		1	*2*	*3*
A. Basic Security Building Blocks				
A.1 Asymmetric encryption				
A.1.1	RSA (Rivest-Shamir-Adleman) is a foundational and widely used public-key (asymmetric) cryptographic algorithm that enables secure data transmission and digital signatures.		√	√
A.1.2	ECC (Elliptic Curve Cryptography, an efficient public-key cryptography method offering strong security with smaller keys compared to methods like RSA. Its security is rooted in the mathematical properties of elliptic curves.)		√	√
A.1.3	ECDSA (Elliptic Curve Digital Signature Algorithm) is a public-key cryptographic algorithm used for creating and verifying digital signatures. It does not provide encryption, but rather guarantees the authenticity, integrity, and non-repudiation of a digital message.		√	
A.1.4	ECDH (Elliptic Curve Diffie-Hellman) is a secure and efficient key agreement protocol that uses Elliptic Curve Cryptography (ECC) to allow two parties to establish a shared secret key over an insecure communication channel.		√	
A.1.5	ML-KEM (formerly CRYSTALS-Kyber): This is the primary PQC algorithm chosen by NIST for general encryption and key exchange. It's a KEM, meaning it enables two parties to establish a shared secret key over a public channel.			
A.1.7	Classic McEliece: A code-based algorithm also considered for key encapsulation, known for its longevity and resistance to attacks, but with larger key sizes.			
A.1.8	ML-DSA (formerly CRYSTALS-Dilithium): Chosen by NIST as the primary algorithm for digital signatures. It allows for generating and verifying digital signatures, which are used to detect unauthorized modifications and authenticate the sender's identity.			
A.1.9	FN-DSA (formerly FALCON): Also selected for digital signatures, optimized for smaller and faster signatures, although with potential implementation challenges.			
A.1.10	SLH-DSA (formerly SPHINCS+): Standardized as a backup digital signature scheme due to its hash-based approach, offering a different mathematical foundation than the lattice-based ML-DSA and FN-DSA. It is stateless and particularly suitable for scenarios requiring lower throughput and higher memory footprints, such as code signing.			
A.2 Symmetric encryption				
A.2.1	Advanced Encryption Standard (AES): The most prevalent symmetric encryption algorithm, adopted by the U.S. government for classified information.	√	√	
A.2.2	ChaCha20: A modern stream cipher known for its speed and security, especially on devices with limited hardware resources.			
A.2.3	Camellia: A symmetric block cipher developed in Japan with security and performance comparable to AES.			
A.2.5	Triple DES (3DES): An enhancement to DES that applies the DES algorithm three times to each data block, effectively increasing the key length to 112 bits.	√		
A.2.6	Twofish: A symmetric block cipher that was a finalist in the AES competition. It's known for its flexibility in key sizes (up to 256 bits) and efficient performance, especially in low-power devices.			
A.2.7	Serpent: Another symmetric block cipher that was an AES finalist. It was designed with a focus on maximizing security, employing 32 rounds of encryption for a strong security margin.			
A.3 Secure hash function				
A.3.1	SHA-256: Produces a 256-bit hash and is widely used, particularly in the Bitcoin blockchain and for digital signatures.	√	√	
A.3.2	SHA-512: Generates a 512-bit hash, offering higher collision resistance compared to SHA-256.		√	
A.3.3	SHAKE128, SHAKE256: Extendable-Output Functions (XOFs) that can produce output of arbitrary length.		√	
A.3.4	BLAKE2 / BLAKE2s / BLAKE2b: A family of fast and secure hash functions, known for their performance optimizations. BLAKE2 was a finalist in the SHA-3 competition			
A.3.5	RIPEMD-160: A secure hash function, widely used in cryptography, including PGP and Bitcoin.			
A.3.6	Whirlpool: A secure cryptographic hash function producing 512-bit hashes, adopted by ISO/IEC as part of an international standard.			
A.4 Hardware random number generator				
A.4.1	Electronic Noise based: Including Johnson noise (thermal noise in resistors) and Zener noise (voltage fluctuations in Zener diodes).		√	
A.4.2	Quantum Phenomena based: Such as photon path splitting, photon arrival times, or quantum superposition states, which are harnessed by Quantum Random Number Generators (QRNGs).			
A.4.3	Radioactive Decay based: Measuring the unpredictable decay events of radioactive materials.			
A.4.4	Free-Running Oscillators (FROs) based: Electronic oscillators designed to produce chaotic and unpredictable output due to inherent jitter and noise.			
A.5 Anti-tampering circuit				
A.5.1	Conductive Meshes: Delicate patterns of conductive material embedded within the device's casing or over sensitive areas. Any breach or physical manipulation of these meshes breaks the circuit, triggering a tamper event.			
