A Universal Weierstrass Engine for Hardware-Supported Quantum Cryptanalysis of RSA, DH, and ECC

Michał Wroński[1][0000-0002-8679-9399], Łukasz Dzierzkowski[2][0000-0002-9204-4558], Mateusz Leśniak[1][0009-0001-0092-2975], and Ewa Syta[3][0000-0003-0860-0927]

[1] NASK – National Research Institute, Warsaw, Poland
michal.wronski@nask.pl
[2] Military University of Technology, Warsaw, Poland
[3] Trinity College, Hartford, CT, USA

Abstract. Quantum cryptanalysis of deployed public-key cryptography is usually translated into problem-specific Shor engines: modular exponentiation for RSA and finite-field Diffie–Hellman, and reversible point arithmetic for elliptic-curve cryptography. This fragmentation complicates hardware-supported benchmarking, verification, and reconfiguration of cryptanalytic experiments. We show that, in an explicit semi-agnostic model, the three standard targets—integer factoring, finite-field discrete logarithms, and elliptic-curve discrete logarithms—can be represented through a single programmable Weierstrass arithmetic engine.

The algebraic core is an explicit *semi-agnostic Weierstrass/ECDLP oracle*, restricted to the instance families used by the construction: smooth elliptic curves over finite fields, the split nodal cubic $y^2 = x^3 + x^2$ over finite fields of characteristic greater than three, and the same nodal cubic over rings $\mathbb{Z}_N$ with $\gcd(N, 6) = 1$. We prove a polynomial-time equivalence between this explicit oracle and the standard abelian hidden subgroup problems (SAHSPs) arising from factoring, finite-field DLP, and ECDLP. The finite-field reduction uses the fact that the nonsingular locus of the nodal cubic is isomorphic to $\mathbb{G}_m$; over $\mathbb{Z}_N$ we use a projective form of the same parametrization, while affine denominators provide only a classical factoring-by-failure interface.

From the hardware-supported cryptanalytic perspective, the result identifies a universal Shor-type arithmetic backend: a reversible field/ring-arithmetic and Weierstrass-addition block configured by problem parameters. The backend has the same leading estimate as a worst-case elliptic-curve point-addition allocation and the same asymptotic order as the usual specialized designs, while avoiding separate RSA/DH and ECC arithmetic pipelines.

Keywords: quantum cryptanalysis · Shor's algorithm · elliptic-curve discrete logarithm problem · hardware-supported cryptanalysis · universal quantum circuits · Abelian hidden subgroup problem

1 Introduction

Modern software and network infrastructures still rely on classical public-key primitives whose security is based on integer factorization (RSA [11]), the discrete logarithm problem in finite fields (Diffie–Hellman [3]), and the elliptic-curve discrete logarithm problem (ECC [8,6]). These assumptions are mathematically different in the classical setting, but all of them become standard instances of the abelian hidden subgroup problem (AHSP) in the quantum setting.

Shor's algorithm [13] solves all of these standard AHSPs in polynomial time. Existing quantum cryptanalytic implementations, however, are usually engineered as separate attack pipelines: RSA and finite-field DLP rely on modular exponentiation, whereas ECDLP relies on reversible point addition. For hardware-supported quantum cryptanalysis this fragmentation is undesirable. A platform used to benchmark, validate, or reconfigure Shor-type attacks on software-deployed public-key cryptography must either maintain separate arithmetic engines or recompile the arithmetic layer when the target family changes.

This paper keeps the cryptanalytic viewpoint but asks whether these attacks really require three different arithmetic back ends. We show that they do not. A single explicit Weierstrass oracle, denoted by $\mathcal{O}_W$ below, covers the standard cryptanalytic targets considered here: integer factoring, finite-field DLP, and ordinary finite-field ECDLP. Consequently, a single programmable Weierstrass arithmetic engine can implement Shor-type attacks on RSA, finite-field DH, and ECDH.

1.1 Hardware-supported cryptanalytic setting

The intended setting is a configurable cryptanalytic engine. The software side specifies the target family and public parameters, such as N for RSA, p, g, h for finite-field DLP, or a curve and points E, P, Q for ECDLP. The hardware side implements one reversible arithmetic schedule based on field or ring arithmetic and Weierstrass point addition. Switching between RSA, DH, and ECC then changes the loaded parameters and precomputed constants rather than the underlying arithmetic pipeline.

This is not a defensive co-processor in the usual sense. It is a hardware-supported cryptanalytic engine for evaluating the quantum vulnerability of public-key mechanisms used by software and network protocols. The contribution is therefore aligned with hardware/software security from the attack and assessment side: the work reduces the fragmentation of quantum cryptanalytic hardware for RSA, DH, and ECC.

1.2 Algebraic unification

We show that algorithms may operate directly on explicit Weierstrass models, either smooth or nodal, defined over fields or rings, without relying on knowledge of the group order. Over finite fields of characteristic greater than three, the nodal cubic realizes $\mathbb{F}_q^*$ inside the nonsingular locus of a Weierstrass cubic; over

$\mathbb{Z}_N$, a projective parametrization realizes $\mathbb{Z}_N^*$, while affine non-unit denominators may reveal factors of N only at the classical reduction interface. Thus every standard cryptanalytic AHSP instance considered in this paper reduces to the explicit semi-agnostic Weierstrass oracle formalized below.

