Secure Montgomery Curves over TMVP-Friendly Primes for High-Performance ECC

Murat Cenk[1], N. Gamze Orhon Kılıç[2], Halil Kemal Taşkın[3] and Oğuz Yayla[2]

[1] Ripple Lab Inc, Canada,
[2] Institute of Applied Mathematics, Middle East Technical University, Türkiye,
[3] ASELSAN, Türkiye.

Abstract. We propose *Curve5453* and *Curve6071*, two Montgomery curves over the Crandall prime $2^{545} - 3$ and Mersenne prime $2^{607} - 1$, respectively, providing 271 and 302 bits of classical security. Comprehensive security analysis shows *Curve6071* passes all verifiable *SafeCurves* criteria while *Curve5453* passes all except completeness. We develop TMVP-optimized field multiplication tailored to the arithmetic structure of these primes for 10-limb representations on 64-bit architectures, achieving 12.0% and 20.4% speedups over the closest alternative. ARM64 benchmarks show scalar multiplication completing in 871,898 and 895,028 cycles, respectively, competitive with existing lower-security alternatives such as E-521 (259-bit security) while delivering higher security levels. These curves address a critical gap for hybrid post-quantum constructions requiring classical security commensurate with quantum-resistant components, blockchain systems with decades-long security requirements, and specialized deployments where implementation robustness and enhanced classical security are essential—providing the first SafeCurves-compliant alternatives beyond 260 bits with demonstrated practical performance on modern architectures.

Keywords: ECC · Efficient implementation · TMVP · Montgomery curves · Crandall primes · Mersenne primes · SafeCurves

1 Introduction

The invention of public-key cryptography by Diffie and Hellman in the 1970s revolutionized secure communication by enabling asymmetric protocols [DH76]. Among public-key primitives, *Elliptic Curve Cryptography* (ECC), introduced independently by Miller [Mil86] and Koblitz [Kob87] in 1985, offers strong security with much smaller key sizes and lower computational cost than RSA, making it ideal for resource-constrained platforms. ECC adoption accelerated in the 1990s with the standardization of ECDSA [Nat00], its integration into protocols like TLS, SSH, and IPsec, and endorsements from standards bodies and agencies [Nat05, IEE00, SEC00, Bra05, Nat23]. Beyond the ECDLP, initiatives such as *SafeCurves* [BL14, BL25] expanded evaluation to include resistance to implementation-level attacks.

Designing secure and efficient curves requires optimizing scalar multiplication—the dominant ECC cost—through arithmetic choices (e.g., projective or Montgomery coordinates [HVM04, Mon87]), algorithmic methods (e.g., windowing, NAF, ladders [HVM04, Ber06]), and fast field arithmetic, Karatsuba, Toom-Cook, GLV, or TMVP [Taş19, AC18]. High-performance curve models such as Montgomery, twisted Edwards [BBJ+08, HWCD08], and others offer constant-time, complete formulas, with further gains from precomputation, parallelization, and architecture-specific optimizations [CL15, Ber08].

ECC remains widely deployed and continues to be refined for improved efficiency and security. However, Shor's algorithm, introduced in 1994 [Sho94], demonstrated that public-key cryptosystems would become vulnerable in the presence of large-scale, fault-tolerant quantum computers. This result has far-reaching implications: once scalable quantum adversaries emerge, current public-key cryptosystems will no longer offer adequate security. In response to the threat posed by quantum computing to classical public-key cryptosystems, the cryptographic community launched a global effort to develop quantum-resistant alternatives. A key milestone in this effort is the NIST Post-Quantum Cryptography (PQC) Standardization Project [Nat25], which aims to identify and standardize cryptographic algorithms that remain secure in the presence of quantum adversaries. The project has progressed through multiple evaluation rounds, engaging researchers worldwide in assessing the security, performance, and implementation feasibility of proposed schemes.

While these efforts are critical for long-term preparedness, large-scale, fault-tolerant quantum computers remain unavailable. Häner et al. [HJN+20] estimated that breaking elliptic curves with 128, 192, and 256 bits of classical security would require approximately 2,124, 3,151, and 4,258 logical qubits, respectively—substantial quantum resources that reinforce ECC's practical security in the near term. Consequently, elliptic curve cryptography remains widely deployed and continues to be optimized for performance and robustness, with ongoing interest in curves offering stronger classical security margins while maintaining efficiency on modern platforms. However, a gap exists between theoretical proposals and practical deployment: while the SafeCurves project has established rigorous criteria for implementation security and several compliant curves have been proposed for high security levels, none have achieved significant adoption in blockchain systems or specialized security deployments where non-standardized curves are acceptable.

Addressing this gap is particularly important for hybrid post-quantum constructions, which combine classical ECC with quantum-resistant primitives to provide defense-in-depth security during the quantum transition. Such hybrid approaches are actively deployed in production systems: Google employs X25519Kyber768 hybrid key exchange in Chrome and internal communications [Goo24], while Cloudflare experimented with CECPQ2 combining X25519 and HRSS [Clo19]. Academic research has extensively studied hybrid key exchange protocols [GCRH23, SM19], emphasizing the importance of balanced security levels between classical and post-quantum components to avoid creating weak links in the combined scheme. Current hybrid deployments typically use X25519 (providing approximately 128 bits of security), yet NIST's ML-KEM-768 and other post-quantum components target significantly higher security levels. This asymmetry motivates the development of higher-security elliptic curves that can be paired with stronger post-quantum primitives to maintain security balance in hybrid constructions. Beyond hybridization, blockchain systems and distributed ledger technologies, where cryptographic operations dominate performance and security requirements span decades, can leverage curves specifically optimized for their threat models when standardization constraints do not apply. Specialized security deployments, including custom protocols, high-value communications, and systems with extended operational lifetimes, similarly require curves combining strong security guarantees with practical implementation characteristics. These application domains—particularly hybrid post-quantum constructions requiring balanced security levels—motivate the development of high-security Montgomery curves satisfying SafeCurves criteria for implementation robustness while achieving competitive performance through optimized field arithmetic on modern 64-bit architectures.

To address these requirements, we propose two new Montgomery curves—*Curve5453* and *Curve6071*—over primes with efficient reduction properties that combine a higher classical security level with efficient arithmetic on 64-bit platforms. *Curve6071* is defined over the Mersenne prime $2^{607} - 1$, the next Mersenne prime beyond $2^{521} - 1$ currently employed in NIST P-521. *Curve5453* uses the Crandall prime $2^{545} - 3$, which offers similarly

favorable modular reduction properties. Both curves provide security levels exceeding existing alternatives—271 and 302 bits, respectively—while satisfying SafeCurves criteria for implementation robustness (*Curve6071* satisfies all verifiable criteria; *Curve5453* satisfies all except completeness, which is readily addressed through careful Montgomery ladder implementation; detailed analysis in Section 3). Our curves are designed to satisfy stringent criteria beyond the hardness of the ECDLP, following the SafeCurves methodology to ensure real-world robustness. To enable high-performance implementations on 64-bit architectures, we adopt a field multiplication strategy based on Toeplitz matrix-vector products (TMVP), as introduced in [TC20], which aligns well with 10-limb arithmetic typical for ∼550-600-bit primes. Furthermore, we derive detailed arithmetic complexity models tailored to 64-bit architectures, accounting for the cost of word-level operations such as additions, multiplications, carry propagation, and overflow avoidance via carefully chosen limb sizes. In addition to the TMVP-based strategy, we observe that a specific field element representation for *Curve5453* exhibits a matrix structure on 64-bit platforms analogous to that used for Curve25519 [Ber06, BL09] on 32-bit systems. We implemented and benchmarked field arithmetic for both proposed curves, including multiple multiplication strategies for comparison. The results show that our TMVP-based approach achieves significant performance improvements over the state-of-the-art donna implementation—specifically, a speedup of 12.0% for *Curve5453* and 20.4% for *Curve6071* in field multiplication. Scalar multiplication on our curves requires 871,898 cycles for *Curve5453* and 895,028 cycles for *Curve6071* on ARM64, demonstrating competitive performance with existing lower-security alternatives while providing 271 and 302 bits of security, respectively. These results demonstrate that curves at 271 and 302 bit security levels can achieve practical performance on modern architectures while satisfying implementation robustness criteria. The proposed curves are therefore well-suited for specialized deployments, hybrid post-quantum constructions, and blockchain applications where non-standardized curves with enhanced classical security and verified implementation properties are valuable.

The remainder of this paper is organized as follows. Section 2 introduces Montgomery curves, outlines their essential features, describes the Montgomery ladder, and presents the proposed curves along with their properties. Section 3 summarizes the SafeCurves criteria and analyzes the new curves with respect to these requirements. Section 4 discusses special prime forms, the representation of field elements, associated operations, and efficient field multiplication using TMVP. Section 5 details the implementation and presents the benchmarking results. Section 6 positions our curves within the landscape of existing high-security alternatives, and Section 7 offers concluding remarks.

2 Montgomery Curves

This section outlines Montgomery curves, their key features, and the Montgomery ladder, and then introduces our new curve models. These Montgomery-form curves, defined over large special-structure prime fields, are designed for efficient x-only arithmetic while meeting strong cryptographic security requirements.

