execution Enclave
HS	Half-Space
HT	Hardware Trojan
HW	Hardware
IC	Integrated Circuit
ICCAD	International Conference on Computer-Aided Design
IoT	Internet of Things
IT	Information Technology
LBP	Local Binary Patterns
LR	Logistic Regression
MAC	Message Authentication Code
ML	Machine Learning
NFV	Network Functions Virtualisation
OC-SVM	One-Class SVM
OS	Operating System
PA	Posture Attribute
PA-TNC	Posture Attribute - Trusted Network Connect
PCA	Principle Component Analysis
pRoT	primary RoT

RATS	Remote Attestation
RBF	Radial Basis Function Network
REE	Regular Execution Environment
RG	Research Group
RoT	Root of Trust
SACM	Security Automation and Continuous Monitoring
SDO	Standard Development Organisation
SEIM	Security Event and Incident Management
SIM	Subscriber Identity Module
SoC	System on a Chip
SRAM	Static Random-Access Memory
sRoT	secondary RoT
SUIT	Software Updates for IoT
SVM	Support Vector Machine
SWIMA	SACM Software Inventory Message and Attributes
TA	Trusted Application
TCG	Trusted Computing Group
TEE	Trusted Execution Environment
TEEP	TEE Provisioning
TNC	Trusted Network Connect
TPM	Trusted Processing Module
TPU	Tensor Processing Unit
TTP	Trusted Third Party
WG	Working Group

4 General purpose secure hardware

4.1 Overview

[i.86] considers threat models, security assumptions, design principles, and associated trade-offs (e.g. between data utility, cost, and confidentiality) for use cases where data owners depend on various untrusted hardware and OS platforms (e.g. public clouds, edges, and machine learning service providers) for scalable processing or collaborative learning. In this paper, authors use Cloud Services as a representation of possible untrusted ML platforms and discuss Hardware-Assisted approaches to ML hardware platform integrity, approaches already discussed in clause 4 of the present document.

Figure 1 describes possible use cases for attacking and defending AI systems where AI hardware and platform can be used for mitigation of possible attacks as well as for the advancement of attacks.

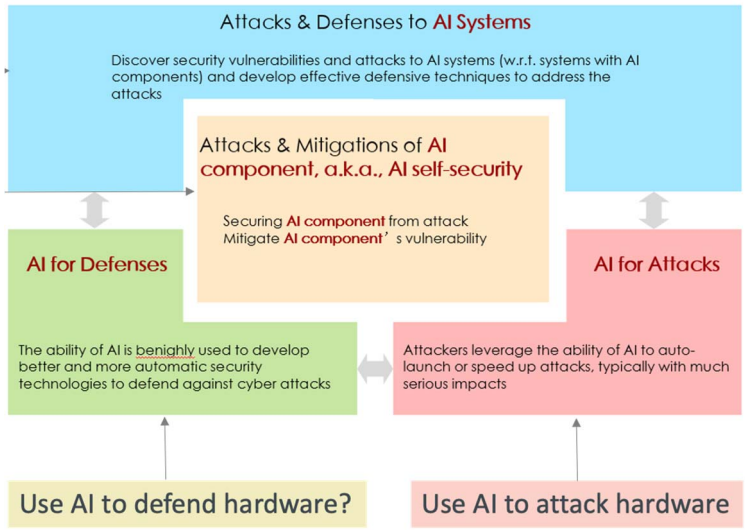

Figure 1: Possible use cases for attacking and defending of AI systems

4.2 Hardware-Mediated Execution Enclave

4.2.1 Introduction

ETSI GS NFV-SEC 009 [i.12], clause 6.16 defines Hardware-Mediated Execution Enclave (HMEE) as an area of process space and memory within a system environment within a computer host which delivers confidentiality and integrity of instructions and data associated with that enclave. HMEE enclave is protected from eavesdropping, replay and alteration attacks as the programs within the enclave are executed.

ETSI GS NFV-SEC 009 [i.12] considers HMEE enclave capable of executing processes, and executable code can be loaded into it. Various capabilities can be provided by such an enclave, but at minimum, the following capabilities are expected by ETSI GS NFV-SEC 009 [i.12]:

- The ability for executable code to be loaded into the enclave.
- The ability for the host to attest to the integrity of the executable code prior to execution.
- The ability to load data into the enclave.

4.2.2 Trusted Execution Environment

4.2.2.1 General

TEE is specified by GlobalPlatform and is a proprietary manifestation of Hardware-Mediated Execution Enclave (HMEE), ETSI GS NFV-SEC 009 [i.12].

4.2.2.2 TEE conceptual goals. Hardware dependency

The TEE concept implies the availability of a safe bastion offering hardware-assisted confidentiality and integrity guarantees to process and data. These guarantees can be (partially) provided even when execution takes place on hostile and exposed IT environments with a malicious kernel or under the control of a malicious operator with root access on the machine. The TEE concept is related to process memory page access (read and write) security management. The definition of a TEE covers the following set of security functions:

- Isolation: Memory partitioning and access restriction.
- Code confidentiality: TEE-located software (process memory pages) cannot be accessed when the code is running. Provisions are taken to ensure that the same code cannot be analysed outside the TEE (before it is loaded). This is achieved by encryption.
- Data confidentiality: TEE-located data (memory pages) cannot be accessed by any means except than by software located at the same TEE.
- Code integrity: TEE-located software (memory pages) cannot be accessed. No software change can be operated. A code change outside will be detected by the remote attestation.
- Data integrity: TEE-located data will only be accessed and eventually modified by the same TEE-residing software.
- Remote attestation of Trusted Computing Base (the software that integrates a TEE): The integrity and authenticity checks are done before the software is launched inside the TEE.
- Secure provisioning of Hardware TEE (i.e. the genuine TEE-enabled check).
- Secure data sealing-storage: The TEE establish a secure channel with externally located sealed storage.