More precisely, for the explicit oracle domain used by our construction, we prove

$$\text{SAHSP} \ =_T^P \ \mathcal{O}_W,$$

where $\mathcal{O}_W$ is the restricted Weierstrass/ECDLP oracle defined in Section 5, and $A =_T^P B$ means that each problem can be reduced to the other in polynomial time by a Turing reduction. This statement should not be read as an equivalence with a finite-field-DLP-only backend.

1.3 Contributions

From the viewpoint of hardware-supported cryptanalysis, the paper contributes: (i) a single reversible Weierstrass arithmetic block for Shor-type attacks on RSA, finite-field DH, and ECDH; (ii) a polynomial-time equivalence between SAHSP and the explicit Weierstrass oracle $\mathcal{O}_W$ used by the construction; (iii) nodal-cubic reductions, including a projective $\mathbb{Z}_N$ parametrization with a classical affine factoring-by-failure interface; and (iv) a resource comparison with recompilation and dynamic-switching baselines.

2 Related Work

Two lines of prior work are most relevant. First, classical reductions and pairing-based embeddings give partial connections among standard AHSPs. Field-restriction reductions move DLP or ECDLP from $\mathbb{F}_p$ to $\mathbb{F}_q$ when $q = p^k$. MOV-type reductions embed ECDLP into finite-field DLP for curves with small embedding degree [7]; for generic curves the required extension may be too large, and no comparable generic reduction from finite-field DLP to smooth-curve ECDLP is known. Bach's reductions relate factoring and DLP modulo composites: factoring reduces probabilistically to DLP in $\mathbb{Z}_N^*$, while DLP modulo N reduces to factoring and DLP modulo prime powers [1].

Second, quantum circuits for Shor-type cryptanalysis are usually problem-specific. Modular exponentiation circuits for factoring require, for example, $2n+2$ qubits and $64n^3 \log_2 n + \mathcal{O}(n^3)$ gates in the Toffoli-based design of Häner et al. [4]. ECDLP circuits instead rely on reversible point addition; Roetteler et al. estimate $9n + 2\lceil \log_2 n \rceil + 10$ qubits and $448n^3 \log_2 n + \mathcal{O}(n^3)$ gates for affine Weierstrass arithmetic [12], with later optimizations in [5].

The present work connects these two lines: the algebraic reductions in Section 4 justify the single Weierstrass arithmetic backend evaluated in Section 6. The resource estimates should be read as leading-order architectural comparisons rather than exact synthesis counts.

3 Preliminaries

We recall the hidden subgroup problem (HSP) and its abelian variant (AHSP), then give a formalization of the *standard abelian HSP (SAHSP)* encodings used in cryptography (factoring, finite-field DLP, ECDLP). Finally, we state the two algorithmic models we work with: *black-box (agnostic)* and *semi-agnostic*.

3.1 HSP and AHSP

Definition 1 (Hidden Subgroup Problem (HSP)). *Let G be a finite group and X a finite set. An oracle $f : G \to X$ hides a subgroup $H \leq G$ if f is constant on left cosets of H and distinct on different cosets:*

$$f(g_1) = f(g_2) \iff g_1 H = g_2 H.$$

The HSP is: given access to oracle f, recover (generators for) H.

3.2 Standard abelian HSP (SAHSP)

Many cryptographic problems solved by Shor's algorithm are instances of AHSP once G and hiding oracle f are specified. We group the canonical encodings below; each fits the HSP definition (domain G is the group in which the hidden subgroup lives, not merely the group on which an external computation is performed).

Integer factorization (order-finding) Fix composite $N \geq 2$ and a base $a \in \mathbb{Z}_N^*$ with (unknown) order $r = \mathrm{ord}_N(a)$. For an exact finite-HSP formulation, assume a promised multiple M of r and define

$$G = \mathbb{Z}_M, \qquad f(x) = a^x \bmod N.$$

Then

$$f(x) = f(y) \iff x \equiv y \pmod{r},$$

so f hides $H = \langle r \rangle \leq \mathbb{Z}_M$. Recovering r yields factors of N by standard post-processing. In an actual Shor circuit one usually samples with a power-of-two modulus rather than with a known multiple of r; continued-fraction post-processing recovers r with high probability. The exact finite-HSP notation is used only to state the hidden subgroup structure cleanly.

Finite-field DLP Let $g \in \mathbb{F}_p^*$ generate a cyclic subgroup of order $r = \mathrm{ord}(g)$, and let $h = g^d$ with unknown d.

$$G = \mathbb{Z}_r \times \mathbb{Z}_r, \qquad f(x, y) = g^x h^y \in \mathbb{F}_p^*.$$

Then f hides the cyclic subgroup $H = \langle (d, -1) \rangle \leq \mathbb{Z}_r \times \mathbb{Z}_r$. Recovering H reveals d (up to r), since $(d, -1)$ generates the kernel of $(x, y) \mapsto g^x h^y$.

Elliptic-curve DLP (ECDLP) Let $E/\mathbb{F}_q$ and $P \in E(\mathbb{F}_q)$ have prime order r, and let $Q = [d]P$. Define

$$G = \mathbb{Z}_r \times \mathbb{Z}_r, \qquad f(x, y) = [x]P + [y]Q \in E(\mathbb{F}_q).$$

Then f hides the cyclic subgroup $H = \langle (d, -1) \rangle$. Recovering H again yields $d \pmod r$.

