

Comparison of Cryptographic Hardware Implementations for RSA, DH and ECDH Using a Semi-Agnostic Approach in FPGA Architectures*

Wiktoria CIEPLA

Military University of Technology, Warsaw, Poland

Correspondence should be addressed to: Wiktoria CIEPLA, wikiciepla@gmail.com

*Presented at the 47th IBIMA International Conference, 29-30 June 2026, Madrid, Spain

Abstract

This work investigates whether the computational models used to implement the ECDH protocol can also serve as a common computational framework for the classical Diffie-Hellman protocol and RSA. The starting point was the assumption that, despite the algebraic differences between these protocols, their hardware implementations rely on the same fundamental operations: modular addition, subtraction, multiplication, and reduction.

As part of the study, three variants of the algebraic-curve-based component were designed: one based on singular curves, one based on twisted Edwards curves, and one based on the RCB model. The obtained synthesis results and comparative analysis demonstrated that the semi-agnostic approach is practically justified. In particular, the variants based on twisted Edwards curves and the RCB model proved especially promising, as in these cases the computational machinery typically associated with ECC can also be used to perform operations classically realized through modular exponentiation in RSA and Diffie-Hellman.

Introduction

RSA, classical Diffie-Hellman (DH), and ECDH are among the fundamental mechanisms of asymmetric cryptography; however, in the classical view they are usually treated as schemes based on different algebraic structures. RSA operates in the ring of residue classes modulo N , classical DH in the multiplicative group of a finite field, whereas ECDH operates in the group of points of an elliptic curve defined over a finite field.

From the perspective of hardware implementation, however, these differences do not imply the need to construct completely independent computational paths. At the hardware level, all of the considered protocols rely on the same modular operations, which makes it possible to design resource-sharing architectures in which the main differences lie in the control logic and the representation of input data.

The aim of this work was therefore to investigate whether computational models usually associated with ECDH can be extended so as to also cover operations characteristic of DH and RSA. To this end, a semi-agnostic approach was adopted, and three variants of algebraic representations were analyzed, assessed not only from the mathematical point of view, but also with regard to their implementation suitability in FPGA structures.

Theoretical Background

The core idea of the semi-agnostic approach is that RSA and classical DH are viewed through the lens of

modular exponentiation, whereas ECDH is viewed through the lens of scalar multiplication of a point on a curve. Despite this difference, both types of tasks can be decomposed into the same elementary operations, which opens the way to the construction of a shared arithmetic core.

A DH-RSA module for modular exponentiation was developed, together with three different ECDH implementation variants corresponding to the three algebraic models analyzed in this work.

The first variant is based on singular curves, in particular on the singular curve given by the equation $E: y^2 = x^3 + x^2$ as described by Wroński et al. (2025a). This curve has a singularity at the point $(0, 0)$, which is a node. This curve is an important intermediate step, since it makes it possible to relate the point-addition formalism to the operation of the multiplicative group.

An appropriate parametrization makes it possible to describe the affine points of the curve in terms of a single parameter:

$$P(t) = (t^2 - 1, t(t^2 - 1)).$$

The key property is that point addition on such a curve corresponds to multiplication in the multiplicative group, which provides the mathematical foundation for transitioning from the classical discrete logarithm problem to a model based on an algebraic curve. To this end, one uses the function $\psi(t)$ defined by

$$\psi(t) = \frac{t-1}{t+1}.$$

This transition is of considerable importance, as it shows that the computational machinery known from elliptic-curve cryptography does not have to be restricted solely to classical smooth elliptic curves. As a result, operations characteristic of classical DH, and more generally also RSA, can be interpreted within a broader algebraic setting without the need to completely redesign the hardware architecture.

The next step is a transformation to the short Weierstrass form by the substitution $X = x + 1/3$, $Y = y$, which leads to the equation $E_2: Y^2 = X^3 - 1/3 X - 2/27$. This transformation eliminates the quadratic term and formally brings the model closer to the short Weierstrass form, which is widely used in cryptography, thereby making the transition between the classical ECC formalism and semi-agnostic constructions more natural.

In the actual implementation, the Weierstrass-based variant follows the classical ECDH approach, employing separate point addition and point doubling modules, together with a top-level module that performs iterative scalar multiplication using the double-and-add method. This is the most reference-like variant: it clearly demonstrates how modular arithmetic can be shared, but it does not yet integrate the RSA/DH computational path with the point-addition machinery as directly as the remaining two variants.