These functions can be implemented on classical architectures at the kernel level or, with TEE, more securely by the processor directly, through processor ad-hoc design and microcode changes and complementary instructions. This derives into a dependency on the processor-specific architecture. Each processor vendor brings their own, proprietary TEE architectural design, though looking at the same main security goals. TEE designs and implementations are mostly not compatible. As a consequence, there is no commonly accepted unique TEE definition. Execution within a TEE does not imply any kind of vulnerability remediation, as the TEE is focused on protecting the running image and processed data against tampering. Ideally, only vulnerability-free checked processes should be executed inside TEEs.

The TEE concept also embeds hardware-based unique identity and random number generation. Remote attestation of the software inside the TEE (crypto-secured signature verification) as well as crypto-secured proofing that the processor is a real TEE-enabled processor can be offered according to the processor type. Associated with the remote attestation is the establishment of crypto-secured storage outside and secure transmission inside the TEE, referred to as data sealing. Data sealing logic is simple: decryption is only made inside the TEE, where the data are fully confidentiality and integrity protected. Outside the TEE, data are encrypted (which confers the same confidentiality and integrity guarantees) but encrypted data (until homomorphic processors emerge) cannot be processed in that state.

Regarding software integrity, a TEE is capable to generate crypto proven integrity verifications as offered by TPM. Both integrity services differ in the fact that TPMs represent a separate chipset from the processor, while TEEs are embedded in the processor. On the other hand, TPM specifications are part of standards [i.23], [i.24], [i.30] and [i.31], and no functional deviations are expected from one vendor to another, while TEEs are vendor-specific.

4.2.2.3 Securing AI through TEEs

With consideration to the function list enumerated in clause 4.2.2.2, several challenges arise in the design of ad-hoc TEE-secured AI-ML implementations. Recent academic works have reported advancements in this field.

SLALOM (2019) [i.10], from Stanford University, delivers a fast, verifiable, and private execution of neural networks in trusted hardware, leveraging a commercial TEE. SLALOM [i.10] splits the execution between a GPU and the TEE while delivering security assurance on the GPU operation correctness using Freivalds's algorithm. Outsourcing linear process from the TEE to the GPU is aimed at boosting performance, in a scheme that can be applied to any faster co-processor. Full TEE-embedded inference was the bottom line of this research, deemed as not satisfactory on the performance aspect.

MYELIN (2018) [i.11], from Berkeley University, delivers efficient deep learning on multi-source private data, leveraging differential privacy on commercial TEEs. MYELIN [i.11] shows similar performance (or neglectable slow down) when applying DP-protected ML. To do so, their implementation goes through the compilation of a static library embedding the core minimal routines. The static library is then fully run in the TEE, which removes any costly context switch from the TEE mode to the normal execution mode.

Specialized hardware accelerators (TPUs) are also viewed as necessary for highly demanding (fast) decision taking. That is a grey area, with no existing TEE embodiment for specialized hardware to the best of our knowledge. In addition, leveraging TEE data sealing capability looks like another path to consider for further improvements.

Securing AI by means of TEEs is a nascent discipline, while currently at its early stage, and recently reports encouraging results. Performance penalties are well-balanced by the high-security gain. A pending question is whether there will be a market shift of demanding AI-ML towards specialized hardware or, conversely, the use of distributed models on commodity hardware will prevail. One could envision specialized hardware design directly supporting the same security guarantees as those provided by TEEs and becoming the default operational mode without any setup complication. This technical assumption can be taken as specialized hardware design will be carried out from scratch without ascending compatibility issues to deal with so that adding a TEE provision (at least from the technical point of view) will be straightforward.

4.3 Root of Trust (RoT)

Because Roots of Trust (RoTs) are inherently trusted, they are secure by design. As such, many RoTs are implemented in hardware so that malware cannot tamper with the functions they provide. RoTs provide a firm foundation upon which to build security and trust.

US NIST in its Cybersecurity White Paper [i.13] states that all security controls need to have a Root of Trust (RoT) - a starting point that is implicitly trusted. Hardware-based controls can provide an immutable foundation for establishing platform integrity. Combining these functions with a means of producing verifiable evidence that these integrity controls are in place and have been executed successfully is the basis of creating a trusted platform. Minimizing the footprint of this RoT translates to reducing the number of modules or technologies that are expected to be implicitly trusted. This substantially reduces the attack surface.

US NIST White Paper [i.13] also affirms that platforms which secure their underlying firmware and configuration provide the opportunity for trust to be extended higher in the software stack. Verified platform firmware can, in turn, verify the Operating System (OS) boot loader, which can then verify other software components all the way up to the OS itself and the hypervisor or container runtime layers. The transitive trust described in [i.13] is consistent with the concept of the chain of trust (CoT) -a method where each software module in a system boot process is required to measure the next module before transitioning control.

In addition, [i.13] states that rooting platform integrity and trust in hardware security controls can strengthen and complement the extension of the CoT into the dynamic software category. There, the CoT can be extended even further to include data and workload protection. Hardware-based protections through CoT technology mechanisms can form a layered security strategy to protect data and workloads as they move to multi-tenant environments in a cloud data centre or edge computing facility.

Also, per [i.13], there are other hardware platform security technologies that can protect data at rest, in transit, and in use by providing hardware-accelerated disk encryption or encryption-based memory isolation. By using hardware to perform these tasks, the attack surface is mitigated, preventing direct access or modification of the required firmware. Isolating these encryption mechanisms to specific hardware can allow performance to be addressed and enhanced separately from other system processes.

US NIST SP800-155 [i.14] states that Roots of Trust (RoTs) are at the foundation of any BIOS integrity assurance. [i.14] lists software, hardware (or hybrid) platforms as *RoTs* components. [i.14] also states that while the trustworthiness of software agents may leverage the trustworthiness of one or more RoTs, it is dependant on the their trustworthiness and associated attack surfaces.

TCG Glossary [i.15] defines various Roots of Trust.