Definition 2 (SAHSP — Standard Abelian HSP). *The class* SAHSP *consists of AHSP instances arising from the canonical cryptographic encodings above (and close variants with the same hidden subgroup). An algorithm that solves SAHSP solves, in particular, integer factorization, the finite-field DLP, and the elliptic-curve DLP.*

Remark 1 (On domains and finiteness). Shor's period-finding is often implemented with a power-of-two Fourier register whose size is chosen large enough for continued-fraction post-processing. That register need not be a multiple of the hidden period. For the exact finite-HSP formalization used here, we restrict to a finite cyclic $G = \mathbb{Z}_M$ with M a multiple of the order, which preserves the same hidden subgroup. For DLP/ECDLP, restricting to a prime-order subgroup is standard and yields $G = \mathbb{Z}_r \times \mathbb{Z}_r$.

3.3 Algorithmic models: black-box (agnostic) vs. semi-agnostic

Notation for the group domain Given a Weierstrass cubic C/R (smooth or singular), write $C^{\mathrm{ns}}(R)$ for its nonsingular R-points and let $C^{\mathrm{grp}}(R) \subseteq C^{\mathrm{ns}}(R)$ denote the subset on which the chord–tangent addition law is defined and closed. Over fields this is $C^{\mathrm{grp}}(R) = C^{\mathrm{ns}}(R)$ with identity the (smooth) point at infinity. Over rings (e.g., $R = \mathbb{Z}_N$), the declared domain depends on the formulas used. For the split nodal model in the factoring reduction we use a projective parametrization, so the relevant group domain is well defined for all elements of $\mathbb{Z}_N^*$. If an affine formula requiring modular inversion is evaluated and a denominator is a non-unit, a nontrivial $\gcd(\cdot, N)$ is returned as a classical factoring-by-failure event.

We isolate the minimal assumptions our algorithms make about the input structure and oracles.

Definition 3 (Black-box abelian group model (agnostic algorithm)). *An* agnostic *algorithm for AHSP works in the* black-box group model*: it receives (i) uniform encodings of elements of a finite abelian group G, (ii) oracles for equality and group operations (multiplication/inversion or addition/negation), and (iii) oracle access to $f : G \to X$ that hides a subgroup $H \leq G$. The algorithm may not assume any presentation of G beyond these oracles.*

Definition 4 (Semi-agnostic ECDLP solver). *A* semi-agnostic ECDLP solver *is an algorithm that, given an explicit model of a (possibly singular)*

Weierstrass cubic C over a base ring/field R (e.g., $R = \mathbb{F}_q$ or $R = \mathbb{Z}_N$), together with a well-defined group law $\oplus$ on a specified subset $C^{\mathrm{grp}}(R)$, solves:

$$\textit{Given } P, Q \in C^{\mathrm{grp}}(R) \textit{ with } Q = [x]P, \textit{ recover } x \bmod \mathrm{ord}(P).$$

Crucially, the solver:

- *does not require a priori knowledge of $\#C^{\mathrm{grp}}(R)$, its factorization, or a decomposition of $C^{\mathrm{grp}}(R)$;*
- *may operate over rings (e.g., $R = \mathbb{Z}_N$) without knowing the factorization of N;*
- *relies only on the explicit arithmetic of the given model (addition/doubling formulas where defined), i.e., it is* semi-agnostic *to structure beyond the provided law $\oplus$ and its domain of definition.*

Remark 2 (Well-defined instances and domain of definition). All inputs P, Q must lie in the declared group domain $C^{\mathrm{grp}}(R)$ where $\oplus$ is closed and efficiently computable (e.g., the nonsingular locus with the point at infinity as identity for a split nodal cubic). Over rings, this domain may be represented projectively, as in our nodal $\mathbb{Z}_N$ construction. Affine formulas are used only on charts where their inversions are defined; denominator failures are handled outside the oracle by classical gcd computations.

Summary The SAHSP encodings above align each cryptographic task with a bona fide AHSP instance on the right domain group G, and the two models separate the fully black-box setting from our *semi-agnostic* setting, where one uses explicit (possibly singular) Weierstrass models without assuming group-order information.

4 Semi-agnostic model and singular-cubic arithmetic

Throughout, K denotes either a field $\mathbb{F}_q$ of characteristic $p > 3$ or the ring $\mathbb{Z}_N$ with the unit-domain conventions below. Geometric assertions about singularities are stated over fields; over $\mathbb{Z}_N$ the same formulas are interpreted componentwise after CRT or directly as rational/projective formulas on their declared domains.

4.1 Convention (rings)

When working over $K = \mathbb{Z}_N$, we assume $\gcd(N, 6) = 1$; otherwise a nontrivial factor is obtained immediately by $\gcd(N, 2)$ or $\gcd(N, 3)$. Under $\gcd(N, 6) = 1$, both 2 and 3 are units in $\mathbb{Z}_N$, so the formulas below are valid on their stated unit domains.

Over a ring $K = \mathbb{Z}_N$, projective formulas are used for the nodal group map in the formal reduction. Affine formulas may still be used as an implementation interface on charts where the required denominators are units. Whenever such an affine denominator fails to be a unit, we detect and report this by a nontrivial gcd, which can be used to factor N. Thus factoring-by-failure is an explicit outcome of attempting classical modular inversions, not an internal side channel of the quantum oracle.

4.2 The nodal cubic $E: y^2 = x^3 + x^2$ and its group law

Consider the affine plane cubic

$$E: \quad y^2 = x^3 + x^2 = x^2(x+1).$$

Let $F(x,y) = y^2 - x^3 - x^2$. Then the partial derivatives are

$$\frac{\partial F}{\partial x} = -3x^2 - 2x, \qquad \frac{\partial F}{\partial y} = 2y.$$

If K is a field of characteristic different from 2 and 3, the only common zero of $F, \frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}$ is $(x,y) = (0,0)$.