Definition 1. Let $\mathbb{F}_q$ be a finite field of odd characteristic. A *Montgomery curve* over $\mathbb{F}_q$ is an elliptic curve defined by the affine equation

$$E_{\mathcal{M}}\colon By^2 = x(x^2 + Ax + 1), \tag{1}$$

where $A, B \in \mathbb{F}_q$ with $B \neq 0$ and $A^2 \neq 4$. These conditions ensure that the curve is nonsingular.

The projective form of this equation is: $\overline{E}_{\mathcal{M}} : BY^2Z = X(X^2 + AXZ + Z^2)$. The point $\mathcal{O} = (0 : 1 : 0)$ serves as the unique point at infinity. The parameter A dictates the

geometry of $E_{\mathcal{M}}$ and determines its j-invariant: $j(E_{\mathcal{M}}) = (256(A^2-3)^3)/(A^2-4)$. This j-invariant indicates the isomorphism class of $E_{\mathcal{M}}$ over $\mathbb{F}_q$. Not all elliptic curves can be expressed in Montgomery form since not all j-invariants correspond to an $A \in \mathbb{F}_q$.

A *quadratic twist* E' of E is an elliptic curve that is not isomorphic to E over $\mathbb{F}_q$ but becomes isomorphic over the quadratic extension $\mathbb{F}_{q^2}$. For Montgomery curves, the parameter B—often called the *twisting factor*—determines the twist: two curves $E_{\mathcal{M}}$ and $E'_{\mathcal{M}}$ with parameters B and B' are isomorphic over $\mathbb{F}_q$ if B/B' is a square in $\mathbb{F}_q$.

A Montgomery curve defines an abelian group with the point at infinity $\mathcal{O}$ as the identity and negation given by $\ominus : (x : y : 1) \mapsto (x : -y : 1)$. For two points $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$, the group law on Montgomery curves is given by the following formulas for addition:

$$\begin{aligned} x_\oplus &= B\lambda^2 - (x_P + x_Q) - A, \\ y_\oplus &= \lambda(x_P - x_\oplus) - y_P \end{aligned} \qquad \lambda = \begin{cases} \dfrac{y_Q - y_P}{x_Q - x_P}, & P \neq Q, \\ \dfrac{3x_P^2 + 2Ax_P + 1}{2By_P}, & P = Q. \end{cases}$$

Every Montgomery curve $E_{\mathcal{M}}/\mathbb{F}_q$ contains the rational point $T = (0, 0)$ of order 2, with translation map

$$\tau_T(x, y) \mapsto \left(1/x,\ -y/x^2\right), \tag{2}$$

which is an involution on the curve. Consequently, $\#E_{\mathcal{M}}(\mathbb{F}_q)$ is always divisible by 4 under the nonsingularity conditions $A \pm 2 \neq 0$ and $B \neq 0$.

The 4-torsion subgroup structure depends on the quadratic residuosity of $B(A+2)$, $B(A-2)$, and A^2-4 in $\mathbb{F}_q$. For example, if $B(A+2)$ is a residue, the points $(1, \pm\sqrt{(A+2)/B})$ have order 4; similarly for $B(A-2)$. If neither is a residue, A^2-4 must be, yielding full rational 2-torsion. The twist $E'_{\mathcal{M}}$ complements this structure, ensuring that both $\#E_{\mathcal{M}}$ and $\#E'_{\mathcal{M}}$ remain divisible by 4, a property relevant for secure cryptographic protocols.

2.1 Single Coordinate Arithmetic

Point addition and doubling can be performed directly via coordinate formulas, but *differential arithmetic* offers greater efficiency on Montgomery curves by using only x-coordinates and avoiding costly y-coordinate computations. Given P, Q on a Montgomery curve with known difference $P - Q$, and projective coordinates $pX = (X_P : Z_P)$, $qX = (X_Q : Z_Q)$, $sX = (X_\ominus : Z_\ominus)$, the sum $P + Q = (X_A : Z_A)$ is computed [CS18] as:

$$\begin{aligned} X_A &= Z_\ominus[(X_P - Z_P)(X_Q + Z_Q) + (X_P + Z_P)(X_Q - Z_Q)]^2, \\ Z_A &= X_\ominus\left[(X_P - Z_P)(X_Q + Z_Q) - (X_P + Z_P)(X_Q - Z_Q)\right]^2, \end{aligned} \tag{3}$$

and the point doubling $[2]P = (X_D : Z_D)$ is given by

$$\begin{aligned} X_D &= t_0^2\, t_1^2, \\ Z_D &= \left(t_0^2 - t_1^2\right)\left[t_1^2 + ((A+2)/4)\left(t_0^2 - t_1^2\right)\right] \end{aligned} \tag{4}$$

where $t_0 = X_P + Z_P$ and $t_1 = X_P - Z_P$. These formulas require no field inversion, and when $P - Q$ is fixed, sX can be normalized to $(X_P : 1)$ to save one multiplication. The addition formula is curve-independent, whereas doubling depends on the curve parameter A.

2.2 Montgomery Ladder

The *Montgomery ladder* is a scalar multiplication method offering inherent resistance to side-channel attacks by ensuring a uniform execution pattern. For curves in *Montgomery form*, it operates solely on x-coordinates using the differential formulas in (3) and (4). Given $pX = (X_P : Z_P) \in E_{\mathcal{M}}/\mathbb{F}_q$, $n \in \mathbb{F}_q$ and the curve constant A, the scalar multiplication $[n]P$ is computed by the ladder in Algorithm 1.

Algorithm 1 `xSMUL`

Input: X_P, $n = (n_0, n_1, \ldots, n_{l-1})_2$, A.
Output: $[n]P$.
1: $Z_P \leftarrow 1, \quad (X_{\mathsf{x}_0}, Z_{\mathsf{x}_0}) \leftarrow (X_P, Z_P)$
2: $(X_{\mathsf{x}_1}, Z_{\mathsf{x}_1}) = \texttt{xDBL}(X_P, Z_P, A)$
3: **for** $i = l-2$ **to** 0 **do**
4: **if** $n_i == 0$ **then**
5: $(X_{\mathsf{x}_0}, Z_{\mathsf{x}_0}, X_{\mathsf{x}_1}, Z_{\mathsf{x}_1}) = \texttt{xDBLADD}(X_{\mathsf{x}_0}, Z_{\mathsf{x}_0}, X_{\mathsf{x}_1}, Z_{\mathsf{x}_1}, X_P, Z_P, A)$
6: **else**
7: $(X_{\mathsf{x}_0}, Z_{\mathsf{x}_0}, X_{\mathsf{x}_1}, Z_{\mathsf{x}_1}) = \texttt{xDBLADD}(X_{\mathsf{x}_1}, Z_{\mathsf{x}_1}, X_{\mathsf{x}_0}, Z_{\mathsf{x}_0}, X_P, Z_P, A)$
8: **end if**
9: **end for**
10: **return** $X_{\mathsf{x}_0}, Z_{\mathsf{x}_0}$.

Observe that in Algorithm 1, the function `xDBL` computes $[2]P$ using equation (4), while `xDBLADD` simultaneously computes $[2]P$ and $P+Q$ using equations (4) and (3).

2.3 New Curves

We propose two new elliptic curves in Montgomery form, defined over large prime fields with special structure. These curves are designed to enable efficient arithmetic while satisfying essential cryptographic criteria. The curve parameters were selected through a fully transparent, deterministic process to ensure rigidity; the complete methodology is detailed in Section 2.3.3.

2.3.1 Curve5453:

Let $\mathbb{F}_{5453}$ be the prime field with $p_{5453} = 2^{545} - 3$. Then *Curve5453* is the Montgomery-form elliptic curve defined over $\mathbb{F}_{5453}$ by the equation

$$E_{\mathcal{M}_{5453}} : y^2 = x^3 + 115794x^2 + x.$$

The curve parameters are $(A, B) = (115794, 1)$. Since $B(A^2 - 4) = 1 \cdot (115794^2 - 4) = 13408250432 \neq 0$, the curve $E_{\mathcal{M}_{5453}}$ is nonsingular. Using the substitutions $x = X/Z$ and $y = Y/Z$, the projective closure of $E_{\mathcal{M}_{5453}}$ is given by $\overline{E}_{\mathcal{M}_{5453}} : Y^2Z = X(X^2 + 115794XZ + Z^2)$, with the unique point at infinity $\mathcal{O} = (0 : 1 : 0)$.

Neither $B(A+2)$ nor $B(A-2)$ is a square in $\mathbb{F}_{5453}$, while $A^2 - 4$ is a quadratic residue, implying that $E_{\mathcal{M}_{5453}}$ has full rational 2-torsion. If α is a root of $E_{\mathcal{M}_{5453}}$, then $1/\alpha$ is also a root; the projective points $(\alpha : 0 : 1)$ and $(1 : 0 : \alpha)$ are of order two and are interchanged by the involution (2). The quadratic twist $E'_{\mathcal{M}_{5453}}$ exhibits a mirrored torsion structure: since both $B(A \pm 2)$ are quadratic nonresidues, $(1, \pm\sqrt{(A+2)/B'})$ and $(-1, \pm\sqrt{(A-2)/B'})$ are twist points of order four.