5 Specialized AI processing hardware

5.1 Neural processors and neural networks

5.1.1 Secure Hardware Accelerators

[i.39] states that neither cloud computing nor embedded microcontroller can fulfil the response time and throughput requirements simultaneously for latency and performance-critical applications and applications that require the inference to be made on the endpoints where data are generated and processed locally without a network connection (e.g. self-driving car, missile guidance system). Note that such endpoints can also be resource-constrained.

[i.39] also lists specialized AI processing hardware and dedicated hardware accelerators for low-latency, high-throughput, and energy-efficient DNN processing have emerged as demonstrated in [i.40], [i.41] and [i.42]. Customized dataflow, model compression, and Dynamic Voltage Frequency Scaling (DVFS) are the major and common optimization techniques used in many DNN accelerators. To mitigate the memory access bottleneck and maximize the reuse of on-chip data the specialized dataflow architecture emerged as a solution [i.40]. Model compression and data quantization are effective methods to reduce the number of MAC calculations, see [i.43] and [i.44], and the computational complexity of each arithmetic operation as [i.45] shows. Voltage scaling has been widely implemented to reduce dynamic power consumption. It exploits the excess positive slack of non-critical paths and the guard band preserved at design time to allow the DNN to operate at a lower than rated supply voltage whenever possible without compromising the throughput and accuracy, see [i.46] and [i.47]. The throughput is usually boosted by an array of parallel Processing Engines (PE) running at a higher clock frequency, refer to [i.48]. In addition, per [i.50], heavy usage of on-chip SRAM can also create additional reliability problems, and the performance degradation due to premature aging of hardware components can be an exploitable security vulnerability. In addition, the power management unit and separate clock and power domains for DVFS can also be exploited for a fault injection attack, refer to [i.49].

6 Mitigations available in hardware to prevent attacks

6.1 Protection of model hyperparameters and parameters

ETSI GR SAI 005 [i.9] reflects that protecting model hyperparameters and parameters is a traditional data protection requirement. Such protection can be achieved by adapting conventional data protection techniques. Data protection can be defined as the process of maintaining the confidentiality, integrity, and availability of an organization's data in a manner consistent with the organization's risk strategy.

In addition, ETSI GR SAI 005 [i.9] discusses two approaches to AI security:

- The use of secure hardware as one of the main approaches against side-channel attacks and unauthorized memory access, and
- Keeping model parameters confidentiality-protected to protect model parameters and hyperparameters from unauthorized access.

7 General requirements on hardware to support SAI

7.1 Expanding from ETSI GR SAI 002

The scope of ETSI GR SAI 002 [i.7] states that compromising the integrity of data has been demonstrated to be a viable attack vector against an AI system. The integrity of data can be compromised by various methods. One of them could be the use of compromised hardware for storage or processing of both raw data, information, and feedback from other AI systems and humans in the AI control loop.

ETSI GR SAI 002 [i.7] also considers in clause 6.3.2, Cryptographic mechanisms, that the standard way for ensuring the integrity of data is the application of cryptographic hash functions (and hash trees) to the data and storage of the resulting hash values while the hash values are signed using a digital signature algorithm. Arguably, compromised hardware will not only make such integrity protection also compromised but will enable selective leakage of AI data.

7.2 Expanding from ETSI GR SAI 004

ETSI GR SAI 004 [i.8], clause 4.3.8, Deployment and Inference, discusses various hardware-related issues, including TEE [i.16], back-door attacks that can compromise the confidentiality of the training set and can result in a denial of service attacks.

This clause also states that hardware deployments are attractive due to high levels of performance and are being used for both, the deployment of machine learning systems and by attackers wishing to exploit vulnerabilities.

8 Hardware vulnerabilities and common weaknesses in AI systems

8.1 Features of hardware-specific vulnerabilities and how to avoid them

The following list attempts to summarize features of hardware-specific vulnerabilities [i.36]:

- Mitigation - Fundamentally, it is assumed that software can be patched relatively easy, this might not be the case in hardware and hardware mitigations are often only partial.

- Effort - hardware attack surfaces are available to an attacker (e.g. power analysis). However, the effort required to use such a technique can be prohibitive. Thus, quantitative estimation of the attack effort is essential.
- Patchability - Vulnerable hardware (especially in physical systems) can be difficult to access to apply a mitigation.
- Hardware has an overlapping but different set of threats from software (e.g. remote exploitation, reverse engineering, counterfeiting). These threats are industry-specific and need to be addressed to be able to estimate a vulnerability score.
- Hardware systems are often consisting of deep layered systems most of which is inscrutable to any one party.

12 top categories, [i.37], have been itemized as of March 2021 and each item contains sub items.

The MITRE Common Vulnerabilities and Exposures (CVE®) Program mission goal is to identify, define, and catalogue publicly disclosed cybersecurity vulnerabilities. There is one CVE Record for each vulnerability in the catalogue. The vulnerabilities are discovered then assigned and published by organizations from around the world that have partnered with the CVE Program. There were 150 723 CVE records as of 22 March 2021.

To avoid hardware-related vulnerability in the design phase, the present clause also provides a list of common weaknesses that have been found and published as a catalogue called HW CWE (Common Weakness Enumeration). CWE is a catalogue maintained by MITRE corporation which is also maintaining CVE. An overview of MITRE CWE is available in [i.38].

In further detail, Common Weakness Enumeration (CWE™) is a community-developed list of common software and hardware weakness types that have security ramifications. "Weaknesses" are flaws, faults, bugs, or other errors in software or hardware implementation, code, design, or architecture that if left unaddressed could result in systems, networks, or hardware being vulnerable to attack.

CWE helps developers and security practitioners to:

- Describe and discuss software and hardware weaknesses in a common language.
- Check for weaknesses in existing software and hardware products.
- Evaluate coverage of tools targeting these weaknesses.
- Leverage a common baseline standard for weakness identification, mitigation, and prevention efforts.
- Prevent software and hardware vulnerabilities prior to deployment.