A.5.2	Environmental Sensors: Integrated temperature, voltage, or current sensors that constantly monitor the device's operating conditions and detect deviations from expected norms.		√	
A.5.3	Secure Enclosures and Packaging: Designed to make physical access difficult and leave clear evidence if a breach occurs, sometimes incorporating sensors directly.			
A.6 Physical Unclonable Function (PUF)				
A.6.1	SRAM PUF: The most common and mature type, used frequently with open-source frameworks like OpenTitan. It relies on the manufacturing variations that cause a standard SRAM cell to unpredictably settle into a 0 or 1 state when powered on.		√	
A.6.2	DRAM PUF: Exploits the random initial power-up state of DRAM cells or differences in their data retention characteristics. It is useful for generating device-specific IDs in systems that already contain DRAM.			
A.6.3	Flip-flop (or Butterfly) PUF: Creates a PUF by cross-coupling two latches. Upon power-up, random process variations cause the circuit to settle into one of two stable states, generating a unique and repeatable bit.			
A.6.4	Memristor PUF: An emerging technology that uses the unique properties of memristors (a type of passive circuit element) to derive a device's identity.			
B. Digital Side Channel Prevention Techniques				
B.1 Constant-time algorithms				

B.1.1	Cryptographic Accelerators/Instructions: CPUs often include specialized hardware units or instructions (e.g., Intel AES-NI) designed to perform cryptographic operations like AES or SHA in a fixed number of cycles, independent of the input data or key values. This provides a strong guarantee of constant-time execution for these specific operation	√	√	
B.1.2	Constant Memory Access Patterns: Design the hardware so that memory access patterns (e.g., cache hits/misses, memory addresses accessed) are independent of secret values. This can involve techniques like always accessing all possible memory locations within a relevant range, or using dummy reads/writes to equalize access times.			
B.1.3	Dataflow Linearization: Ensure that the data flow within the hardware design is always the same, regardless of the secret values. This helps prevent variations in execution paths or resource contention that could leak information.			
B.1.4	Control Flow Linearization: Design the hardware's control flow to avoid data-dependent branches or loops, which could lead to variable execution times. This might involve using conditional assignments and bitwise operations instead of if statements or variable-length loops.			
B.2 Data Masking				
B.2.1	Boolean masking: Splits sensitive data into multiple shares using bitwise XOR operations, e.g., representing a sensitive variable 'a' as a_m = a ⊕ r, where 'r' is a random mask. The original value can be recovered by a second XOR operation, a_m ⊕ r = a.		√	
B.2.2	Arithmetic masking: Splits sensitive data into multiple shares using modular arithmetic, where the sum of the shares equals the original data.			
B.3 Randomization and Diversification				
B.3.1	Blinding: Randomizing the inputs to cryptographic operations (e.g., XORing data with random masks) ensures that the actual operation is performed on a randomized version of the data, which the attacker cannot directly control or observe.		√*	√
B.3.2	Timing Randomization (Random Delays/Dummy Operations): Concept: Introduce variable delays or execute computationally irrelevant "dummy" operations during sensitive computations to obscure timing patterns.	√	*	
B.3.3	Clock/Frequency Randomization: Dynamically varying the clock frequency or introducing jitters during cryptographic operations. his desynchronizes the observable signals (like power consumption) from the actual operations, complicating power analysis and electromagnetic attacks.			
B.3.4	Dynamic Partial Reconfiguration (DPR) on FPGAs and Structured ASICs:DPR allows specific regions (reconfigurable partitions) of an FPGA or Structured ASICs (predefined logic blocks and routing, and customization is limited to a few specific layers during manufacturing) to be reconfigured at runtime with new functionality or different implementations of the same function, without interrupting the operation of other parts of the chip.			
B.3.5	Redundant swappable unit (RSU): Multiple, functionally equivalent implementations of a cryptographic algorithm, perhaps with different internal structures or data paths, can be swapped in and out. This introduces diverse leakage patterns over time, requiring an attacker to build distinct models for each variant, significantly increasing their effort.			
B.3.6	Clearing Registers with Pseudo-Random Data: Upon reset or if initiated by software, all major key and data registers inside the AES module are cleared with pseudo-random data (PRD). This helps to reduce SCA leakage when both writing these registers for reconfiguration and when clearing the registers after use.		√	
B.4 Noise Injection				
B.4.1	Power Noise Injectors (Dedicated Noise Generators):These are specialized hardware modules designed to inject random or pseudorandom current fluctuations directly into the power supply lines of sensitive components (e.g., cryptographic cores).	√	*	
B.4.2	Jitter Injection in Clocking Mechanisms: Intentionally introducing small, random variations (jitter) into the clock signal feeding sensitive digital logic.			