The curve order satisfies $\#E_{\mathcal{M}_{5453}} = p_{5453} + 1 - t_{5453}$, where t_{5453} is the trace of Frobenius. Its base point $P_{B_{5453}} = (24, \sqrt{24^3 + 115794 \cdot 24^2 + 24})$ has prime order ℓ_{5453}, giving $\#E_{\mathcal{M}_{5453}} = 4\ell_{5453}$ and cofactor 4. Similarly, $\#E'_{\mathcal{M}_{5453}} = p_{5453} + 1 + t_{5453}$ with base point $P'_{B'_{5453}}$ of prime order ℓ'_{5453}, so $\#E'_{\mathcal{M}_{5453}} = 4\ell'_{5453}$ and the twist cofactor is also 4.

2.3.2 Curve6071:

Let $\mathbb{F}_{6071}$ be the prime field with $p_{6071} = 2^{607}-1$. Then *Curve6071* is the Montgomery-form elliptic curve defined over $\mathbb{F}_{6071}$ by the equation

$$E_{\mathcal{M}_{6071}} : y^2 = x^3 + 2720010x^2 + x.$$

The curve parameters are $(A, B) = (2720010, 1)$. Since $B(A^2-4) = 1 \cdot (2720010^2 - 4) = 7398454400096 \neq 0$, the curve $E_{\mathcal{M}_{6071}}$ is nonsingular. Using the substitutions $x = X/Z$ and $y = Y/Z$, the projective closure of $E_{\mathcal{M}_{6071}}$ is given by $\overline{E}_{\mathcal{M}_{6071}} : Y^2Z = X(X^2 + 2720010XZ + Z^2)$, with the unique point at infinity $\mathcal{O} = (0 : 1 : 0)$.

In $\mathbb{F}_{6071}$, neither $B(A+2)$ nor A^2-4 is a quadratic residue, while $B(A-2)$ is, so $(-1, \pm\sqrt{(A-2)/B})$ are order-2 points on $E_{\mathcal{M}_{6071}}$. For the quadratic twist $E'_{\mathcal{M}_{6071}}$ with constant B', $B(A+2)$ being a nonresidue implies that $(1, \pm\sqrt{(A+2)/B'})$ are order-4 points.

The curve order is $\#E_{\mathcal{M}_{6071}} = p_{6071} + 1 - t_{6071}$, where t_{6071} is the trace of Frobenius. Its base point $P_{B_{6071}} = (11, \sqrt{11^3 + 2720010 \cdot 11^2 + 11})$ has prime order ℓ_{6071}, giving $\#E_{\mathcal{M}_{6071}} = 4\ell_{6071}$ and cofactor 4. Similarly, $\#E'_{\mathcal{M}_{6071}} = p_{6071} + 1 + t_{6071}$, with base point $P'_{B'_{6071}}$ of prime order ℓ'_{6071}, so $\#E'_{\mathcal{M}_{6071}} = 4\ell'_{6071}$ and the twist cofactor is also 4.

2.3.3 Parameter Selection:

Our parameter selection follows a deterministic, transparent process that leaves no room for manipulation. For each prime field $\mathbb{F}_p$ with $p = 2^n - c$, we selected the smallest Montgomery parameter A for which both the curve $E_{\mathcal{M}} : y^2 = x^3 + Ax^2 + x$ and its quadratic twist $E'_{\mathcal{M}}$ have group orders of the form 4ℓ and $4\ell'$ with ℓ, ℓ' prime.

The search was implemented using the Magma computer algebra system [BCP97] with the Schoof-Elkies-Atkin (SEA) algorithm [Sch95] for efficient point counting. Starting from the smallest candidates $A \equiv 2 \pmod 4$, we verified that both curve and twist orders had the required structure. After determining valid curve parameters, we selected base points with minimal x-coordinates that generate the prime-order subgroup.

For *Curve5453* over $\mathbb{F}_{2^{545}-3}$, the smallest valid parameter is $A = 115794$ with base point x-coordinate 24. For *Curve6071* over $\mathbb{F}_{2^{607}-1}$, the smallest valid parameter is $A = 2720010$ with base point x-coordinate 11. This methodology follows established practices analogous to Curve25519 [Ber06] and Curve448 [Ham15], ensuring rigidity and full reproducibility of our selection process.

3 Designing Elliptic Curves for Cryptography

ECC security relies on the intractability of the ECDLP, with the fastest generic attacks requiring $O(\sqrt{\ell})$ operations [Pol78, HVM04]. However, curve choice must also address implementation risks such as weak torsion, incomplete formulas, and side-channel leakage [Koc96, BT11]. This section summarizes the SafeCurves criteria [BL14, BL25] and evaluates *Curve5453* and *Curve6071* for both mathematical strength and implementation robustness.

3.1 Parameter Security

We outline general security considerations for the finite fields and curve equations used in elliptic curve cryptography [HVM04, CFA+05]. ECC typically defines a prime p and an elliptic curve over $\mathbb{F}_p$, with coefficients in $\mathbb{F}_p$. Special-form primes, such as Mersenne and Crandall (pseudo-Mersenne) primes [Cra92, Ber06], enable fast modular arithmetic without introducing exploitable structure, whereas random primes offer no security gain

and often reduce efficiency. Primes with $p \equiv 3 \bmod 4$ are favored for efficient square-root computation and point decompression [HVM04], though other forms (e.g., $p \equiv 5 \bmod 8$) can also be efficient. Binary fields $\mathbb{F}_{2^n}$, once common in standards, are now largely avoided due to index-calculus attacks exploiting subfield structures [Jou13, BGJT14]; prime fields dominate modern ECC for their robustness and simplicity.

Curve representation also affects efficiency, implementation complexity, and side-channel resilience. The short Weierstrass form, $y^2 = x^3 + ax + b$, is standard for characteristic > 3 and ensures broad standards compatibility [HVM04]. Montgomery curves (Definition 1) [Mon87] allow efficient, constant-time scalar multiplication via the Montgomery ladder, making them well-suited for protocols like ECDH. Edwards curves, $x^2+y^2 = 1+dx^2y^2$ with $d(1-d) \neq 0$ [BBJ+08], can offer complete addition formulas when d is a non-square, simplifying side-channel protection and benefiting signature schemes. Choice of form depends on priorities: Montgomery for speed and side-channel security, Edwards for completeness, and short Weierstrass for interoperability.

3.2 ECDLP Security

The security of elliptic curves against the ECDLP depends on resisting a range of known attack strategies, each exploiting different mathematical or structural properties of the curve. The following key considerations outline the main approaches and the rationale for parameter choices that mitigate them.

The *rho method* targets the ECDLP by employing random walks to find collisions in the elliptic curve group. For a point P of prime order ℓ, it can recover x from $Q = [x]P$ in about $0.886\sqrt{\ell}$ additions, where $0.886 = \sqrt{\pi/4}$. The success probability after m additions is approximately m^2/ℓ, which motivates selecting a large ℓ to keep this probability negligible.

A *transfer* reduces the ECDLP to a discrete logarithm problem in either $\mathbb{F}_p^+$ or $\mathbb{F}_{p^k}^\times$. An *additive transfer* occurs when $\ell = p$, rendering the ECDLP trivial in $\mathbb{F}_p^+$. A *multiplicative transfer* arises when $\ell \mid p^k - 1$ for some k, reducing the problem to $\mathbb{F}_{p^k}^\times$ where subexponential index calculus methods may apply. To prevent such reductions from weakening security, curves are chosen with high embedding degrees, ensuring that any transfer still leads to computationally infeasible problems.

The *complex multiplication (CM) discriminant* D of $E/\mathbb{F}_p$, computed from the trace t and the largest square s^2 dividing $t^2 - 4p$, influences susceptibility to certain endomorphism-based speedups. If $(t^2 - 4p)/s^2 \equiv 1 \pmod 4$, then $D = (t^2 - 4p)/s^2$; otherwise, $D = 4(t^2 - 4p)/s^2$. Curves with small $|D|$ values may permit modest improvements to generic attacks like the rho method. While these do not constitute a full break, they can slightly erode security margins, so large $|D|$ values are preferred.

Finally, *rigidity* ensures trustworthy curve generation by using a deterministic, transparent process from minimal public input, preventing selection bias and enabling verifiable, manipulation-free parameters.

3.3 ECC Security

Beyond resistance to the ECDLP itself, the practical security of elliptic curve cryptography depends on structural properties of the curve and its implementation, which collectively ensure robustness against malicious inputs, exceptional cases, and distinguishability-based attacks.

One critical aspect is the *quadratic twist*. In x-only scalar multiplication, as in ECDH, protocols may process points from both the curve and its twist without validation. If the twist has small subgroups, an adversary can exploit this via maliciously crafted inputs. To prevent such attacks, both the curve and its twist must exhibit strong security properties, with large prime-order subgroups and no exploitable torsion components.

Completeness ensures addition and doubling formulas work for all valid inputs, avoiding exceptional cases that risk errors or side-channel leakage. Montgomery curves with the Montgomery ladder in ECDH achieve complete, constant-time arithmetic, but extending this to other protocols is harder, and optimizations like fixed-base precomputation may reintroduce exceptions.