Note that the CWE list is still strongly biased toward software weaknesses - the original focus of the MITRE Common Vulnerabilities and Exposures (CVE) Program.

These lists are relevant to hardware security because the only practical defence against many of the most serious software weaknesses (e.g. out-of-bounds memory read or write, null pointer dereference) is based on hardware design choices. MITRE CVE lists are available from these sources:

- https://cwe.mitre.org/top25/archive/2019/2019_cwe_top25.html
- https://cwe.mitre.org/top25/archive/2020/2020_cwe_top25.html

MITRE has by now produced two annual reports of CWE Top 25 Most Dangerous Software Weaknesses. However, MITRE has not yet produced a report on Top 25 Most Dangerous Hardware Weaknesses.

MITRE CWE View: Hardware Design (View ID 1194) is accessible at https://cwe.mitre.org/data/definitions/1194.html

This view organizes weaknesses around concepts that are frequently used or encountered in hardware design. Accordingly, this view can align closely with the perspectives of designers, manufacturers, educators, and assessment vendors. It provides a variety of categories that are intended to simplify navigation, browsing, and mapping.

The top-level categories in this view represent commonly understood areas/terms within hardware design and are meant to aid the user in identifying potential related weaknesses. Note that it is possible for the same weakness to exist within multiple different categories.

The twelve categories of hardware design weaknesses that are based on 95 CWEs out of the total of 918 CWE weaknesses are reproduced below:

- Manufacturing and Life Cycle Management Concerns - (1195).
- Security Flow Issues - (1196).
- Integration Issues - (1197).
- Privilege Separation and Access Control Issues - (1198).
- General Circuit and Logic Design Concerns - (1199).
- Core and Compute Issues - (1201).
- Memory and Storage Issues - (1202).
- Peripherals, On-chip Fabric, and Interface/IO Problems - (1203).
- Security Primitives and Cryptography Issues - (1205).
- Power, Clock, and Reset Concerns - (1206).
- Debug and Test Problems - (1207).
- Cross-Cutting Problems - (1208).

9 AI and ML use for Hardware Security and Mitigation of Hardware vulnerabilities

9.1 Detection of Hardware Trojans (HTs) and Counterfeit Integrated Circuits (ICs)

9.1.1 Detection of Hardware Trojans (HTs)

9.1.1.1 Introduction

Per [i.39], modern IC are dispersed in their development and production that is characterized by globally distributed outsourcing activities. The focus of these activities is on core competency, manufacturing efficiency, and reduction of design cycle and turnaround time. The use of third-party EDA tools and untrusted foundries creates opportunities for malicious entities to insert Hardware Trojan at various stages of the IC design process. The malicious modifications of a clean design by HT can lead to such severe consequences as disclosure of confidential and/or secret information, denial of service, modification of major functionalities, etc. HT has become one of the most critical threats to IC production for commercial, consumer as well as military applications, refer to [i.52].

[i.39] lists these two typical HT parts: the Trojan trigger and the Trojan payload. The trigger is designed to activate the Trojan under certain conditions, and the effect of the Trojan depends on the Trojan payload. The Trojan trigger is usually designed to be off the critical functional path and is rarely activated. In addition, the Trojan activity is usually dormant during the normal functional execution of the circuit. These factors make it extremely challenging to detect the Trojan due to its stealthy nature. [i.53] and [i.54] classify HTs based on their five attributes: insertion phase, abstraction level, location, trigger, and payload. Per [i.55], an attacker can utilize malicious processors to violate operating system exceptions and modify the open-source processor to create malicious firmware. [i.56] and [i.57] demonstrated that Trojans can be introduced during EDA design flow and high-level synthesis Additionally, [i.59] and [i.80] stipulated that Trojan can be activated under unexpected conditions or by silicon wear-out.

Various ML algorithms can be used to detect HTs. [i.39] mentions that process drift (i.e. the improvement in fabrication process over time that is not reflected in the technology models released to a design house, see [i.51]) can affect the accuracy of ML-based Trojan detection schemes in cases where the model has been trained using the pre-fabrication data. Therefore, as stated in [i.39], the impact of such drift needs to be modelled to enhance the Trojan detection accuracy.

9.1.1.2 Use of SVM

SVM has been applied in many prior works for detecting HTs. When a golden (i.e. Trojan-free) design is available, on-chip data analysis, gate-level netlist analysis, and runtime traffic information are utilized as feature vectors to distinguish between Trojan and Trojan-free chips using SVM. [i.60] recommends the on-chip power consumption traces in the frequency domain are adapted to classify Trojans with a two-class SVM. [i.61] states that an OC-SVM with RBF kernel trained by the transmission power data collected from on-chip data sensor demonstrates its high accuracy of Trojan detection. [i.68] proposes another OC-SVM for Trojan detection using the minimum number of gates between the input and output nets. [i.67] describes the extraction of Trojan attack features from the on-chip traffic to train an SVM for Trojan classification. These models can be further updated to include the latest attacks with Modified Balanced Winnow (MBW) algorithm [i.63] and [i.62] show how SVM is chosen to detect Trojans at run-time based on the hardware complexity analysis of traffic diversion, route looping or core spoofing attack.

9.1.1.3 Use of DNN

DNN models such as Back-Propagation (BP) and Multi-layer Neural Networks are used in various Trojan detection approaches when a golden design is not available. Direct current measurements of a post-deployment chip can be performed anytime when needed. A one-class neural network is trained by the trusted evaluation chip to classify whether the fabricated chip contains Trojans, see [i.64]. BP neural networks are used to classify the features of power consumption traces to detect designs subjected to trojan modification, refer to [i.65]. By passing the features through the hidden layers of the neural network, relevant features will be extracted and used to train the classifier and enhance the classification accuracy. A multi-layer neural network is also applied on gate-level netlist to classify the design of the Trojan, see [i.66] where several features of the netlist are defined and used to train the model for detecting the network corresponding to a Trojan.