B.4.3	Noise injection in operations/Cycles: Concept: Incorporating hardware that can execute "dummy" operations or insert random idle cycles during sensitive computations.		*	
B.4.4	Substrate Noise Injection: Intentional injection of noise currents into the silicon substrate of an Integrated Circuit (IC).			
B.5 Dynamic Power Management				
B.5.1	Dynamic Voltage and Frequency Scaling (DVFS): Adjusting the operating voltage and frequency of a processor or other digital component in real-time. Lowering voltage and frequency significantly reduces dynamic power consumption.			
B.5.2	Dynamic Clock Gating: Disabling the clock signal to inactive or idle blocks within a circuit, thereby eliminating dynamic power consumption in those blocks. Clock signals consume a significant portion of power due to switching activity.			
B.5.3	Dynamic Power Gating / Sleep Transistors: Physically disconnecting idle blocks or entire regions of a chip from the power supply, eliminating both dynamic and leakage power consumption (leakage becomes dominant at smaller technology nodes).			
B.5.4	Adaptive Body Biasing (ABB): Dynamically adjusting the bulk voltage (body bias) of transistors to control their threshold voltage vth, thereby managing leakage current. Increasing vth reduces leakage but can impact performance.			
B.5.5	Dynamic Memory Management: Optimizing power consumption in memory sub-systems by dynamically controlling refresh rates, memory access patterns, and putting memory banks into low-power states.			
C. Analog Side Channel Prevention Techniques				
C.1 Power balancing and Filtering:				
C.1.1	Complementary Circuitry: This involves designing circuits where operations on a '1' bit consume power in a way that is compensated by complementary circuitry for a '0' bit, ensuring that the total power draw remains constant irrespective of the data's Hamming weight.			
C.1.2	Constant Weight Codes: Data can be encoded using constant weight codes (e.g., each codeword always has the same number of '1's), ensuring that the power consumption associated with processing the encoded data remains consistent, irrespective of the original data value.			
C.1.3	Dual-Rail Logic: This technique uses two complementary wires for each signal, where one wire is high and the other is low, ensuring a more balanced power consumption profile.		√	
C.1.4	Reducing Power Fluctuations: By regulating voltage and filtering out electrical noise, power line conditioning reduces the subtle power variations that attackers might exploit for power analysis. A more stable and "clean" power supply offers fewer discernible patterns for an attacker to analyze.			
C.1.5	Isolation from External Monitoring: Power line isolation, a form of conditioning, involves decoupling the power supply of a sensitive device from the external source. This can be achieved by using capacitors and control switches that supply power to the device from charged capacitors while the external source charges others, effectively presenting a less informative power signature to an attacker.			
C.1.6	Power Supply Decoupling: Designing decoupling circuits to isolate the power supply of sensitive logic gates from the main chip power supply, making it harder for an adversary to observe instantaneous power consumption related to specific operations.			
C.2 Voltage Monitoring:				
C.2.1	Resistor Divider: This is a simple and common method where resistors divide down the voltage to a level measurable by a comparator or ADC, often used for undervoltage detection.			√

C.2.2	Comparator and Reference Voltage: A comparator compares the monitored voltage to a precise reference voltage, triggering an alert or action if the voltage deviates from a set threshold (e.g., overvoltage or undervoltage).			
C.2.3	Onboard Analog-to-Digital Converter (ADC): Many microcontrollers and processors include built-in ADCs that can be used to sample and digitize voltage levels for software-based monitoring and analysis.		√	
C.2.4	Voltage Supervisor ICs / Reset ICs: These integrated circuits are specifically designed for voltage monitoring, often incorporating features like accurate voltage thresholds, hysteresis, and reset delay, particularly useful for power-on reset or fault detection in power supplies.			
C.2.5	Dedicated Voltage Monitoring ICs: Specialized ICs with high precision for monitoring multiple system power-supply currents and voltages, often with digital interfaces like I2C for host controller communication.			
C.2.6	Line Voltage Monitors: Devices that can monitor AC line voltages for various anomalies such as imbalance, high/low voltage, phase loss, or incorrect sequencing, crucial for industrial applications.			
C.2.7	Battery Voltage Monitoring Systems: Implementing voltage monitoring in battery-powered devices to track battery status, prevent over/undercharging, and ensure proper shutdown before critical voltage depletion.			
C.3 Clock Monitoring:				
C.3.1	Frequency Monitoring: This involves continuously checking if the clock frequency stays within a specified tolerance range. Deviations (e.g., higher or lower than expected, or even a complete clock stop) can indicate malfunctions or malicious attacks.			√
C.3.2	Clock Glitch Detection: Monitoring for sudden, brief fluctuations in the clock signal (glitches) that can cause errors or enable security exploits.			