Lemma 1 (Singularity and type). *Over any field K of characteristic $\neq 2, 3$, the point $(0,0)$ is the unique singular point of E. It is an ordinary double point (a split node), with distinct tangent directions $y = \pm x$.*

Proof. From $2y = 0$ we get $y = 0$. The equation $-3x^2 - 2x = 0$ gives $x = 0$ or $x = -2/3$; the latter does not satisfy $x^2(x+1) = 0$ because it gives $4/27 \neq 0$. Thus $(0,0)$ is the only singular point. The quadratic part at the origin is $y^2 - x^2 = (y-x)(y+x)$, which has distinct factors in characteristic different from two.

Write E^{ns} for the nonsingular locus of E. We use the standard rational parametrization of E^{ns} by the parameter $t \in \mathbb{P}^1 \setminus \{\pm 1\}$:

$$P(t) = \big(x(t), y(t)\big) = \big(t^2 - 1\ ,\ t(t^2-1)\big),$$

where $t = \pm 1$ map to the node $(0,0)$, and $t = \infty$ maps to the point at infinity (which we take as the identity).

A parameter that linearizes the group law is

$$\psi : \mathbb{P}^1 \setminus \{\pm 1\} \longrightarrow K^*, \quad \psi(t) = \frac{t-1}{t+1}, \quad \psi(\infty) = 1. \tag{1}$$

Its inverse on $K^* \setminus \{1\}$ is $t = (1+u)/(1-u)$, and we extend it by $u = 1 \mapsto t = \infty$. Substituting yields the birational map

$$\Phi(u) = \Big(\frac{4u}{(1-u)^2}, \frac{4u(1+u)}{(1-u)^3}\Big) \qquad (u \neq 1), \tag{2}$$

and we set $\Phi(1) = \mathcal{O}$, the point at infinity and identity of the chord–tangent group law. For computations over rings it is safer to use the equivalent projective parametrization

$$\widetilde{\Phi}(u) = \big(4u(1-u) : 4u(1+u) : (1-u)^3\big) \in E^{\mathrm{ns}}(K), \tag{3}$$

which agrees with (2) whenever $1-u$ is a unit and sends $u = 1$ to $(0:1:0) = \mathcal{O}$. If $2 \in K^*$ and $u \in K^*$, these homogeneous coordinates generate the unit ideal, so (3) is a well-defined projective point even when the affine denominators in (2) are not units.

Proposition 1 (Group structure on E^{ns}). *Let K be a field of characteristic $\neq 2, 3$. Then $(E^{\mathrm{ns}}(K), \oplus)$ with the chord–tangent law and identity the point at infinity is canonically isomorphic to K^* via Φ:*

$$\Phi(uv) = \Phi(u) \oplus \Phi(v),$$

with $u = 1$ mapping to the identity.

Proof. If a line meets E at three affine points $P(t_1), P(t_2), P(t_3)$ (counting multiplicities), then

$$\psi(t_1)\psi(t_2)\psi(t_3) = 1.$$

This is checked by substituting $x(t), y(t)$ into the line equation $ax + by + c = 0$, yielding a polynomial of degree at most three in t; after including the possible intersection at infinity, Vieta relations imply $\prod(t_i - 1) = \prod(t_i + 1)$. Reflection across the x-axis corresponds to $t \mapsto -t$ and satisfies $\psi(-t) = 1/\psi(t)$. Consequently, the group law defined by

$$P(t_1) \oplus P(t_2) = P(-t_3)$$

is transported by ψ into multiplication:

$$\psi(\text{parameter of } P(t_1) \oplus P(t_2)) = \psi(t_1)\psi(t_2).$$

For a tangent at $P(t)$ intersecting again at t_3, one obtains

$$\psi(-t_3) = \psi(t)^2.$$

Thus the doubling law also matches multiplicative structure.

Using the "three collinear points" identity and the involution $t \mapsto -t$ corresponding to $y \mapsto -y$, one derives $\psi(t_1)\psi(t_2)\psi(t_3) = 1$ whenever $P(t_1), P(t_2), P(t_3)$ are collinear (with multiplicity). Translating via (2) yields multiplicativity.

Remark 3 (Affine chart versus projective group map). When $K = \mathbb{Z}_N$, the affine formula (2) is available only on the chart

$$K^*_{(1)} = \{\, u \in \mathbb{Z}^*_N : \; 1 - u \in \mathbb{Z}^*_N \,\}.$$

This chart is not a subgroup of $\mathbb{Z}^*_N$, so it should not be used as the group domain in the ring reduction. The projective map (3), however, is defined for all $u \in \mathbb{Z}^*_N$ and gives the group-theoretic embedding used below. If an implementation first tries the affine chart and finds that $1 - u$ is a non-unit, then $\gcd(1 - u, N)$ either gives a nontrivial factor of N or shows that $u = 1$ in $\mathbb{Z}_N$.

4.3 Finite-field DLP $\leq$ semi-agnostic ECDLP

Let $K = \mathbb{F}_q$ with $\mathrm{char}(K) > 3$, let $g \in K^*$ have order r, and let $h = g^x$.