9.1.1.4 Use of other methods

Statistical learning and Bayesian inference are used to classify the gate profiles and process variation collected from the netlists of Trojan-free and Trojan-inserted chips, refer to [i.69]. A run-time monitoring approach is proposed to detect HTs in microprocessor cores by utilizing Half-Space trees (HS-trees), see [i.70]. HS-trees constitutes a one-class classifier that is trained to provide an early alert of Trojan activation by detecting anomalies in the data streams. Controllability and Observability for hardware Trojan Detection (COTD), refer to [i.71] use unsupervised k-means clustering to isolate Trojan signals based on the controllability and observability analysis of gate-level netlist. This technique shows its capability in detecting Trojan with high accuracy and low cost, even in the absence of a golden design. Clustering-based learning is also demonstrated to detect Trojan logic by classifying weakly correlated nodes or functionally isolated gates in the netlist, see [i.72].

9.1.2 Detection of Counterfeit Integrated Circuits (ICs)

9.1.2.1 Introduction

[i.73] states that the number of ICs used in electronic systems has increased significantly over the past decades, due to the enhanced complexity of applications and systems. The fabrication of these ICs is outsourced to reduce the overall manufacturing cost, which can lead to the presence of counterfeit IC components. Per [i.74], these components can become a crucial threat to the applications related to critical infrastructure systems. Since counterfeiting is a rising threat to the IC manufacturing industry, it is increasingly important to investigate the vulnerabilities of the IC supply chain. [i.73] lists analogue ICs, microprocessor ICs, memory ICs, programmable logic ICs, and transistors as the most commonly counterfeited components. [i.73] also states that a large proportion of counterfeit ICs are recycled.

The detection of counterfeit components faces significant challenges, e.g. a wide variety of counterfeit types and the difficulty to inspect potential counterfeit ICs. To improve the detection, it is important to develop the regulation of defects and a unique classification of counterfeit components.

[i.39] explains the use of ML for this purpose as follows. Similar to the case of HT detection, if the model deployed for identifying a counterfeit IC is built based on the pre-fabrication chip specifications, the impact of process variations and process drifts needs to be taken into account when building the ML-based model.

9.1.2.2 Use of SVM

Local Binary Patterns (LBP) are non-parametric local features that can be used to train an SVM model to distinguish counterfeit and authentic ICs from their registered x-ray images as shown in [i.75]. An OC-SVM is used to classify the used and recycled components from the tests, measurements, and analyses of Early Failure Rate (EFR), see [i.82] and [i.84]. This model is also used to compare frequency, noticeable performance degradation, and other quality metrics under certain stress conditions to identify recycled FPGAs, refer to [i.83].

9.1.2.3 Use of ANNs

Artificial Neural Network (ANN) models have been used to provide efficient visual inspections by classifying images of defective and non-defective ICs with image processing techniques, see [i.76]. A similar strategy is applied with X-ray microscopy of an IC die to differentiate counterfeit from authentic devices with auto-encoder and DNN, see [i.75]. DNN is also used to train water-level parametric measurement to identify ICs fabricated in different facilities, which makes it possible to determine if the chips are of the same origin, refer to [i.77].

9.1.2.4 Use of other methods

[i.39] states that LR is used with x-ray 3D imaging to distinguish authentic and recycled ICs by detecting traces of delamination of the dies, as stated in [i.78]. Path delays due to aging devices in ICs are leveraged to identify recycled and brand-new devices by establishing fingerprints of brand-new devices with Principle Component Analysis (PCA), refer to [i.79]. [i.80] states that Look-up table characteristics and performance degradation values are used to train k-mean clustering algorithms, where the recycled FPGAs suffer a higher variation of the performance profile. Per [i.81], EM fingerprints of ICs are also used in PCA to detect cloned counterfeit ICs.

Annex A:
Hardware security standardization ecosystem

A.1 IETF RATS WG (Remote Attestation Procedures)

The IETF RATS WG defines "Remote Attestation" as the process of establishing the state of the hardware and software executing on a remote endpoint, such as the processor, device type, or operating system.

The IETF RATS effort is strongly supported by participation from TCG Network Equipment WG, Global Platform Trusted Platform Services WG, US NIST, FIDO, and other SDOs as well as many operating system and chip vendors. TCG has already contributed documents on RATS Architecture [i.20], RATS Time-Based Uni-Directional Attestation (TUDA) [i.22], RATS TPM-based Network Device Remote Integrity Verification [i.23] and RATS YANG Data Model for Challenge-Response-based Remote Attestation Procedures using TPMs [i.24]. GlobalPlatform has already contributed a document on RATS Entity Attestation Token (EAT) [i.21].

Remote endpoints can attest to the Platform Integrity of endpoints by sending trusted assertions about the security-related functionality of those endpoints. A number of ad hoc solutions already exist in this space, but SDO alignment is sorely needed on terminology, e.g. what can be considered attestation evidence, interfaces for establishing trust, and attestation data models.

IETF RATS has chosen JSON Web Token [i.19] and CBOR Web Token [i.18] to encode the claims and evidence that comprise trusted assertions. IETF RATS has chosen RATS Entity Attestation Token (EAT) [i.21] to convey the claims and evidence that comprise trusted assertions.

A.2 IETF SACM WG (Security Automation and Continuous Monitoring)

The IETF SACM WG defines "Security Automation" as the process of integrating all of the network security components (such as firewalls, anti-virus engines, SEIMs, Network Management Systems, etc.) into a coherent composite system that spans an entire enterprise network or telecom operator network. A SACM system continuously gathers runtime health and posture information from endpoints, intermediate systems, and servers into a central database and supports automated policy-based mitigation of issues in these network components.

The IETF SACM effort is strongly supported by participation from TCG, GlobalPlatform, US NIST, and other SDOs as well as operating system and router vendors. TCG has already contributed documents on SACM Requirements IETF RFC 8248 [i.27], SACM Software Inventory Message and Attributes (SWIMA) for PA-TNC IETF RFC 8412 [i.28], SACM Endpoint Posture Collection Profile [i.26] and SACM Concise Software Identification Tags [i.25].