Finally, *indistinguishability* ensures that group elements and protocol outputs are indistinguishable from random values, a property essential to censorship-resistant systems such as Tor [DMS04] and other obfuscation protocols. This property is typically achieved through encoding techniques such as Elligator [BHKL13], which maps field elements to curve points in a way that produces outputs indistinguishable from uniform random strings. It requires large prime-order subgroups without special structure, complete and uniform arithmetic, twist security in x-only settings, and rigid parameter generation to avoid hidden weaknesses.

3.4 Security Analysis of New Curves

The proposed *Curve5453* over $\mathbb{F}_{5453}$ and *Curve6071* over $\mathbb{F}_{6071}$ have been designed to meet stringent security and efficiency requirements. Both employ large, special-form primes—$2^{545} - 3$ (Crandall) and $2^{607} - 1$ (Mersenne)—that enable fast modular reduction while avoiding exploitable algebraic structure. The curves are given in Montgomery form with parameters satisfying $B(A^2-4) \neq 0$, ensuring a valid group law. Their group orders are $2^2 \cdot \ell$ with large prime ℓ, providing a prime-order subgroup and small cofactor to eliminate small-subgroup vulnerabilities. Parameter generation followed a fully deterministic and auditable process, ensuring rigidity and excluding hidden weaknesses.

From an ECDLP perspective, both curves provide large prime-order subgroups that make the best known generic attacks, such as Pollard's rho, computationally infeasible: approximately $2^{271.3}$ group operations for *Curve5453* and $2^{302.3}$ for *Curve6071*. The embedding degrees are sufficiently large to prevent multiplicative transfers: the embedding degree for *Curve5453* exceeds $(l-1)/100$, as does that of *Curve6071*, satisfying the SafeCurves requirement. Furthermore, $\ell \neq p$ precludes additive transfers. The CM field discriminants are exceptionally large: $|D| \approx 2^{545.33}$ for *Curve5453* and $|D| \approx 2^{607.32}$ for *Curve6071*. These values exceed the SafeCurves minimum requirement of 2^{100} by factors of approximately 2^{445} and 2^{507}, respectively, providing substantial margin against endomorphism-based speedups and preserving the intended security levels.

Beyond the hardness of the ECDLP, both curves exhibit strong properties for practical ECC security. Their quadratic twists have large prime-order subgroups and no small torsion components, ensuring twist security. This property is critical for x-only protocols such as ECDH, where the Montgomery ladder processes x-coordinates without distinguishing whether a point lies on the curve or its quadratic twist [Ber06]. If the twist contained small subgroups, an attacker could submit malicious x-coordinates that confine the computation to these weak subgroups, enabling key recovery through small-subgroup attacks [LL97]. Since both our curves and their twists have group orders of the form 4ℓ with large prime ℓ, the only small subgroup is the cofactor-4 component, which is insufficient to mount such attacks. Scalar multiplication is implemented using the constant-time, uniform Montgomery ladder, providing resilience against timing and cache-based side channels. *Curve6071* achieves completeness for all inputs, while *Curve5453* has a small set of exceptional points, which can be efficiently mitigated in software without impacting performance or security. Together, these features ensure that *Curve5453* and *Curve6071* meet the highest standards of modern elliptic curve cryptography. Table 1 presents the evaluation of the proposed curves against the SafeCurves criteria[1].

[1]Abbreviations in the table: *fi* = field, *eq* = equation, *ba* = base, *rh* = rho, *tr* = transfer, *di* = CM discriminant, *ri* = rigidity, *la* = ladder, *tw* = twist, *co* = completeness, *in* = indistinguishability.

Table 1: Proposed curves against SafeCurves criteria.

	parameter			**ecdlp**				**ecc**			
	fi	eq	ba	rh	tr	di	ri	la	tw	co	in
Curve5453	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓
Curve6071	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

3.5 Quantum Attack Resistance Analysis

While the proposed curves provide strong security against classical attacks, it is essential to assess their vulnerability to quantum adversaries. Shor's algorithm [Sho94] enables polynomial-time solution of the ECDLP on a sufficiently large quantum computer, rendering all elliptic curve cryptosystems—including our proposed curves—vulnerable once fault-tolerant quantum computing becomes practical.

To quantify the quantum resources required to break our curves, we apply the methodology of Häner et al. [HJN+20], who provided detailed estimates for attacking elliptic curve cryptography using Shor's algorithm. Their analysis accounts for the number of logical qubits, the depth of quantum circuits, and the overhead from quantum error correction, enabling a realistic assessment of the quantum threat timeline.

For comparison, breaking curves providing approximately 128 bits of classical security (such as Curve25519 or NIST P-256) requires approximately 2,124 logical qubits, while curves providing 192 and 256 bits of security require around 3,151 and 4,258 logical qubits, respectively [HJN+20]. Using this scaling, we estimate that *Curve5453*, providing approximately 271 bits of classical security, would require roughly 4,500—5,000 logical qubits to break. Similarly, *Curve6071*, providing approximately 302 bits of classical security, would require approximately 5,000—5,500 logical qubits.

Importantly, these estimates target the prime-order subgroup of order ℓ, which is the computational basis for Shor's algorithm [Sho94]. The cofactor-4 structure of our curves does not introduce additional quantum vulnerabilities: the small subgroup can leak at most 2 bits of information, which is negligible compared to the 271-bit or 302-bit secrets protected by the main subgroup. A quantum adversary would directly solve the ECDLP in the prime-order subgroup rather than exploit the cofactor, confirming that the group structure of our curves provides consistent quantum security.

These estimates place our curves in the range of medium-scale fault-tolerant quantum computers, requiring substantially more quantum resources than widely deployed curves. Like all ECC, these curves are not intended as post-quantum solutions, but rather for applications where enhanced classical security margins are valuable: hybrid cryptographic constructions combining classical ECC with post-quantum primitives, blockchain systems, and specialized deployments where implementation robustness takes precedence over broad standardization.

4 Field Arithmetic

The choice of the underlying field in elliptic curve cryptography plays a critical role in determining the efficiency and security of the resulting cryptographic protocol. While ECC can utilize curves defined over any sufficiently large field, primes with special algebraic structure facilitate faster arithmetic and more secure implementations.

4.1 Special Prime Forms

In ECC, the fastest field primes are *Mersenne primes* $p = 2^k - 1$, which allow reduction via a few bit shifts and additions, avoiding division entirely. Their alignment with word-sized arithmetic makes them ideal for performance, though they are rare. *Crandall primes* $p = 2^k - a$, with small a, generalize this idea, offering similar efficiency while allowing more flexible prime sizes. An example is Curve25519's $p = 2^{255} - 19$, optimized for 32-bit platforms. Unlike arbitrary primes that require general-purpose methods such as Barrett or Montgomery reduction, these special-form primes allow simple and fast modular reduction using only shifts and additions, with complexity linear in the number of limbs.

Both of our curves are defined over finite fields with special prime structures that enable efficient and secure arithmetic. The field of $E_{\mathcal{M}_{5453}}$ is defined by the Crandall-type prime $2^{545} - 3$ that supports fast modular reduction and constant-time implementation techniques due to the small offset from the power of two. The field of $E_{\mathcal{M}_{6071}}$ is defined by the Mersenne prime $2^{607} - 1$, which is particularly advantageous from a performance standpoint and aligns naturally with binary hardware arithmetic. Additionally, since $p \equiv 3 \pmod 4$, the field $\mathbb{F}_{6071}$ enables efficient square root extraction, facilitating operations such as point decompression and verification.

4.2 Representing Field Elements

For security, cryptographic systems use large prime fields, typically with $p > 2^{200}$. On n-bit architectures (commonly 32 or 64), field elements are stored as $t = \lceil k/n \rceil$ words for a k-bit prime p: $\alpha = (\alpha_{t-1}, \alpha_{t-2}, \ldots, \alpha_0)$, where α_{t-1} may be shorter than n bits. Exact n-bit words risk overflow during arithmetic, so a smaller word size $r < n$ is often chosen, leaving $n - r$ bits for carry handling. This *radix*-2^r representation expresses α as

$$\alpha(x) = \sum_{i=0}^{t-1} \alpha_i 2^{\lceil ri \rceil} x^i.$$

In our implementation, elements of $\mathbb{F}_{5453}$ and $\mathbb{F}_{6071}$ are represented on a 64-bit CPU using 10 limbs. The following subsections describe the specific limb decompositions used for each field.