C.3.3	Clock Stop Detection: Identifying a complete cessation of the clock signal, which is a critical fault in any clocked system.			
C.3.4	Duty Cycle Monitoring: Ensuring the clock signal maintains a consistent high-to-low ratio, as variations can impact timing and performance.			
C.3.5	Jitter Measurement: Quantifying the short-term variations in the clock signal's period, which can accumulate and cause timing violations in high-speed systems.			
C.3.6	Phase Noise Analysis: Characterizing the spectral purity of the clock signal, crucial for applications like wireless communication where clean, stable frequencies are vital.			
C.3.7	Security-Focused Clock Monitors: Integrated within secure ICs, these specialized monitors detect attempts at physical clock attacks (e.g., clock glitching, frequency manipulation) aimed at bypassing security mechanisms or extracting sensitive data, and trigger countermeasures like system shutdown or alarm generation.			
D. Control Flow Manipulation Attack Prevention/Detection Techniques				
D.1 Data Execution Prevention (DEP) / No-Execute (NX):				
D.1.1	Data Execution Prevention (DEP)/NX Bit: This is a core hardware security feature present in most modern processors.			
D.1.2	The NX (No Execute) bit, when set, prevents code from being executed in memory pages designated for data.		√	
D.2 Shadow Stack:				
D.2.1	Shadow Stacks (Hardware-enforced): This technique involves a separate, hardware-protected stack (the shadow stack) that mirrors the regular call stack. When a function is called, the return address is pushed onto both stacks. Upon return, the two addresses are compared, and a mismatch triggers an error.			
D.3 Address Space Layout Randomization (ASLR):				
D.3.1	Address Space Layout Randomization (ASLR) is a computer security technique used by modern operating systems to prevent the exploitation of memory corruption vulnerabilities, such as buffer overflow attacks. While ASLR is often implemented at the OS level, its effectiveness relies on the processor's ability to randomize memory layouts. The NX bit, for instance, which is a hardware security feature, is required for ASLR on some platforms			
D.4 Tagged data path and control path:				
D.4.1	Data integrity checks: Tags can be used to store integrity information, such as cryptographic hashes or checksums, associated with data values. Any attempt to modify the data without updating the tag would be detected, preserving data integrity and potentially thwarting data-only attacks that modify data without altering control flow			√
D.4.2	Validating control flow transitions: Tags associated with instructions can define the valid targets for control flow transfers (e.g., function calls, returns, jumps). Hardware or software checks can then ensure that the program's execution follows the intended control flow graph, preventing control flow hijacking attacks like ROP and JOP			√
E. Memory corruption and Rowhammer Attack Prevention Techniques				
E.1 Memory Tagging				
E.1.1	Memory Tagging Extensions (MTE) and Application Data Integrity (ADI): Hardware-based extensions like ARM MTE and SPARC ADI use tags associated with memory granules and pointers. The hardware then verifies that these tags match on memory accesses, probabilistically detecting memory corruption issues like out-of-bounds accesses and use-after-free errors.			√
E.2 Memory corruption Attack Prevention				
E.2.1	Tamper-Proof Memory: techniques like One Time Programmable (OTP) eFuse memory, to store critical assets like cryptographic keys, can be highly resistant to physical attacks and tampering.		√	
E.2.2	Memory Protection Units (MPUs): These are hardware components within the CPU that control access to different memory regions. They enforce access rules, preventing one process from accessing memory allocated to another process and isolating privileged code (like the operating system kernel) from user applications.			
E.2.3	Memory Isolation using Hardware Virtualization: Features like Windows' Core Isolation and Memory Integrity use hardware virtualization to create isolated environments for critical system processes, protecting them from tampering and unauthorized access.			
E.3 Direct Memory Access (DMA) safety protection				
E.3.1	A dedicated controller that provides a hardware isolation layer.	√	√	
E.3.2	Use I/O Memory Management Unit (IOMMU) as a gatekeeper between peripheral devices and the system's main memory.			
E.4 Rowhammer Attack Prevention Techniques				
E.4.1	ECC Memory: Error-correcting code (ECC) memory can detect and correct single-bit errors, making it more challenging for attackers to leverage Row Hammer for data corruption or privilege escalation.	√	√	
E.4.2	Target Row Refresh (TRR): This is the most common hardware defense. The memory controller or the DRAM chip itself tracks the number of times a row is accessed within a given time window. If a threshold is reached, it proactively issues a refresh command to the adjacent "victim" rows to prevent bit flips.			
E.4.3	Increased Refresh Rate: Globally increasing the DRAM refresh rate ensures that memory cells are refreshed more often, giving them less time to lose charge due to electromagnetic interference. This can be implemented via BIOS updates but results in higher power consumption and reduced performance.			