Theorem 1 (Finite-field DLP reduces to semi-agnostic ECDLP). *Given* $(g, h = g^x) \in (\mathbb{F}_q^*)^2$*, set* $P = \Phi(g)$ *and* $Q = \Phi(h) \in E^{\mathrm{ns}}(\mathbb{F}_q)$*. Then* $Q = [x]P$ *in* $(E^{\mathrm{ns}}(\mathbb{F}_q), \oplus)$*. Hence any semi-agnostic ECDLP solver (Def. 4) returns* $x \bmod \mathrm{ord}(P)$ *in polynomial time in* $\log q$*.*

Proof. Over $\mathbb{F}_q$ with $\mathrm{char}(\mathbb{F}_q) > 3$, the map

$$\psi(t) = \frac{t-1}{t+1}, \qquad \Phi(u) = \Big(\frac{4u}{(1-u)^2}\,,\ \frac{4u(1+u)}{(1-u)^3}\Big)$$

gives an isomorphism $(\mathbb{F}_q^*, \cdot) \simeq (E^{\mathrm{ns}}(\mathbb{F}_q), \oplus)$, sending $u = 1$ to the identity (the point at infinity). Hence

$$Q = \Phi(h) = \Phi(g^x) = [x]\,\Phi(g) = [x]P.$$

Since Φ is an isomorphism, $\mathrm{ord}(P) = \mathrm{ord}(g) = r$. Any semi-agnostic ECDLP solver that returns $x \bmod \mathrm{ord}(P)$ thus recovers the DLP exponent x modulo r. This reduction runs in time polynomial in $\log q$; the degenerate case $g = 1$ can be handled separately.

4.4 Working over $\mathbb{Z}_N$: domain, failure modes, and factoring

Proposition 2 (Projective multiplicative parametrization over $\mathbb{Z}_N$). *Assume* $\gcd(N, 6) = 1$ *and let*

$$E/\mathbb{Z}_N: \quad Y^2Z = X^3 + X^2Z$$

be the projective closure of $y^2 = x^3 + x^2$*. The map*

$$\widetilde{\Phi} : \mathbb{Z}_N^* \longrightarrow E^{\mathrm{grp}}(\mathbb{Z}_N), \qquad u \longmapsto \big(4u(1-u) : 4u(1+u) : (1-u)^3\big)$$

transports multiplication in $\mathbb{Z}_N^*$ *to the chord–tangent law on the split nodal cubic:*

$$\widetilde{\Phi}(uv) = \widetilde{\Phi}(u) \oplus \widetilde{\Phi}(v), \qquad u, v \in \mathbb{Z}_N^*.$$

On the affine chart $1 - u \in \mathbb{Z}_N^*$*, this map coincides with* (2)*. If the affine chart is attempted and a denominator is not a unit, computing the corresponding gcd with* N *gives the advertised factoring-by-failure interface.*

Proof. The displayed homogeneous coordinates satisfy $Y^2Z = X^3 + X^2Z$ by direct substitution. They are never all zero modulo any maximal ideal: if $u \equiv 1$ in the residue field, then $Y \equiv 8$; otherwise $Z \not\equiv 0$ after reduction. Hence they define a projective point whose fibers avoid the node. The identity for the split nodal group is $\widetilde{\Phi}(1) = (0 : 1 : 0)$.

The parameter is recovered on the image by

$$u = \frac{Y - X}{Y + X},$$

since for the displayed coordinates $Y - X = 8u^2$ and $Y + X = 8u$, with $8u$ a unit. Hence the map is injective on $\mathbb{Z}_N^*$.

The equality $\widetilde{\Phi}(uv) = \widetilde{\Phi}(u) \oplus \widetilde{\Phi}(v)$ is the standard group-scheme isomorphism $\mathbb{G}_m \simeq E^{\mathrm{ns}}$ for the split nodal cubic. Concretely, it is obtained from the projective chord–tangent addition formulas as a polynomial identity over $\mathbb{Z}[1/2][u^{\pm 1}, v^{\pm 1}]$ after clearing projective scaling factors; therefore it remains valid after base change to $\mathbb{Z}_N$. The affine statement follows by dehomogenizing when $1 - u$ is a unit.

Proposition 3 (Factoring reduces to semi-agnostic ECDLP on the nodal cubic over $\mathbb{Z}_N$). *Assume* $\gcd(N, 6) = 1$ *and let*

$$E/\mathbb{Z}_N : \quad Y^2Z = X^3 + X^2Z$$

be equipped with the projective unit-domain group law transported from $\mathbb{Z}_N^*$ *by Proposition 2. Let* $\mathcal{O}_{\mathrm{saECDLP}}$ *be a semi-agnostic ECDLP solver (Def. 4) that, on input* $(E/\mathbb{Z}_N, P, Q)$ *with* $P, Q \in E^{\mathrm{grp}}(\mathbb{Z}_N)$ *and* $Q = [x]P$*, returns* $x \bmod \mathrm{ord}(P)$ *in probabilistic polynomial time. Then integer factorization of* N *reduces in probabilistic polynomial time to* $\mathcal{O}_{\mathrm{saECDLP}}$*.*

Proof (Proof sketch). We use Bach's probabilistic reduction from factoring N to solving DLP instances in $\mathbb{Z}_N^*$ [1]. Given such an instance $(g, h = g^x \bmod N)$, map

$$P = \widetilde{\Phi}(g), \qquad Q = \widetilde{\Phi}(h)$$

by Proposition 2. Then

$$Q = \widetilde{\Phi}(h) = \widetilde{\Phi}(g^x) = [x]\widetilde{\Phi}(g) = [x]P,$$

and the projective parametrization is injective, so $\mathrm{ord}(P) = \mathrm{ord}_N(g)$. The semi-agnostic ECDLP oracle therefore returns the same exponent modulo the same order as the original DLP instance. Composing this DLP solver with Bach's reduction gives a probabilistic polynomial-time reduction from factoring to $\mathcal{O}_{\mathrm{saECDLP}}$.

