Table 4. Comparison of implementations using a 0.13-µm application-specific integrated circuit library and a 500-kHz clock frequency.

Implementation	Power (µW)			Area[1]	Delay (ns)	Clock cycles[2]	PDP (ns × µW)
	P_{Dyn}	P_{Leak}	Total				
NtruEncrypt (1 arithmetic unit)	4.03	15.1	19.1	2,850	0.69	29,225	13.18
NtruEncrypt (8 arithmetic units)	5.00	22.5	27.5	3,950	0.69	3,682	18.96
NH (integer)	5.47	28.1	33.6	5,291	9.92	64	333.31
PH (polynomial)	3.41	12.1	15.5	2,356	1.35	64	20.93
AES S-box (logic)	0.42	7.67	8.10	1,397	1.61	1	13.04
AES S-box (algebraic)	1.39	2.68	4.07	431	4.68	1	19.05

[1] Area is given in terms of equivalent two-input NAND gates
[2] Number of clock cycles to complete one operation

cuits often require manual design changes.

- At the algorithmic level, a high degree of regularity expresses the uniformity of operations necessary to perform a task, while irregularity characterizes very complex tasks consisting of many atomic operations.

Rabin's scheme, NH, and WH are examples of algorithms with high regularity. Each has one simple underlying function. Block ciphers require serialization of the round function because even a single round can take a considerable amount of chip area. Ciphers with a homogeneous round function, such as AES, have high regularity and therefore seem better suited for serialization than ciphers with a heterogeneous structure, such as DES.

Energy equals the amount of power dissipated over time. Increasing the degree of parallelism increases power consumption, but it also decreases computation time. Because certain elements can have constant size, the power-energy tradeoff depends on the architecture's overall structure. We can find the point of optimality by modeling the energy consumption as a function of the degree of parallelism. Energy per bit encrypted describes the amount of energy needed to encrypt a single message bit. We can use this metric, which is independent of the actual operand length, to compare the energy efficiency of cryptosystems at an equivalent security level.

Functional primitives

Each group of primitives has specific characteristics and suitability for ultralow-power implementation.

Simple logic functions. In this group, the logic function output depends on a small, fixed set of inputs. This includes functions such as XORs of two bit strings (`AddRoundKey` in AES), bit multipliers, and multiplexers. The number of logic gates scales linearly with the data path's width.

Fixed shifts and permutations. In this article's context, *fixed* means that the shifts and permutations aren't data-dependent. Block ciphers such as DES, Serpent, and AES use permutations and expansions as nonlinear diffusion elements. Fixed shifts and rotations serve the same purpose and are frequently used in both block ciphers (such as AES, MARS, Serpent, and Twofish) and hash functions (such as MD2, SHA-1, MD4, and MD5). Common to all of these functions is that their implementation introduces virtually no cost because they require only wiring resources and no logic. They are perfect for any hardware implementation.

Data-dependent shifts. Because of their resistance to differential cryptanalysis, data-dependent shifts (or variable shifts) are used in block ciphers such as RC5, RC6, and MARS. Implementations frequently use barrel shifters to support all possible shifts or rotations.

The delay for one shift operation is proportional to $\log_2 n$, but its area scales with $n\log_2 n$. For situations in which a register follows the shift or rotation, implementing the register as a shift register with parallel load and combining it with additional control logic and a counter might be more power-efficient. Because of the relatively high area cost, variable shifts are poorly suited for ultralow-power implementations unless they're combined with existing registers.

Integer arithmetic. Integer arithmetic primitives, such as addition and multiplication, are often the most costly functions in a cryptographic algorithm. Although we can often implement them in a bit-serial fashion (such as multiplication), the efficient propagation of carries presents a major problem. The carry propagate, or ripple carry adder (the simplest adder form), scales linearly with the word size n, but glitches in the carry chain cause high dynamic power consumption. Various alternatives exist, but they carry a penalty in terms of area and therefore static power consumption. Because of these costs, new ultralow-power algorithms should avoid integer arithmetic.

Arithmetic primitives are frequently combined with modular reduction steps. In trivial cases, a modulus of 2^k means that the algorithm keeps only k bits of the result, truncating excess bits. Finite field arithmetic with a nontrivial modulus adds a fair amount of complexity to the circuit. Simple implementations perform conditional subtractions of the modulus from the result, depending on its most significant bit's value. For certain classes of public-key algorithms that depend heavily on modular arithmetic, using residue number system arithmetic has proven effective for efficient implementation.

Polynomial arithmetic. Polynomial arithmetic—arithmetic in extension fields—is preferable for ultralow-power implementations because of limited carry propagation and improved regularity. Therefore, several algorithms, including AES, are specifically tailored for arithmetic in $GF(2^k)$. We can implement additions in fields of characteristic two