The second variant is based on twisted Edwards curves described by the equation $E_{TEd}: ax^2 + y^2 = 1 + dx^2y^2$, where the parameters a and d are chosen so that the curve is nonsingular as described by Wroński et al. (2025b). This model is highly attractive from the implementation point of view, since it has a simple neutral element $(0, 1)$, a straightforward point negation operation, and convenient point-addition formulas. For points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, their sum $P_1 + P_2 = (x_3, y_3)$ is given by

$$x_3 = \frac{x_1 y_2 + y_1 x_2}{1 + dx_1 x_2 y_1 y_2}, \quad y_3 = \frac{y_1 y_2 - ax_1 x_2}{1 - dx_1 x_2 y_1 y_2}$$

With an appropriate choice of parameters, these addition formulas are complete, that is, they do not require extensive handling of exceptional cases. For hardware implementation, this translates into greater computational regularity, simpler control logic, and a more uniform sequential architecture.

In this variant, the slice mode is of particular importance. For a special representation of the input data, part of the TED computational machinery can execute a simplified path corresponding to modular multiplication, according to the relation

$$(0, y_1) \oplus_{\text{slice}} (0, y_2) = (0, y_1 y_2)$$

This means that a module originally designed for operations on points of a twisted Edwards curve can also be used to perform operations characteristic of RSA and DH. In practice, this does not mean that pairs $(0, y)$ are treated as ordinary curve points; rather, they are used as a special register-level encoding that makes it possible to reuse the existing computational path for modular multiplication.

From the implementation perspective, the addpointTED module therefore operates in two roles: in the classical mode, it performs point addition, whereas in the slice mode it executes a shortened sequence corresponding to the multiplicative path. This means that the same computational machinery can support both ECC tasks and the classical RSA/DH computations. The third variant is based on the RCB model, i.e., the complete Renes-Costello-Batina formulas for curves in short Weierstrass form as described by Wroński et al. (2025c). Their main advantage is that they enable a uniform implementation of point addition without splitting the computation into numerous special cases, which is highly beneficial from the perspective of sequential architecture and resource sharing.

In the RCB model, point addition is realized as a fixed sequence of operations on intermediate registers, regardless of whether the input consists of two distinct points or a special case. Such regularity simplifies the control logic and makes this representation particularly well suited for FPGA implementations.

Here as well, the slice mode plays a key role. In the special mode, for inputs encoded as $(x_1, 0, 1)$ and $(x_2, 0, 1)$, a selected fragment of the existing computation sequence can correspond to modular multiplication, according to the relation

$$(x_1, 0, 1) \oplus_{\text{slice}} (x_2, 0, 1) = (x_1 x_2, 0, 1).$$

As a result, the RCB pipeline can operate both as a mechanism for full point addition on the curve and as a simplified computational path for multiplicative problems. Consequently, similarly to the TED-based variant, the RCB variant provides strong support for the construction of a common architecture for ECDH, DH, and RSA while preserving a high degree of computational regularity.

Developed Architecture

Based on the shared modular arithmetic framework, a set of modules implementing RSA, DH, and ECDH operations was designed. The core component was the fieldunit module, which integrates modular operations and shares resources among higher-level submodules.

For classical RSA and DH, a DHRSA module based on modular exponentiation was developed. For the elliptic-curve part, three ECDH module variants were designed: a variant based on singular curves transformed into short Weierstrass form, with separate point addition and point doubling blocks - addpoint and dblpoint; a TED variant with the addpointTED module; and an RCB variant with the addpointRCB module.

The work also considers multi-mode architectures in which the RSA/DH mode and the ECDH mode coexist within a single design as separate computational paths using the same arithmetic resources. This organization illustrates two possible integration approaches: either by directly reusing the ECC computational machinery for the multiplicative path, or by allowing two operating modes to coexist within one shared structure.

Results

Table 1 presents a summary of the synthesis results for the baseline module implementing RSA/DH operations and for the three analyzed variants of the elliptic-curve architecture: the variant based on the short Weierstrass form, the variant based on twisted Edwards curves (TED) and the variant using the RCB formulas. For each analyzed variant, the table reports the ALM logic resource utilization, the maximum operating frequency, and the number of clock cycles required to complete the computation.

The rows of the table are organized in pairs corresponding to the successive architectural variants. In each case, the first row describes a module in which the computational machinery derived from ECC can also be used to perform operations characteristic of RSA and DH through the use of the slice mode; this means that the same computational path carries out both ECDH operations and operations classically associated with modular exponentiation.

Table 1: Performance of System Components

Variant	Module	ALM	Fmax	Latency [cycles]
Baseline RSA/DH Module				
	DH_RSA	2 412 (13%)	51,2 MHz	540 403
Comparison with ECDH Modules				
Weierstrass	ECDH	12 633 (68%)	42,58 MHz	7 185 780
	ECDH_RSA_DH	13 842 (75%)	37,29 MHz	RSA/DH: 540 403; ECDH: 7 185 780
TED	ECDH_TED	9 308 (50%)	35,26 MHz	RSA/DH: 541 315; ECDH: 7 862 356
	ECDH_RSA_DH_TED	9 512 (51%)	38,55 MHz	RSA/DH: 539 693; ECDH: 7 862 363
RCB	ECDH_RCB	8 681 (47%)	42,57 MHz	RSA/DH: 1 061 907; ECDH: 9 183 510
	ECDH_RSA_DH_RCB	8 962 (48%)	38,32 MHz	RSA/DH: 540 403; ECDH: 9 183 504

The second row in each pair, in turn, represents a multi-mode architecture in which RSA/DH and ECDH operations are handled as separate computational paths sharing a common modular arithmetic framework, but without using the slice mechanism to directly transfer classical computations to the ECC computational machinery. Such a comparison makes it possible to evaluate two different approaches to integrating the RSA, DH, and ECDH protocols within a single hardware structure.