A.3 IETF SUIT WG (Software Updates for IoT)

Although the original focus of the IETF SUIT WG was the Internet of Things (IoT), there has already been widespread adoption of their architecture and protocol for many networked devices that do not have constrained resources, but do need rapid and frequent updates of their software. GlobalPlatform, Trusted Computing Group (TCG) and several individual organizations are all active contributors.

The IETF SUIT WG defines a Firmware Update Architecture for Internet of Things IETF RFC 9019 [i.32] (now in IETF Last Call) and points out the general need for a reliable and secure firmware update mechanism suitable for devices with constrained resources. It also states that incorporating such an update mechanism is not only a fundamental requirement for fixing vulnerabilities but also an enablement for other important capabilities such as updating configuration settings as well as adding new functionality. In addition to the definition of terminology and architecture, IETF RFC 9019 [i.32] motivates the standardization of a manifest format as a transport-agnostic means for describing and protecting firmware.

The IETF SUIT WG [i.33] has defined an Information Model for Firmware Updates in IoT Devices (now in IETF Last Call) and states that vulnerabilities with Internet of Things (IoT) devices have raised the need for a reliable and secure firmware update mechanism that is also suitable for constrained devices. IETF RFC 9124 [i.33] also affirms that ensuring that for devices to function and remain secure over their service life requires an update mechanism to fix vulnerabilities, to update configuration settings, as well as adding new functionality.

In addition, IETF RFC 9124 [i.33] states that one component of such a firmware update is a concise and machine-processable meta-data document, or manifest, that describes the firmware image(s) and offers appropriate protection. [i.33] describes the information present in the manifest.

The IETF SUIT WG has defined a CBOR-based Serialization Format for the SUIT Manifest [i.34] (work-in-progress) while describing the format of a manifest. [i.34] defines a manifest as a bundle of metadata about code/data obtained by a recipient (chiefly the firmware for an IoT device), where to find that code/data, the devices to which it applies, and cryptographic information protecting the manifest. [i.34] also states that software updates and Trusted Invocation both tend to use sequences of common operations, so the manifest encodes those sequences of operations, rather than declaring the metadata.

A.4 IETF TEEP WG (TEE Provisioning)

The IETF TEEP WG was formed to generalize the concepts of the GlobalPlatform TEE [i.16] across multiple CPU and OS architectures and to support a common approach to provisioning the Hardware-Mediated Environment Enclaves (HMEEs). GlobalPlatform, and Trusted Computing Group (TCG) and several individual organizations are all active contributors.

The IETF TEEP WG has defined a Trusted Execution Environment Provisioning (TEEP) Architecture [i.34] with the following Abstract:

> *"A Trusted Execution Environment (TEE) is an environment that enforces that any code within that environment cannot be tampered with, and that any data used by such code cannot be read or tampered with by any code outside that environment. This architecture document motivates the design and standardization of a protocol for managing the lifecycle of trusted applications running inside such a TEE".*

The IETF TEEP WG has defined a Trusted Execution Environment Provisioning (TEEP) Protocol [i.35] with the following Abstract:

> *"This document specifies a protocol that installs, updates, and deletes Trusted Applications (TAs) in a device with a Trusted Execution Environment (TEE). This specification defines an interoperable protocol for managing the lifecycle of TAs".*

A.5 Trusted Computing Group (TCG)

Trusted Computing Group has defined the architecture and details of security components using hardware Roots-of-Trust (RoTs) that have been implemented by the computer industry over the past 20 years to protect computing infrastructure and billions of end points.

TCG Trusted Platform Module (TPM)

TCG has created the Trusted Platform Module [i.30] cryptographic capability, which enforces specific behaviours and protects the system against unauthorized changes and attacks such as malware and root kits. As computing has expanded to different devices and infrastructure has evolved, so too has TCG extended the concept of trusted systems well beyond the computer-with-a-TPM to other devices, ranging from hard disk drives and mobile devices to IoT devices, network nodes, wearables, network fog and cloud nodes.

Standards-based Trusted Computing technologies developed by TCG members now are deployed in enterprise systems, storage systems, networks, embedded systems, and mobile devices and can secure cloud computing and virtualized systems. Thousands of vendors offer a variety of Trusted Computing-based products, including hardware and applications.

TCG building blocks include boot and run-time platform integrity, runtime integrity protection, secure storage, and remote attestation (of platform integrity).

TCG Runtime Integrity Preservation for Mobile Devices

Modern mobile devices use secure boot to ensure they start up in an expected state. Runtime integrity preservation ensures that the device continues to behave in an expected manner following a successful secure boot. This capability is particularly relevant because mobile devices operate for weeks or months after each secure boot. TCG Runtime Integrity Preservation in Mobile Devices (RIP) [i.29] addresses the challenge of platform integrity by recommending best practices and mechanisms to preserve the critical portions of the runtime state of mobile devices during operation.

A.6 GlobalPlatform (GP)

GlobalPlatform is a non-profit industry association driven by approximately 80 member companies.

GlobalPlatform protects digital services by standardizing and certifying a security hardware/firmware combination, known as a secure component, which acts as an on-device trust anchor. This facilitates collaboration between service providers and device manufacturers, empowering them to ensure adequate security within all devices to protect against threats.

GlobalPlatform specifications also standardize the secure management of digital services and devices once deployed in the field. Altogether, GlobalPlatform enables convenient and secure digital service delivery to end users, while supporting privacy, regardless of market sector or device type. Devices secured by GlobalPlatform include connected cars, set top boxes, smart cards, smartphones, tablets, wearables, and other internet of things (IoT) devices.

Per GlobalPlatform GP-REQ-025 [i.17], a Chain of Trust will always start with a Root of Trust (RoT). If a platform implements a Chain of Trust (CoT), it is initialized by a RoT. However, it is not necessary that the RoT be a pRoT (i.e. Primary RoT). An sRoT (i.e. Secondary RoT) can also serve as the Root for a Chain of Trust.