If the implementation uses the affine form (2) as a first step, a failed inversion of $1-g$ or $1-h$ yields $\gcd(1-g, N)$ or $\gcd(1-h, N)$. Whenever this gcd is neither 1 nor N, a nontrivial factor has already been found; otherwise the projective map above still provides the well-defined oracle instance. Thus factoring-by-failure is a classical reduction-interface feature, not an assumption about the internal behavior of the quantum oracle.

4.5 Transformation to short Weierstrass form

For compatibility with standard cryptographic tools, E can be written in short Weierstrass form by shifting $X = x + \frac{1}{3}$, $Y = y$ (over fields of characteristic $\neq 3$ or rings where 3 is a unit), yielding

$$E_2 : \quad Y^2 = X^3 - \tfrac{1}{3}X + \tfrac{2}{27}.$$

This preserves intersection multiplicities and the chord–tangent law, so the ψ-multiplicativity carries over unchanged.

5 Completeness of the Explicit Weierstrass Oracle

This section proves the equivalence needed by the cryptanalytic engine. The statement is deliberately tied to the explicit oracle domain used in the construction: smooth elliptic curves over finite fields, the split nodal cubic over finite fields, and the same nodal cubic over $\mathbb{Z}_N$ with the projective parametrization of Proposition 2.

5.1 Scope of the explicit oracle

Let $\mathcal{O}_W$ denote the explicit semi-agnostic Weierstrass/ECDLP oracle restricted to the following instance families:

(i) smooth elliptic curves over finite fields $\mathbb{F}_q$ of characteristic greater than three;
(ii) the split nodal cubic $y^2 = x^3 + x^2$ over such finite fields with the ψ/Φ parametrization;
(iii) the same split nodal cubic over $\mathbb{Z}_N$, evaluated through the projective map $\widetilde{\Phi}$ and the declared unit-domain group law.

This is the oracle implemented by the proposed Weierstrass backend. The theorem below should not be read as an unconditional lower bound for every conceivable oracle formulation of ECDLP; it is an equivalence for this explicit SAHSP-oriented domain.

Theorem 2 (Completeness of the explicit Weierstrass oracle). *We have*

$$\text{SAHSP} =_T^P \mathcal{O}_W.$$

Proof. *SAHSP* $\leq_T^P \mathcal{O}_W$. Finite-field DLP reduces to the nodal finite-field family by Theorem 1. Integer factoring reduces, by Bach's probabilistic reduction, to DLP in $\mathbb{Z}_N^*$ and then to the nodal ring family by Proposition 3. Ordinary ECDLP over finite fields is solved directly by the smooth-curve family.

$\mathcal{O}_W \leq_T^P$ *SAHSP.* A smooth finite-field ECDLP instance is itself an SAHSP instance. A nodal finite-field instance is, via Proposition 1, exactly a DLP instance in $\mathbb{F}_q^*$. A nodal $\mathbb{Z}_N$ instance is, via Proposition 2, a DLP instance in $\mathbb{Z}_N^*$; Bach's reductions express such DLP instances using factoring and finite-field DLP modulo the prime-power factors. These are SAHSP instances. Hence $\mathcal{O}_W$ and SAHSP are polynomial-time Turing equivalent.

5.2 Scope limitation

The equivalence above should not be interpreted as saying that ordinary smooth-curve ECDLP is generically reducible to finite-field DLP in a comparably sized multiplicative group. The nodal families realize finite-field DLP and the ring setting used for factoring, whereas the smooth-curve family remains the branch used for ordinary ECDLP instances. Thus the universal backend is a Weierstrass backend, not a finite-field-DLP-only backend.

6 Universal Weierstrass Engine for Hardware-Supported Quantum Cryptanalysis

We now show how Shor's SAHSP algorithm specializes to factoring, finite-field DLP, and elliptic-curve DLP in the semi-agnostic model, and we present a universal reconfigurable circuit that supports all of these instances. The emphasis in this section is cryptanalytic: the same hardware-level arithmetic engine is configured to execute Shor-type attacks against RSA, finite-field DH, and ECC.

6.1 Application of Shor's algorithm

For the hardware-supported cryptanalytic setting considered here, it is sufficient to cover the standard prime-field and integer-modulus targets used by RSA, finite-field DH, and ECC. Hence, our circuit discussion focuses on Factor(N), DLP($\mathbb{F}_p^*$), and ECDLP($\mathbb{F}_p$).

Shor's algorithm, introduced in [13], reduces these problems to order finding in finite abelian groups [10]. In general, the circuit for Shor's algorithm consists of three layers, as illustrated in Figure 1.

The structure of each layer depends on the problem under consideration. The first layer consists of three registers: the upper register reg_U and the lower register reg_L, to which the Hadamard gate is applied, together with an additional register reg_A whose initial state and size depend on the specific problem. The second layer performs the group action by implementing the quantum transformation U_f. Depending on the problem, this transformation is realized either as a sequence of controlled modular multiplications or as controlled additions of points on an elliptic curve, denoted by $\oplus$. The third layer applies the Quantum Fourier Transform (QFT) to the upper and lower registers, followed by measurement. The outcomes of these measurements are then processed classically to recover the final solution.

Here m_A depends on the problem: for factoring it is n qubits, for finite-field DLP it is n qubits, and for ECDLP it is typically $2n$ qubits to store affine coordinates. In the following displays, the second ket is the initialized output/work register on which the reversible group action is accumulated.