The baseline RSA/DH module, for a 256-bit value of p , uses 2412 ALMs, operates at a frequency of 51.2 MHz, and completes the operation in an average of 540403 clock cycles.

The variant based on singular curves transformed into short Weierstrass form proved to be the most resource-demanding in the ECDH part. The ECDH module occupies 12633 ALMs, operates at 42.58 MHz, and requires an average of 7185780 clock cycles, whereas the ECDH_RSA_DH architecture occupies 13842 ALMs and reaches 37.29 MHz, performing RSA/DH in 540403 cycles and ECDH in 7185780 cycles, respectively.

The TED variant offers a more favorable trade-off between resource utilization and architectural flexibility. The ECDH TED module uses 9308 ALMs, reaches 35.26 MHz, and performs ECDH in 7862356 clock cycles, while in the slice mode RSA/DH operations can be executed in an average of 541315 clock cycles. The extended ECDH_RSA_DH TED architecture occupies 9512 ALMs, operates at 38.55 MHz, and maintains very similar latencies for both modes.

Equally interesting results were obtained for the RCB variant. The ECDH RCB module uses 8681 ALMs, reaches 42.57 MHz, and performs the ECDH operation in 9183510 clock cycles, whereas the RSA/DH path implemented via the slice mode requires an average of 1061907 clock cycles. In turn, the ECDH_RSA_DH RCB architecture

occupies 8962 ALMs, operates at 38.32 MHz, and performs RSA/DH in 540403 clock cycles and ECDH in 9183504 clock cycles.

Discussion of Results

The obtained results show that extending the ECDH architecture to support RSA and DH is less costly than building entirely independent computational blocks. Resource sharing makes it possible to reduce hardware redundancy and simplify the organization of the design, which is particularly important in FPGA-based implementations.

The variant based on singular curves serves in this work as the main reference point. It clearly illustrates the classical way of constructing an ECDH module and the possibility of sharing modular arithmetic, but its more complex structure and distinctly higher resource utilization make it less attractive as the basis for a more universal architecture.

The TED and RCB variants perform significantly better in this respect. In both cases, it was demonstrated that the computational machinery typical of ECC can also be used for tasks classically realized through modular exponentiation, and thus for the implementation of RSA and DH.

The TED variant stands out due to its very favorable balance between resource utilization and functionality. Thanks to the slice mode, it was possible to naturally connect the elliptic-curve path with the multiplicative path, while at the same time preserving the simplicity and regularity of the implementation resulting from the properties of twisted Edwards curves.

The RCB variant also confirmed the usefulness of the semi-agnostic approach. The RCB model provides a highly regular computational structure, complete formulas, and favorable resource utilization, which makes it a strong candidate for the construction of shared cryptographic accelerators.

Conclusions

The conducted study confirmed that the semi-agnostic approach is justified both mathematically and from the hardware perspective. The algebraic representation has a direct impact on the regularity of computations, the complexity of the control logic, and the efficiency of FPGA implementation; therefore, the choice of the curve model is not merely a formal decision, but also an architectural one.

The main conclusion of this work is that computational models derived from algebraic-curve cryptography can provide a common framework not only for ECDH, but also for DH and RSA. This was particularly well demonstrated by the TED and RCB variants, which, thanks to the slice mode, showed that the ECC computational machinery can practically take over tasks classically performed through modular exponentiation.

The proposed architecture may therefore be regarded as an intermediate step toward more universal, reconfigurable, and resource-efficient cryptographic accelerators.

References

- **E. Burek, M. Leśniak, and M. Wroński,** (2025c), *A Unified Renes-Costello-Batina Weierstrass Oracle for Shor-Style Cryptanalysis*, manuscript, NASK National Research Institute, Warsaw, Poland.
- **M. Wroński, Ł. Dzierzkowski, M. Leśniak and E. Syta,** (2025a), *One for All, All for One: Universal Semi-Agnostic Quantum Circuit for Solving Standard Abelian Hidden Subgroup Problems*, preprint/manuscript.
- **M. Wroński, et al.** (2025b), submitted, *A Uniform Twisted-Edwards Group Oracle over Finite Field Extensions of Degree at Most 6 for Standard Abelian Hidden Subgroup Problems*, manuscript submitted to *Quantum Information & Computation*.