4.2.1 Representing Elements of $\mathbb{F}_{5453}$:

Let $\alpha \in \mathbb{F}_{5453}$ with $k = 545$. One approach is to represent α in radix-2^{55} form by splitting it into nine limbs of 55 bits and one limb of 50 bits. The limb representation and the corresponding polynomial form are:

$$\alpha = [\,\alpha_0/55,\, \ldots,\, \alpha_8/55,\, \alpha_9/50\,], \tag{5}$$

$$\alpha(x) = \sum_{i=0}^{9} \alpha_i 2^{55i} x^i = \alpha_0 + \alpha_1 2^{55} x + \cdots + \alpha_9 2^{495} x^9. \tag{6}$$

An alternative is the radix-$2^{54.5}$ representation, where α is split into five 55-bit limbs and five 54-bit limbs. The limb representation and the corresponding polynomial form are:

$$\alpha = [\,\alpha_0/55,\, \alpha_1/54,\, \alpha_2/55,\, \ldots,\, \alpha_7/54,\, \alpha_8/55,\, \alpha_9/54\,], \tag{7}$$

$$\alpha(x) = \sum_{i=0}^{9} \alpha_i 2^{\lceil 54.5\, i \rceil} x^i = \alpha_0 + \alpha_1 2^{55} x + \alpha_2 2^{109} x^2 + \cdots + \alpha_9 2^{491} x^9. \tag{8}$$

Note that each coefficient α_i of polynomials given in (6) and (8) lies in the range $[-2^{55}, 2^{55}]$.

4.2.2 Representing Elements of $\mathbb{F}_{6071}$:

Let $\alpha \in \mathbb{F}_{6071}$ with $k = 607$. It is represented using nine limbs of 61 bits and one limb of 58 bits, known as the radix-2^{61} representation. The limb representation and its corresponding polynomial form are:

$$\alpha = [\ \alpha_0/61,\ \ldots,\ \alpha_8/61,\ \alpha_9/58\ ] \tag{9}$$
$$\alpha(x) = \sum_{i=0}^{9} \alpha_i 2^{\lceil 61i \rceil} x^i = \alpha_0 + \alpha_1 2^{61} x + \alpha_2 2^{122} x^2 + \ldots + \alpha_9 2^{549} x^9.$$

4.3 Arithmetic Operations

In this subsection, we use the notation $\alpha(x) = \alpha_0 + \alpha_1 x + \cdots + \alpha_{t-1} x^{t-1}$ and $\beta(x) = \beta_0 + \beta_1 x + \cdots + \beta_{t-1} x^{t-1}$ to denote elements of $\mathbb{F}_p$.

4.3.1 Addition and Subtraction:

Addition and subtraction in $\mathbb{F}_p$ are performed limb-wise with carry or borrow propagation. For addition, each limb sum $\gamma_i = \alpha_i + \beta_i + c$ is computed, updating the carry c if the result exceeds 2^r. After all t limbs, a final conditional subtraction of p ensures the result is reduced. Subtraction follows the same structure but tracks a borrow when a limb difference $\gamma_i = \alpha_i - \beta_i - b$ is negative, propagating it to the next limb. If the result is negative modulo p, a final conditional addition of p restores it to the canonical range. While the logic is symmetrical, addition ends with a conditional subtraction of p, whereas subtraction ends with a conditional addition of p.

4.3.2 Multiplication:

Let $\Gamma = \alpha \cdot \beta$, then $\Gamma(x)$ is represented as $\Gamma_0 + \Gamma_1 x + \ldots + \Gamma_{2t-1} x^{2t-1}$. For each limb α_i of α, it is multiplied by each limb β_j of β, and the partial product $\alpha_i \cdot \beta_j$ is accumulated into the corresponding position in the output array at index $i + j$. To accommodate carry propagation from lower to higher significance within each partial sum, a temporary variable ω_1 is used to store the carry, and ω_0 holds the lower n-bit portion of the sum:

$$(\omega_1, \omega_0) \leftarrow \Gamma_{i+j} + \alpha_i \cdot \beta_j + \omega_1,$$
$$\Gamma_{i+j} \leftarrow \omega_0.$$

The carry ω_1 is propagated to Γ_{i+t} at the end of the inner loop. Note that the output polynomial $\Gamma(x)$ must be reduced modulo p. To achieve this, the coefficients of terms with degree exceeding $t - 1$ in $\Gamma(x)$ are eliminated. The reduction procedure depends on the form of the prime p.

4.3.3 Reduction:

Reduction cost in field multiplication heavily influences ECC performance, making low-cost methods essential. Prime form is key: while Barrett and Montgomery reduction handle arbitrary primes, primes with special structure allow faster algorithms. This work employs a Crandall prime of the form $p = 2^k - a$ and a Mersenne prime (where $a = 1$), both of which allow simple and efficient modular reduction by folding higher limbs into lower ones using shifts and additions. Since the reduction algorithm naturally handles the case $a = 1$, the same approach is directly applicable to the Mersenne prime. The intermediate values ω_0' and ω_1' are $(k - r(t-1))$-bit integers, as defined in the field element representation section.

4.3.4 Inversion:

In $\mathbb{F}_p$, the inverse of $x \neq 0$ is the element x^{-1} satisfying $xx^{-1} \equiv 1 \pmod{p}$. Classical computation uses the extended Euclidean algorithm, while the binary inversion variant replaces costly divisions with shifts and subtractions. Montgomery's trick amortizes the cost when inverting multiple elements. Alternatively, Fermat's method computes $x^{p-2} \bmod p$ using fixed, constant-time exponentiation, which is especially efficient for (pseudo-)Mersenne primes when paired with optimized multiplication. For our field constructions, we compute $x^{p_{5453}-2}$ and $x^{p_{6071}-2}$ using optimized addition chains following the technique of [NS18], which minimizes the number of multiplications and squarings required for the exponentiation.

4.4 Efficient Multiplication with TMVP

A Toeplitz matrix has constant values along each descending diagonal, i.e., $T_{i,j} = t_{i-j}, \quad 1 \leq i, j \leq n$. This structure allows matrix-vector multiplication in $O(n \log n)$ using FFT-based methods, compared to $O(n^2)$ for the general case. TMVP exploits this property to accelerate modular multiplication by reformulating it as a Toeplitz matrix-vector multiplication, reducing the number of field operations. For size 2:

$$\begin{bmatrix} \alpha_0 & \alpha_1 \\ \alpha_2 & \alpha_0 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} P_1 + P_2 \\ P_1 + P_3 \end{bmatrix}, \tag{10}$$

where $P_1 = \alpha_0(\beta_0 + \beta_1)$, $P_2 = (\alpha_1 - \alpha_0)\beta_1$, and $P_3 = (\alpha_2 - \alpha_0)\beta_0$. The total cost is $3\mathbf{M} + 3\mathbf{A} + 2\mathbf{A}_d$[2], compared to $4\mathbf{M} + 2\mathbf{A}_d$ for the schoolbook method. For size 3:

$$\begin{bmatrix} \alpha_0 & \alpha_1 & \alpha_2 \\ \alpha_3 & \alpha_0 & \alpha_1 \\ \alpha_4 & \alpha_3 & \alpha_0 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} P_3 + P_4 + P_6 \\ P_2 - P_4 + P_5 \\ P_1 - P_2 - P_3 \end{bmatrix} \tag{11}$$

with $P_1 = (\alpha_4 + \alpha_3 + \alpha_0)\beta_0$, $P_2 = \alpha_3(\beta_0 - \beta_1)$, $P_3 = \alpha_0(\beta_0 - \beta_2)$, $P_4 = \alpha_1(\beta_1 - \beta_2)$, $P_5 = (\alpha_0 + \alpha_1 + \alpha_3)\beta_1$, and $P_6 = (\alpha_0 + \alpha_1 + \alpha_2)\beta_2$. This requires $6\mathbf{M} + 8\mathbf{A} + 6\mathbf{A}_d$ operations, compared to $9\mathbf{M} + 6\mathbf{A}_d$ for the schoolbook method.

To multiply two field elements via TMVP, let $\alpha, \beta \in \mathbb{F}_p$ be in radix-2^r form. Schoolbook multiplication modulo $x^t - 1$ can be written as a $t \times t$ Toeplitz matrix-vector product, enabling TMVP acceleration. Recursive composition of 2×2 (10) and 3×3 blocks (11) yields TMVPs of any size t.

For $t = 10$, the method of [Taş19] costs $77\mathbf{M} + 77\mathbf{A} + 109\mathbf{A}_d$ vs. $101\mathbf{M} + 92\mathbf{A}_d$ for schoolbook. A hybrid approach—TMVP for the 4×4 block and schoolbook for 2×2 blocks—costs $95\mathbf{M} + 23\mathbf{A} + 109\mathbf{A}_d$, offering a balance of efficiency and simplicity.