Factoring For factorization of an integer N with bitlength n, the algorithm uses a combined upper and lower register of size $2n$, together with an additional register of size n [9]. Transformation U_f is defined as follows:

$$U_f : \ |x\rangle\,|1\rangle \longmapsto |x\rangle\,|a^x \bmod N\rangle\,.$$

Using the binary notation of the value x, we get sequence of controlled modular multiplications, by a^{2^i} [9,2].

Finite-field discrete logarithm For solving discrete logarithms over finite fields, the algorithm uses upper, lower and additional register size of n bits. For

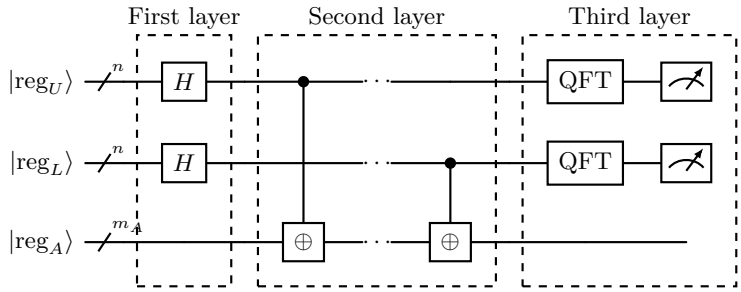

Fig. 1. Schematic Shor layout with an abstract arithmetic block $\oplus$.

$\mathbb{F}_p^* = \langle g \rangle$ and $h = g^d$, transformation U_f is defined as follows:

$$U_f : \ |x, y\rangle |1\rangle \longmapsto |x, y\rangle |g^x h^y\rangle .$$

As with factorization, the transformation can be decomposed as a sequence of controlled modular multiplications, by g^{2^i} and h^{2^j}.

Elliptic-curve discrete logarithm The algorithm for solving discrete logarithms on elliptic curves uses upper and lower n bit-registers and $2n$-bit additional register which is supposed to hold results of point addition [5]. For $Q = [d]P$, transformation U_f is defined as follows:

$$U_f : \ |x, y\rangle |\mathcal{O}\rangle \longmapsto |x, y\rangle |[x]P + [y]Q\rangle .$$

As for the two previous problems, this transformation can be decomposed as a sequence of controlled additions of precomputed doubles $[2^i]P$ and $[2^j]Q$ [10].

6.2 Universal circuits for group operations

In this section we examine the construction of a universal Shor circuit capable of solving Factor(N), DLP($\mathbb{F}_p^*$), and ECDLP($\mathbb{F}_p$). As described in the previous subsection, the circuits for these problems differ in the second layer, since each requires a distinct group operation.

To build a single circuit that solves all three problems without relying on three separate implementations, we need a universal U_f transformation. We compare our proposal for such a transformation with two baseline approaches that do not introduce unified arithmetic: static allocation of resources and dynamic switching between two arithmetic units. Our design, which we call the Universal Weierstrass Circuit, enables solving all three problems within a single framework based on Weierstrass cubic equation.

Approach 1: Static allocation with recompilation In this approach, the number of qubits and gates is allocated according to the requirements of the

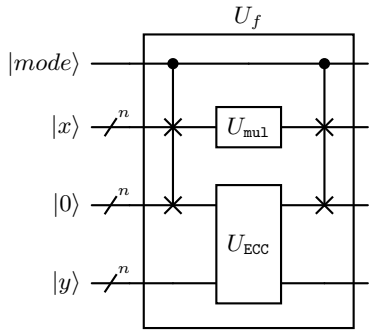

Fig. 2. Universal circuit for the hidden subgroup problem implementing a group operation.

most costly circuit. The resource demands are dominated by elliptic curve point addition. If the problem to be solved changes, the entire circuit must be recompiled using the preallocated resources.

Approach 2: Dynamic switching with controlled swaps This approach uses controlled swap gates, known as Fredkin gates. Figure 2 presents a schematic diagram of the proposed universal arithmetic circuit. The U_f transformation consists of two preloaded arithmetic blocks: modular multiplication $U_{\texttt{mul}}$ [2,4] and point addition on an elliptic curve $U_{\texttt{ECC}}$ [12,5]. The block selection is done by an additional qubit and a Fredkin gate. This approach requires $n + 1 + |q_{\texttt{mul}}| + |q_{\texttt{ECC}}|$ qubits, $\texttt{max}\{D_{\texttt{mul}}, D_{\texttt{ECC}}\}$ depth and $|G_{\texttt{mul}}| + |G_{\texttt{ECC}}| + 2n$ gates.

Our proposal. Universal Weierstrass engine This approach allows each SAHSP problem to be attacked using the same Weierstrass arithmetic. As described in Section 4, the multiplicative group actions for factoring and finite-field DLP are represented as point additions on the split nodal cubic after the parametrization is applied. Thus the engine does not implement a separate modular-exponentiation backend; it implements the corresponding group action through the Weierstrass addition layer and supports ring arithmetic over $\mathbb{Z}_N$. Its hardware-supported cryptanalytic advantages are:

- the same leading estimate as a static allocation provisioned for the specialized elliptic-curve circuit;
- one arithmetic backend for RSA, finite-field DH, and ECC cryptanalysis;
- practical factoring-by-failure detection over $\mathbb{Z}_N$ at the classical reduction interface.