A.7 The National Institute of Standards and Technology (NIST)

US NIST Cybersecurity White Paper [i.13] states that all security controls need to have a root of trust (RoT) - a starting point that is implicitly trusted. Hardware-based controls can provide an immutable foundation for establishing platform integrity. Combining these functions with a means of producing verifiable evidence that these integrity controls are in place and have been executed successfully is the basis of creating a trusted platform. To substantially reduce the attack surface, [i.13] recommends minimizing the footprint of this RoT that translates to the reduction of the number of modules or technologies to be implicitly trusted.

In addition, [i.13] states that platforms that secure their underlying firmware and configuration provide the opportunity for trust to be extended higher in the software stack. Verified platform firmware can, in turn, verify the operating system (OS) boot loader, which can then verify other software components all the way up to the OS itself and the hypervisor or container runtime layers. The transitive trust described here is consistent with the concept of the chain of trust (CoT)-a method where each software module in a system boot process is required to measure the next module before transitioning control.

[i.13] considers that platform integrity and trust assurance based on hardware security controls can strengthen and complement the extension of the CoT into the dynamic software category. There, according to [i.13] the CoT can be extended even further to include data and workload protection. In addition, hardware-based protections through CoT technology mechanisms can form a layered security strategy to protect data and workloads as they move to multi-tenant environments in a cloud data centre or edge computing facility.

Further, according to [i.13], there are other hardware platform security technologies that can protect data at rest, in transit, and in use by providing hardware-accelerated disk encryption or encryption-based memory isolation. By using hardware to perform these tasks, the attack surface is mitigated, preventing direct access or modification of the required firmware. Isolating these encryption mechanisms to specific hardware can allow performance to be addressed and enhanced separately from other system processes as well.

Annex B: Bibliography

Attack AI-hardware: Fault injection, Side channel

- Y. Liu, L. Wei, B. Luo and Q. Xu: "Fault injection attack on deep neural network", Proc. 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 131-138, November 2017.

NOTE: Available at https://ieeexplore.ieee.org/abstract/document/8203770.

- Lingxiao Wei, Bo Luo, Yu Li, Yannan Liu, Qiang Xu: "I Know What You See: Power Side-Channel Attack on Convolutional Neural Network Accelerators", ACM Computer Security Applications Conference 2018: 393-406.

NOTE: Available at https://arxiv.org/pdf/1803.05847.pdf.

- Jakub Breier, Xiaolu Hou, Dirmanto Jap, Lei Ma, Shivam Bhasin, and Yang Liu: "DeepLaser: Practical Fault Attack on Deep Neural Networks", 2018.

NOTE: Available at https://arxiv.org/pdf/1806.05859.pdf.

- V. Duddu, D. Samanta, D. V. Rao, and V. E. Balas: "Stealing neural networks via timing side channels" arXiv preprint arXiv:1812.11720, 2018.

NOTE: Available at https://arxiv.org/abs/1812.11720.

- W. Hua, Z. Zhang, and G. E. Suh: "Reverse engineering convolutional neural networks through side-channel information leaks", in 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC). IEEE, 2018, pp. 1-6.

- Joseph Clements, Yingjie Lao: "Hardware Trojan Attacks on Neural Networks", 2018 ArXiv.

NOTE: Available at https://arxiv.org/abs/1806.05768.

- L. Batina, S. Bhasin, D. Jap, and S. Picek: "CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel", in 28th USENIX Security Symposium 2019, pp. 515-532.

- Y. Zhao, X. Hu, S. Li, J. Ye, L. Deng, Y. Ji et al.: "Memory trojan attack on neural network accelerators", 2019 Design Automation Test in Europe Conference Exhibition (DATE), pp. 1415-1420.

- Adnan Siraj Rakin, Zhezhi He and Deliang Fan: "TBT: Targeted Neural Network Attack with Bit Trojan", CVPR 2020.

- Dubey, Anuj & Cammarota, Rosario & Aysu, Aydin. (2019): "MaskedNet: A Pathway for Secure Inference against Power Side-Channel Attacks".

NOTE: Available at https://arxiv.org/pdf/1910.13063.pdf.

- Shumailov, Ilia & Zhao, Yiren & Bates, Daniel & Papernot, Nicolas & Mullins, Robert & Anderson, Ross. (2020): "Sponge Examples: Energy-Latency Attacks on Neural Networks".

NOTE: Available at https://arxiv.org/pdf/2006.03463.pdf.

Use AI to attack hardware

- Takaya Kubota, Kota Yoshida, Mitsuru Shiozaki, Takeshi Fujino: "Deep Learning Side-Channel Attack Against Hardware Implementations of AES", 2019 22nd Euromicro Conference on Digital System Design (DSD).

- Debayan Das, Anupam Golder, Josef Danial, Santosh Ghosh, Arijit Raychowdhury, Shreyas Sen: "X-DeepSCA: Cross-Device Deep Learning Side Channel Attack", DAC 2019.

Protecting AI Model/Data using TEE or accelerators

- S. P. Bayerl et al.: "Offline Model Guard: Secure and Private ML on Mobile Devices", 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 2020, pp. 460-465, doi: 10.23919/DATE48585.2020.9116560.

NOTE: Available at https://arxiv.org/pdf/2007.02351.pdf.

- O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani and M. Costa: "Oblivious Multi-Party Machine Learning on Trusted Processors", in USENIX Security 2016.

- T. Hunt, C. Song, R. Shokri, V. Shmatikov and E. Witchel: "Chiron: Privacy-preserving Machine Learning as a Service", CoRR.

NOTE: Available at https://arxiv.org/abs/1803.05961

- S. Chandra, V. Karande, Z. Lin, L. Khan, M. Kantarcioglu and B. Thuraisingham: "Securing Data Analytics on SGX with Randomization", in ESORICS. Springer, 2017.