4.5 Multiplication Over $\mathbb{F}_{5453}$

Using the radix-2^{55} representation given in (5), field element multiplication over $\mathbb{F}_{5453}$ can be performed by the following matrix-vector multiplication

$$\begin{bmatrix} \alpha_0 & d\alpha_9 & d\alpha_8 & d\alpha_7 & d\alpha_6 & d\alpha_5 & d\alpha_4 & d\alpha_3 & d\alpha_2 & d\alpha_1 \\ \alpha_1 & \alpha_0 & d\alpha_9 & d\alpha_8 & d\alpha_7 & d\alpha_6 & d\alpha_5 & d\alpha_4 & d\alpha_3 & d\alpha_2 \\ \alpha_2 & \alpha_1 & \alpha_0 & d\alpha_9 & d\alpha_8 & d\alpha_7 & d\alpha_6 & d\alpha_5 & d\alpha_4 & d\alpha_3 \\ \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & d\alpha_9 & d\alpha_8 & d\alpha_7 & d\alpha_6 & d\alpha_5 & d\alpha_4 \\ \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & d\alpha_9 & d\alpha_8 & d\alpha_7 & d\alpha_6 & d\alpha_5 \\ \alpha_5 & \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & d\alpha_9 & d\alpha_8 & d\alpha_7 & d\alpha_6 \\ \alpha_6 & \alpha_5 & \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & d\alpha_9 & d\alpha_8 & d\alpha_7 \\ \alpha_7 & \alpha_6 & \alpha_5 & \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & d\alpha_9 & d\alpha_8 \\ \alpha_8 & \alpha_7 & \alpha_6 & \alpha_5 & \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & d\alpha_9 \\ \alpha_9 & \alpha_8 & \alpha_7 & \alpha_6 & \alpha_5 & \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 \end{bmatrix} \cdot \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \\ \beta_6 \\ \beta_7 \\ \beta_8 \\ \beta_9 \end{bmatrix} = \begin{bmatrix} \gamma_0 \\ \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \\ \gamma_5 \\ \gamma_6 \\ \gamma_7 \\ \gamma_8 \\ \gamma_9 \end{bmatrix}, \tag{12}$$

[2] $\mathbf{M}$, $\mathbf{A}$, and $\mathbf{A}_d$ denote the cost of multiplication, addition, and double-precision addition, respectively.

where $d = 96$. Modular reduction in the radix-2^{55} representation is carried out with respect to the modulus $2^{550}x^{10} - 96$. Furthermore, the structure of the radix-2^{55} representation in $\mathbb{F}_{5453}$ gives rise to a Toeplitz matrix, which can be decomposed into upper and lower triangular matrices as follows

$$\begin{bmatrix} \alpha_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha_1 & \alpha_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha_2 & \alpha_1 & \alpha_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & 0 & 0 & 0 & 0 & 0 \\ \alpha_5 & \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & 0 & 0 & 0 & 0 \\ \alpha_6 & \alpha_5 & \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & 0 & 0 & 0 \\ \alpha_7 & \alpha_6 & \alpha_5 & \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & 0 & 0 \\ \alpha_8 & \alpha_7 & \alpha_6 & \alpha_5 & \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & 0 \\ \alpha_9 & \alpha_8 & \alpha_7 & \alpha_6 & \alpha_5 & \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 \end{bmatrix} + d \cdot \begin{bmatrix} 0 & \alpha_9 & \alpha_8 & \alpha_7 & \alpha_6 & \alpha_5 & \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 \\ 0 & 0 & \alpha_9 & \alpha_8 & \alpha_7 & \alpha_6 & \alpha_5 & \alpha_4 & \alpha_3 & \alpha_2 \\ 0 & 0 & 0 & \alpha_9 & \alpha_8 & \alpha_7 & \alpha_6 & \alpha_5 & \alpha_4 & \alpha_3 \\ 0 & 0 & 0 & 0 & \alpha_9 & \alpha_8 & \alpha_7 & \alpha_6 & \alpha_5 & \alpha_4 \\ 0 & 0 & 0 & 0 & 0 & \alpha_9 & \alpha_8 & \alpha_7 & \alpha_6 & \alpha_5 \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha_9 & \alpha_8 & \alpha_7 & \alpha_6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha_9 & \alpha_8 & \alpha_7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha_9 & \alpha_8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha_9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{13}$$

This decomposition enables the application of the 10-dimensional TMVP technique proposed in [Taş19]. As a result, the multiplication by the constant 96 can be deferred and performed as a final step after the main computation. Since $96 = 2^6 + 2^5$, the operation $x \cdot 96$ can be efficiently realized using two shift operations and a single addition: $x \cdot 96 = (x \ll 6) + (x \ll 5)$.

In addition, the radix-$2^{54.5}$ representation of elements in $\mathbb{F}_{5453}$, as given in (7), exhibits a similar matrix structure to the radix-$2^{25.5}$ representation used for Curve25519 on 32-bit architectures [Ber06].

$$\begin{bmatrix} \alpha_0 & 6\alpha_9 & 3\alpha_8 & 6\alpha_7 & 3\alpha_6 & 6\alpha_5 & 3\alpha_4 & 6\alpha_3 & 3\alpha_2 & 6\alpha_1 \\ \alpha_1 & \alpha_0 & 3\alpha_9 & 3\alpha_8 & 3\alpha_7 & 3\alpha_6 & 3\alpha_5 & 3\alpha_4 & 3\alpha_3 & 3\alpha_2 \\ \alpha_2 & 2\alpha_1 & \alpha_0 & 6\alpha_9 & 3\alpha_8 & 3\alpha_7 & 3\alpha_6 & 6\alpha_5 & 3\alpha_4 & 6\alpha_3 \\ \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & 3\alpha_9 & 3\alpha_8 & 3\alpha_7 & 3\alpha_6 & 3\alpha_5 & 3\alpha_4 \\ \alpha_4 & 2\alpha_3 & \alpha_2 & 2\alpha_1 & \alpha_0 & 6\alpha_9 & 3\alpha_8 & 6\alpha_7 & 3\alpha_6 & 6\alpha_5 \\ \alpha_5 & \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & 3\alpha_9 & 3\alpha_8 & 3\alpha_7 & 3\alpha_6 \\ \alpha_6 & 2\alpha_5 & \alpha_4 & 2\alpha_3 & \alpha_2 & 2\alpha_1 & \alpha_0 & 6\alpha_9 & 3\alpha_8 & 6\alpha_7 \\ \alpha_7 & \alpha_6 & \alpha_5 & \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & 3\alpha_9 & 3\alpha_8 \\ \alpha_8 & 2\alpha_7 & \alpha_6 & 2\alpha_5 & \alpha_4 & 2\alpha_3 & \alpha_2 & 2\alpha_1 & \alpha_0 & 6\alpha_9 \\ \alpha_9 & \alpha_8 & \alpha_7 & \alpha_6 & \alpha_5 & \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 \end{bmatrix} \cdot \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \\ \beta_6 \\ \beta_7 \\ \beta_8 \\ \beta_9 \end{bmatrix} = \begin{bmatrix} \gamma_0 \\ \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \\ \gamma_5 \\ \gamma_6 \\ \gamma_7 \\ \gamma_8 \\ \gamma_9 \end{bmatrix}. \tag{14}$$

This structure naturally supports a matrix-vector multiplication scheme suitable for efficient implementation on 64-bit systems. The modular reduction in this representation is performed with respect to the modulus $2^{545}x^{10} - 3$.

4.6 Multiplication Over $\mathbb{F}_{6071}$

To multiply over $\mathbb{F}_{6071}$ in the radix-2^{61} form of (9), use the matrix-vector multiplication in (12) with $d = 8$. Modular reduction in the radix-2^{61} representation is carried out with respect to the modulus $2^{610}x^{10} - 8$. In $\mathbb{F}_{6071}$, the radix-2^{61} representation yields a Toeplitz matrix, decomposable into upper and lower triangular matrices as in (13) with $d = 8$. This decomposition also facilitates the application of the 10-dimensional TMVP technique for $\mathbb{F}_{6071}$. Consequently, the multiplication by the constant 8 can be postponed and executed as a final step following the main computation. Since $8 = 2^3$, this operation can be efficiently implemented using a single bit shift: $x \cdot 8 = (x \ll 3)$.

5 Implementation

We implemented field arithmetic for $\mathbb{F}_{5453}$ and $\mathbb{F}_{6071}$ using the limb representations in (5) and (9), respectively. All code is written in C, based on the implementation provided in [BL09], and is publicly available (see Data Availability).

Five field multiplication methods were benchmarked: **ref10** and **donna** (ported from Curve25519 [Ber06] via SUPERCOP [BL09]), **kummer** [Taş19], and two TMVP variants—**tmvp**, applying TMVP to both 4×4 and 2×2 blocks, and **tmvpsb**, applying TMVP only to 4×4 with schoolbook for 2×2. For *Curve5453*, we also implemented **ref10** for the radix-$2^{54.5}$ representation in (7), corresponding to the matrix-vector product in (14), denoted $\textbf{ref10}_{54.5}$.

Benchmarks were run on a MacBook Pro (Apple M1 Pro, 16 GB RAM, macOS Sequoia 15.5, Apple Clang 17.0.0, `arm64-apple-darwin24.5.0`) with flags `-ffast-math` and `-fno-common`. Cycle counts are measured using `mach_absolute_time` converted to CPU cycles at 3.228 GHz. Each benchmark includes a warm-up phase of 2.5×10^5 iterations, followed by 4 measurement rounds of 10^6 iterations each; the minimum cycle count across rounds is reported. Fixed seeds ensure identical inputs across tests. Multiplication benchmarks computed the product of two field elements; inversion used Fermat's method $x^{p-2} \bmod p$. Scalar multiplication (Algorithm 1) was benchmarked for both fixed-base and variable-base cases, each implemented with the Montgomery ladder followed by a single field inversion. Table 2 reports field multiplication and inversion over $\mathbb{F}_{5453}$, along with fixed-base and variable-base scalar multiplication on $E_{\mathcal{M}_{5453}}$. The row $\textbf{ref10}_{54.5}$ denotes the **ref10** implementation using the radix-$2^{54.5}$ representation; all other entries use radix-2^{55}. Table 3 presents corresponding results for $\mathbb{F}_{6071}$ and $E_{\mathcal{M}_{6071}}$.