In the quantum circuit layer, U_f is implemented as a fully reversible/unitary arithmetic map; in the ring case this means using the projective group-law representation rather than affine inversions inside the quantum oracle. "Factoring-by-failure" pertains only to the classical reduction interface (explicit modular inversion attempts), not to any internal side channel of the quantum oracle.

Table 1. Resource estimates for universal Shor circuits with parameter size n (bitlength of modulus or field size). Estimates from [4,12].

Circuit design	Qubits	Gates	Depth
Static allocation with recompilation	$9n + 2\lceil \log n \rceil + 10$	$448n^3 \log n + \mathcal{O}(n^3)$	$\mathcal{O}(n^3)$
Dynamic switching with controlled swaps	$12n + 2\lceil \log n \rceil + 13$	$512n^3 \log n + \mathcal{O}(n^3)$	$\mathcal{O}(n^3)$
Universal Weierstrass (Our proposal)	$9n + 2\lceil \log n \rceil + 10$	$448n^3 \log n + \mathcal{O}(n^3)$	$\mathcal{O}(n^3)$

This circuit avoids a complete rearrangement of the arithmetic logic when changing the cryptanalytic target; only the mode, public parameters, and precomputed constants need to be updated. Moreover, our design does not require implementing two separate arithmetic circuits at the same time.

6.3 Complexity discussion

We now estimate the resources required for each of the approaches discussed. As noted in [12], providing precise estimates is difficult, and the numbers below should be read as leading-order architecture estimates rather than exact synthesis counts. The estimates for modular multiplication are taken from [4], while those for elliptic curve point addition are taken from [12]. Specialized RSA/DH modular-exponentiation circuits have smaller constants than an elliptic-curve point-addition backend; the universal engine trades those constants for a single reconfigurable Weierstrass arithmetic layer.

Table 1 compares the resource requirements of the different approaches. Our Universal Weierstrass Circuit is provisioned at the same leading estimate as static allocation to the elliptic-curve backend, but it eliminates the need for full arithmetic-pipeline recompilation when switching between different SAHSP instances. In contrast, dynamic switching between arithmetic circuits with Fredkin gates consumes significantly more resources: the Fredkin-switching circuit requires more qubits due to instantiating both arithmetic units, while the leading-order gate and depth costs remain dominated by the selected arithmetic block (the routing overhead is lower order).

Summarizing, a single Universal Weierstrass Circuit suffices for factoring, finite-field DLP, and ECDLP in the explicit semi-agnostic model. This approach matches the worst-case elliptic-curve allocation used by the recompilation baseline while avoiding the need to recompile the arithmetic pipeline. In addition, it requires significantly fewer resources than Fredkin switching with two preloaded arithmetic units.

7 Conclusions

We presented a hardware-supported quantum-cryptanalytic framework in which RSA, finite-field DH, and ECC are attacked through a single explicit semi-agnostic Weierstrass oracle. The technical mechanism is an explicit Weierstrass representation: smooth curves handle ordinary ECDLP, the nodal cubic $y^2 =$

$x^3 + x^2$ realizes finite-field DLP, and the corresponding ring setting over $\mathbb{Z}_N$ supports factoring-by-failure at the classical reduction interface while the quantum oracle remains projective and reversible.

The resulting Universal Weierstrass engine is a single programmable arithmetic backend for Shor-type cryptanalysis of RSA, finite-field DH, and ECDH. It matches the leading resource estimate of a worst-case elliptic-curve allocation while avoiding both full arithmetic-pipeline recompilation and dynamic switching between separate modular-multiplication and elliptic-curve units. This makes the proposed architecture a compact target for hardware-supported quantum cryptanalytic benchmarking and assessment of software-deployed public-key cryptography.

References

1. Bach, E.: Discrete logarithms and factoring. Tech. rep., University of California, Berkeley (1984)
2. Beauregard, S.: Circuit for Shor's algorithm using 2n+3 qubits (2003)
3. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory **22**(6), 644–654 (1976)
4. Haner, T., Roetteler, M., Svore, K.M.: Factoring using 2n+2 qubits with Toffoli based modular multiplication (2017)
5. Häner, T., Roetteler, M., Svore, K.M.: Improved quantum circuits for elliptic curve discrete logarithms. In: Post-Quantum Cryptography (PQCrypto 2020). LNCS, vol. 12100, pp. 425–444. Springer (2020)
6. Koblitz, N.: Elliptic curve cryptosystems. Mathematics of Computation **48**(177), 203–209 (1987)
7. Menezes, A., Okamoto, T., Vanstone, S.: Reducing elliptic curve discrete logarithm to the discrete logarithm in a finite field. IEEE Transactions on Information Theory **39**(5), 1639–1646 (1993)
8. Miller, V.S.: Use of elliptic curves in cryptography. In: Advances in Cryptology — CRYPTO '85. LNCS, vol. 218, pp. 417–426. Springer (1986)
9. Pavlidis, A., Gizopoulos, D.: Fast quantum modular exponentiation architecture. In: Proceedings of the 2013 Design, Automation & Test in Europe Conference & Exhibition (DATE). pp. 1–6. IEEE (2013)
10. Proos, J., Zalka, C.: Shor's discrete logarithm quantum algorithm for elliptic curves. Quantum Information & Computation **3**(4), 317–344 (2003)
11. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM **21**(2), 120–126 (1978)
12. Roetteler, M., Naehrig, M., Svore, K.M., Lauter, K.: Quantum resource estimates for computing elliptic curve discrete logarithms. IACR Cryptology ePrint Archive **2017**, 598 (2017)
13. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS). pp. 124–134. IEEE (1994)