- S. Ahmed, A. R. Chowdhury, K. Fawaz, and P. Ramanathan: "Preech: A System for Privacy-Preserving Speech Transcription", CoRR, 2019.

NOTE: Available at https://arxiv.org/abs/1909.04198.

- L. Hanzlik, Y. Zhang, K. Grosse, A. Salem, M. Augustin, M. Backes, and M. Fritz: "MLCapsule: Guarded Offline Deployment of Machine Learning as a Service", CoRR, 2018.

NOTE: Available at https://arxiv.org/abs/1808.00590.

- Volos Stavros, Vaswani Kapil and Bruno Rodrigo: "Graviton: Trusted Execution Environments on GPUs", (2018) OSDI'18: October 2018 Pages 681-696.

- Jang, Insu & Tang, Adrian & Kim, Taehoon & Sethumadhavan, Simha & Huh, Jaehyuk: "Heterogeneous Isolated Execution for Commodity GPUs". ASPLOS'2019.

- Zhu, Yuanxin & Yang, Zhao & Wang, Li & Zhao, Sai & Hu, Xiao & Tao, Dapeng: "Hetero-Center Loss for Cross-Modality Person Re-Identification. 2019.

NOTE: Available at https://arxiv.org/abs/1910.09830.

- Jianping Zhu et. al.: "Enabling Rack-scale Confidential Computing using Heterogeneous Trusted Execution Environment", IEEE S&P 2020.

- Tyler Hunt et al.: "Telekine: Secure Computing with Cloud GPUs, NSDI'20".

- Global Platform (GP), http://www.globalplatform.org/.

- Trusted Computing Group (TCG), http://www.trustedcomputinggroup.org/.

- IETF RATS WG (Remote Attestation Procedures), https://datatracker.ietf.org/wg/rats/about/.

- IETF SACM WG (Security Automation and Continuous Monitoring), https://datatracker.ietf.org/wg/sacm/about/.

- IETF CBOR WG (Concise Binary Object Representation), https://datatracker.ietf.org/wg/cbor/about/.

- IETF TLS WG (Transport Layer Security), https://datatracker.ietf.org/wg/tls/about/.

- IRTF Crypto Forum RG, https://datatracker.ietf.org/rg/cfrg/about/.

- TCG Attestation WG, https://trustedcomputinggroup.org/work-groups/attestation/.

Other relevant sources

- ETSI GR SAI 003: "Securing Artificial Intelligence (SAI); Security Testing of AI".
- IETF RFC 8949: "Concise Binary Object Representation (CBOR)".

NOTE: Available at https://tools.ietf.org/html/rfc8949.

- IETF RFC 8610: "Concised Data Definition Language (CDDL): A Notational Convention to Express CBOR and JSON Data Structures".

NOTE: Available at https://tools.ietf.org/html/rfc8610.

- IETF RFC 8742: "CBOR Sequences".

NOTE: Available at https://tools.ietf.org/html/rfc8742.

- IETF RFC 8746: "CBOR Tags for Typed Arrays".

NOTE: Available at https://tools.ietf.org/html/rfc8746.

- IETF RFC 8446: "Transport Layer Security (TLS) Protocol Version 1.3".

NOTE: Available at https://tools.ietf.org/html/rfc8446.

- IETF RFC 8937: "Randomness Improvements for Security Protocols".

NOTE: Available at https://tools.ietf.org/html/rfc8937.

- TCG: "Storage Interface Interactions Specification (SIIS)".

NOTE: Available at https://trustedcomputinggroup.org/resource/storage-work-group-storage-interface-interactions-specification/.

- US NIST FIPS 140-3: "Security Requirements for Cryptographic Modules".

NOTE: Available at https://csrc.nist.gov/publications/detail/fips/140/3/final.

- US NIST FIPS 186-4: "Digital Signature Standard (DSS)".

NOTE: Available at https://csrc.nist.gov/publications/detail/fips/186/4/final.

- US NIST FIPS 186-5 (draft): "Digital Signature Standard (DSS)".

NOTE: Available at https://csrc.nist.gov/publications/detail/fips/186/5/draft.

- US NIST FIPS 202: "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions".

NOTE: Available at https://csrc.nist.gov/publications/detail/fips/202/final.

- US NIST SP800-53: "Security and Privacy Controls for Information Systems and Organizations".

NOTE: Available at https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final.

- US NIST SP800-160: "Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems".

NOTE: Available at https://csrc.nist.gov/publications/detail/sp/800-160/vol-1/final.

- US NIST SP800-186 (draft): "Recommendations for Discrete Logarithm-Based Cryptography: Elliptic Curve Domain Parameters".

NOTE: Available at https://csrc.nist.gov/publications/detail/sp/800-186/draft.

- US NIST SP800-193: "Platform Firmware Resiliency Guidelines".

NOTE: Available at https://csrc.nist.gov/publications/detail/sp/800-193/final.

- US NIST SP800-207: "Zero Trust Architecture".

NOTE: Available at https://csrc.nist.gov/publications/detail/sp/800-207/final.

- US NIST SP800-208: "Recommendation for Stateful Hash-Based Signature Schemes".

NOTE: Available at https://csrc.nist.gov/publications/detail/sp/800-208/final.

- US NIST SP800-211: "2019 NIST/ITL Cybersecurity Program Annual Report".

NOTE: Available at https://csrc.nist.gov/publications/detail/sp/800-211/final.

- IETF draft-ietf-teep-protocol-07: "Trusted Execution Environment Provisioning (TEEP) Protocol".

NOTE: Available at https://datatracker.ietf.org/doc/draft-ietf-teep-protocol/.

- M. Musser and A. Garriott: "Machine Learning and Cybersecurity", CSET, Center for Security and Emerging Technologies.

NOTE: Available at https://cset.georgetown.edu/publication/machine-learning-and-cybersecurity/.

History

Document history		
V1.1.1	March 2022	Publication