Table 2: CPU cycle counts for various multiplication methods: field multiplication and inversion over $\mathbb{F}_{5453}$, and scalar multiplication (with base point and arbitrary point) on $E_{\mathcal{M}_{5453}}$.

	`mul`	`inv`	`smul_base`	`smul`
donna	152.29	83,779.15	944,402.91	932,983.64
kummer	356.51	207,576.59	2,120,114.83	2,121,771.17
ref10	211.25	120,814.07	1,309,940.94	1,307,011.34
tmvp	**134.02**	79,382.54	**871,188.16**	**871,897.82**
tmvpsb	157.92	84,101.7	934,246.49	933,228.38
$\textbf{ref10}_{54.5}$	138.82	**78,250.35**	900,358.33	914,670.36

Table 3: CPU cycle counts for various multiplication methods: field multiplication and inversion over $\mathbb{F}_{6071}$, and scalar multiplication (with base point and arbitrary point) on $E_{\mathcal{M}_{6071}}$.

	`mul`	`inv`	`smul_base`	`smul`
donna	303.03	90,634.65	1,006,570.01	1,025,239.84
kummer	671.14	224,255.59	2,289,449.28	2,263,124.97
ref10	449.92	147,543.32	1,560,313.77	1,557,261.87
tmvp	**241.09**	**79,807.78**	**891,822.74**	**895,027.51**
tmvpsb	267.30	87,824.05	958,202.55	959,635.47

The main difference between **donna** and **ref10** lies in constant multiplication during limb operations: **ref10** multiplies each limb by reduction constants inline during the carry propagation phase, while **donna** defers this step to the final reduction, enabling cheaper operations through shifts and additions (2 shifts + 1 addition for $\mathbb{F}_{5453}$, 1 shift for $\mathbb{F}_{6071}$). This architectural choice makes **donna** consistently faster than **ref10** across both curves, achieving 27.9% better field multiplication performance for $\mathbb{F}_{5453}$ and 32.7% for $\mathbb{F}_{6071}$.

The **kummer** implementation exhibited significantly poorer performance, with scalar multiplication approximately 2.4× to 2.5× slower than our best **tmvp** implementation. While Kummer surface arithmetic [Gau07] can reduce the number of field multiplications per scalar multiplication step, its benefits are most pronounced for smaller field sizes where this reduction offsets the overhead of additional coordinate operations. At 545—607 bit field sizes, the relative cost of multiplications decreases compared to this fixed overhead, making the standard Montgomery ladder more efficient; consequently, we exclude **kummer** from further comparison.

Across all tested methods, **tmvp** delivers the best performance for field multiplication and scalar multiplication on both curves, making it the best overall choice for elliptic curve operations. For $\mathbb{F}_{5453}$, the radix-$2^{54.5}$ implementation $\mathbf{ref10}_{54.5}$ achieves slightly better inversion performance (78,250.35 cycles versus 79,382.54 for **tmvp**) despite slower multiplication, which may be attributed to its alternating limb structure offering benefits when operations are chained together in the inversion algorithm. We use **donna** as the baseline for comparison since it represents the current state-of-the-art for Montgomery curve arithmetic, being the most efficient among the classical approaches (**ref10** and **donna**) ported from Curve25519. Compared to this baseline, **tmvp** achieves a 12.0% improvement in field multiplication and 6.5% in scalar multiplication for $\mathbb{F}_{5453}$; for $\mathbb{F}_{6071}$, the improvements are even more substantial at 20.4% and 12.7% respectively. The **tmvpsb** variant, which applies TMVP only to 4×4 blocks and uses schoolbook multiplication for 2×2 blocks, also demonstrates strong performance, outperforming **donna** by 11.8% for $\mathbb{F}_{6071}$ multiplication while being 3.7% slower for $\mathbb{F}_{5453}$ multiplication.

We also benchmarked *Curve25519* on the same platform using the **ref10** implementation for comparison. Table 4 presents a comprehensive comparison of our best implementations (**tmvp**) against Curve25519, showing field size, security level, and performance metrics.

Table 4: Performance comparison (CPU cycle counts) of *Curve25519* with *Curve5453* and *Curve6071* (using **tmvp** implementation).

	Field size	Classical security	`mul`	`inv`	`smul`
Curve25519	255	128	105.51	18,288.51	247,124.00
Curve5453	545	271	134.02	79,382.54	871,897.82
(overhead)			(1.27×)	(4.34×)	(3.53×)
Curve6071	607	302	241.09	79,807.78	895,027.51
(overhead)			(2.29×)	(4.36×)	(3.62×)

Despite operating on fields approximately 2× larger (545 and 607 bits versus 255 bits), our curves achieve field multiplication with only 1.27× and 2.29× the cost of *Curve25519* for $\mathbb{F}_{5453}$ and $\mathbb{F}_{6071}$ respectively, demonstrating efficient scaling. The field inversion cost is approximately 4.3× higher for both curves, which is expected due to the larger exponents $(p-2)$ requiring more squaring and multiplication operations in Fermat's method. Note that for $\mathbb{F}_{5453}$, the radix-$2^{54.5}$ implementation $\mathbf{ref10}_{54.5}$ achieves slightly better inversion performance (78,250.35 cycles, corresponding to 4.28× overhead) than **tmvp** while being less efficient for multiplication and scalar multiplication overall. Scalar multiplication overhead is approximately 3.5×—3.6×, reflecting the combined impact of larger field operations across the Montgomery ladder iterations. This represents a favorable trade-off: more than 2× the security level of *Curve25519* with modest computational overhead, maintaining practical efficiency for cryptographic applications while providing substantially enhanced security margins.

6 Comparative Assessment

The landscape of elliptic curves offering security beyond 200 bits reveals a notable gap between theoretical proposals and practical deployment. While the SafeCurves project [BL14] has established rigorous criteria for implementation security, and several curves meeting these criteria have been proposed for high security levels, none have achieved significant adoption in industry applications such as blockchain systems or specialized security deployments.

Among SafeCurves-approved alternatives, the most relevant comparisons for our work are M-511 and E-521. M-511, proposed by Aranha et al. [ABPR13], is a Montgomery curve over $\mathbb{F}_{2^{511}-187}$ providing approximately 254 bits of security. E-521, an Edwards curve over the Mersenne prime $2^{521}-1$ [GS15], offers approximately 259 bits of security. Ed448-Goldilocks [Ham15], while providing only 224 bits of security, has gained traction through IETF standardization for TLS. However, published benchmarks for M-511 on modern platforms remain scarce, and none of these curves have been deployed in blockchain or similar non-standardized contexts where curves like secp256k1 dominate despite failing SafeCurves criteria.

Our proposed curves address this gap by combining enhanced security levels with SafeCurves compliance and practical efficiency. *Curve5453* provides approximately 271 bits of security, exceeding both M-511 and E-521, while *Curve6071* offers approximately 302 bits-the highest among any SafeCurves-compliant curve to date. Importantly, both curves are defined over primes with superior arithmetic properties: $2^{607}-1$ is a Mersenne prime enabling optimal modular reduction, and $2^{545}-3$ is a minimal Crandall prime with offset significantly smaller than M-511's offset of 187. These prime structures directly translate to implementation efficiency, as demonstrated by our TMVP-optimized arithmetic.

The standardized alternatives at comparable security levels—NIST P-521 and Brainpool P-512—fail multiple SafeCurves criteria including twist security, completeness, and constant-time ladder support. While these curves may be acceptable for applications where standardization is paramount, they introduce implementation risks that our Montgomery-form curves avoid through inherent support for constant-time x-only arithmetic.

Our benchmarks on ARM64 demonstrate that enhanced security need not come at prohibitive computational cost. Table 5 presents scalar multiplication performance for high-security curves across different platforms.

Table 5: Scalar multiplication performance comparison for high-security elliptic curves.

	Classical security	Platform	`smul`(cycles)
Curve5453	271	ARM64 (M1 Pro)	871,898
Curve6071	302	ARM64 (M1 Pro)	895,028
E-521 [GS15]	259	Intel Haswell	943,000
Ed448 [Ham15]	224	Intel Haswell	803,000

Although direct comparison across platforms is not possible, our results demonstrate competitive performance: our curves achieve scalar multiplication in roughly comparable cycle counts to E-521 and Ed448 while providing 271-302 bits of security—significantly higher than E-521's 259 bits and Ed448's 224 bits. This indicates that curves at the highest security levels can achieve practical performance on modern 64-bit architectures.

The combination of SafeCurves compliance, favorable prime structure, and demonstrated efficiency positions *Curve5453* and *Curve6071* as candidates for applications where non-standardized curves are acceptable. These include hybrid cryptographic constructions combining classical ECC with post-quantum primitives, private blockchain deployments

requiring enhanced security margins, and specialized systems where implementation robustness takes precedence over broad interoperability.

7 Conclusion

This work introduced *Curve5453* and *Curve6071*, two Montgomery-form elliptic curves defined over a Crandall prime $(2^{545} - 3)$ and the Mersenne prime $(2^{607} - 1)$, respectively. These curves provide security levels of approximately 271 and 302 bits—substantially exceeding existing alternatives—while satisfying most SafeCurves criteria, with detailed analysis provided in Section 3.

We developed TMVP-based field multiplication optimized for 10-limb representations on 64-bit architectures, achieving speedups of 12.0% for *Curve5453* and 20.4% for *Curve6071* over the state-of-the-art **donna** implementation. Our ARM64 benchmarks demonstrate practical efficiency at these higher security levels, with scalar multiplication requiring 871,898 and 895,028 cycles, respectively—competitive with existing lower-security alternatives while providing significantly enhanced security margins.

The proposed curves address a gap in the current landscape where no SafeCurves-compliant high-security curves have achieved practical deployment. While not quantum-resistant and not intended to replace post-quantum standards, these curves address deployment requirements where enhanced classical security, implementation robustness, and computational efficiency must be achieved simultaneously. They are particularly well-suited for hybrid post-quantum constructions requiring classical components with security levels commensurate with quantum-resistant primitives, blockchain systems where cryptographic operations dominate performance and security requirements span decades, and specialized deployments with extended operational lifetimes or high-value communications. By combining substantially higher security levels than current standards with verified implementation properties and competitive performance, the proposed curves provide practical alternatives for applications where these specific characteristics are critical.

References

[ABPR13] Diego F. Aranha, Paulo S. L. M. Barreto, Geovandro C. C. F. Pereira, and Jefferson E. Ricardini. A note on high-security general-purpose elliptic curves. Cryptology ePrint Archive, Paper 2013/647, 2013. https://eprint.iacr.org/2013/647.

[AC18] Shoukat Ali and Murat Cenk. Faster residue multiplication modulo 521-bit mersenne prime and an application to ecc. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(8):2477–2490, 2018.

[BBJ+08] Daniel J. Bernstein, Peter Birkner, Marc Joye, Tanja Lange, and Christiane Peters. Twisted edwards curves. *AFRICACRYPT*, 5023:389–405, 2008.

[BCP97] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system I: The user language. *Journal of Symbolic Computation*, 24(3-4):235–265, 1997.

[Ber06] Daniel J. Bernstein. Curve25519: New diffie-hellman speed records. In *PKC*, volume 3958 of *LNCS*, pages 207–228. Springer, 2006.

[Ber08] Daniel J. Bernstein. qhasm: tools to help write fast crypto. https://cr.yp.to/qhasm.html, 2008.

[BGJT14] Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. Breaking the FF3 cryptosystem in polynomial time. In *Annual Cryptology Conference*, pages 55–71. Springer, 2014.

[BHKL13] Daniel J Bernstein, Mike Hamburg, Anna Krasnova, and Tanja Lange. Elligator: Elliptic-curve points indistinguishable from uniform random strings. In *ACM Conference on Computer and Communications Security*, pages 967–980. ACM, 2013.

[BL09] Daniel J Bernstein and Tanja Lange. Supercop: System for unified performance evaluation related to cryptographic operations and primitives, 2009.

[BL14] Daniel J. Bernstein and Tanja Lange. Safecurves: Choosing safecurves for elliptic-curve cryptography. https://safecurves.cr.yp.to, 2014. Accessed: 2025-07-22.

[BL25] Daniel J Bernstein and Tanja Lange. Safe curves for elliptic-curve cryptography. *Information Security in a Connected World: Celebrating the Life and Work of Ed Dawson*, pages 124–191, 2025.

[Bra05] ECC Brainpool. Ecc brainpool standard curves and curve generation, 2005.

[BT11] Billy Bob Brumley and Nicola Tuveri. Remote timing attacks are still practical. In *European Symposium on Research in Computer Security*, pages 355–371. Springer, 2011.

[CFA+05] Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren. *Handbook of elliptic and hyperelliptic curve cryptography*. CRC press, 2005.

[CL15] Craig Costello and Patrick Longa. Four q: Four-dimensional decompositions on a q-curve over the mersenne prime. In *ASIACRYPT*, volume 9452 of *LNCS*, pages 214–235. Springer, 2015.

[Clo19] Cloudflare. The TLS post-quantum experiment. https://blog.cloudflare.com/the-tls-post-quantum-experiment/, 2019. Accessed: 2025.

[Cra92] Richard E Crandall. Method and apparatus for public key exchange in a cryptographic system. *US Patent 5,159,632*, 1992.

[CS18] Craig Costello and Benjamin Smith. Montgomery curves and their arithmetic: The case of large characteristic fields. *Journal of Cryptographic Engineering*, 8(3):227–240, 2018.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *13th USENIX Security Symposium*, pages 303–320, 2004.

[Gau07] Pierrick Gaudry. Fast genus 2 arithmetic based on theta functions. *Journal of Mathematical Cryptology*, 1(3):243–265, 2007.

[GCRH23] Alexandre Augusto Giron, Ricardo Custódio, and Francisco Rodríguez-Henríquez. Post-quantum hybrid key exchange: a systematic mapping study. *Journal of Cryptographic Engineering*, 13:71–88, 2023.

[Goo24] Google Cloud. Why Google now uses post-quantum cryptography for internal comms. https://cloud.google.com/blog/products/identity-security/why-google-now-uses-post-quantum-cryptography-for-internal-comms, 2024. Accessed: 2025.

[GS15] Robert Granger and Michael Scott. Faster ecc over f2521-1. In Jonathan Katz, editor, *Public-Key Cryptography – PKC 2015*, pages 539–553, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[Ham15] Mike Hamburg. Ed448-goldilocks, a new elliptic curve. Cryptology ePrint Archive, Paper 2015/625, 2015. https://eprint.iacr.org/2015/625.

[HJN+20] Thomas Häner, Samuel Jaques, Michael Naehrig, Martin Roetteler, and Mathias Soeken. Improved quantum circuits for elliptic curve discrete logarithms. In *International conference on post-quantum cryptography*, pages 425–444. Springer, 2020.

[HVM04] Darrel Hankerson, Scott Vanstone, and Alfred Menezes. *Guide to Elliptic Curve Cryptography*. Springer, 2004.

[HWCD08] Huseyin Hisil, Kenneth Koon-Ho Wong, Gary Carter, and Ed Dawson. Twisted edwards curves revisited. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 326–343. Springer, 2008.

[IEE00] IEEE. Ieee standard specifications for public-key cryptography. *IEEE Std 1363-2000*, pages 1–228, 2000.

[Jou13] Antoine Joux. A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. *Advances in Cryptology–EUROCRYPT 2013*, pages 1–16, 2013.

[Kob87] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[Koc96] Paul C Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Advances in Cryptology—CRYPTO'96*, pages 104–113. Springer, 1996.

[LL97] Chae Hoon Lim and Pil Joong Lee. A key recovery attack on discrete log-based schemes using a prime order subgroup. In *Advances in Cryptology—CRYPTO'97*, pages 249–263. Springer, 1997.

[Mil86] Victor S. Miller. Use of elliptic curves in cryptography. In Hugh C. Williams, editor, *Advances in Cryptology — CRYPTO '85 Proceedings*, pages 417–426, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.

[Mon87] Peter L Montgomery. Speeding the pollard and elliptic curve methods of factorization. *Mathematics of computation*, 48(177):243–264, 1987.

[Nat00] National Institute of Standards and Technology (NIST). Digital Signature Standard (DSS). Federal Information Processing Standards Publication FIPS PUB 186-2, National Institute of Standards and Technology (NIST), January 2000.

[Nat05] National Security Agency (NSA). Suite B cryptography. NSA Fact Sheet, 2005. Fort Meade, Maryland.

[Nat23] National Institute of Standards and Technology (NIST). Digital Signature Standard (DSS). Federal Information Processing Standards Publication FIPS PUB 186-5, National Institute of Standards and Technology (NIST), February 2023.

[Nat25] National Institute of Standards and Technology (NIST). Post-quantum cryptography standardization project, 2025. Accessed: 2025-08-07.

[NS18] Kaushik Nath and Palash Sarkar. Efficient arithmetic in (pseudo-) mersenne prime order fields. *Cryptology ePrint Archive*, 2018.

[Pol78] John M Pollard. Monte carlo methods for index computation mod p. *Mathematics of Computation*, 32(143):918–924, 1978.

[Sch95] René Schoof. Counting points on elliptic curves over finite fields. *Journal de théorie des nombres de Bordeaux*, 7(1):219–254, 1995.

[SEC00] SECG SEC. 2: Recommended elliptic curve domain parameters. *Standards for Efficient Cryptography Group, Certicom Corp*, 5, 2000.

[Sho94] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[SM19] Douglas Stebila and Michele Mosca. Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH. In *NIST Second PQC Standardization Conference*, 2019.

[Taş19] Halil Kemal Taşkın. *Efficient implementation of TMVP-based prime field multiplication and its applications to ecc.* PhD thesis, Middle East Technical University (Turkey), 2019.

[TC20] Halil Kemal Taşkın and Murat Cenk. Tmvp-friendly primes for efficient elliptic curve cryptography. In *2020 International Conference on Information Security and Cryptology (ISCTURKEY)*, pages 80–87. IEEE, 2020.