Blind signatures from Zero knowledge in the Kummer variety*

Paulo L. Barreto[1], **Devin D. Reich**[1], **Marcos A. Simplicio Jr.**[2] **Gustavo H. M. Zanon**[2]

[1] School of Engineering and Technology – University of Washington | Tacoma

[2]Escola Politécnica – Universidade de São Paulo

pbarreto@uw.edu,devin.d.reich@gmail.com,msimplicio@larc.usp.br,gustavo.zanon@alumni.usp.br

Abstract. *We show how to apply the BZ methodology (Blind signatures from Zero knowledge) to obtain blind signatures in the Kummer varieties defined by Montgomery curves. We also describe specially-tailored arithmetic algorithms to facilitate their efficient implementation. The result can be proved secure under appropriate assumptions, appears to resist even the ROS attack (to which most elliptic-curve blind signature schemes succumb), and is arguably one of the most efficient among those proposals that offer similar security guarantees.*
Keywords: *Blind signatures, Zero-knowledge arguments, Kummer variety.*

1. Introduction

The cryptographic importance of Montgomery curves [Montgomery 1987] resides in their ability to sustain arithmetic operations involving only the x-coordinates of points: since the y-coordinates are not required, they can be omitted altogether, leading to highly efficient computations. This comes at the cost of no longer distinguishing between a point P and its opposite $-P$. That is, operations are not really carried out in the group of points (of a certain large prime order) on the curve, but rather in the so-called *Kummer variety* of the curve. This drawback is inconsequential for simple key agreement protocols, which rely only on multiplications by scalars. However, it could be a hindrance for digital signatures and other more elaborate protocols relying on point addition as well, where the ambiguity between points and their opposites can lead to adverse effects.

The qDSA signature [Renes and Smith 2017] addresses such a drawback, verifying signatures entirely under the Kummer variety. The scheme is the Kummer analog of a Schnorr signature [Schnorr 1990], with the verification constraint relaxed to allow for two verifying conditions instead of one. The existence of a plain signature scheme in the Kummer variety poses the interesting question of what other types of signatures are possible in the same scenario, particularly *blind signatures*, which are commonly used in privacy-related protocols (e.g., e-cash or e-voting) [Tessaro and Zhu 2022] and are known to be hard to obtain securely even in a more conventional setting. This is all the more important when one considers that most (though not all) blind signature schemes built on elliptic curves are susceptible to a recent, powerful attack against the principle underlying those constructions, the so-called ROS (Random inhomogeneities in an Overdetermined Solvable system of linear equations) problem [Benhamouda et al. 2021]. One approach that appears to resist the ROS attack is the $BZ[\mathsf{DL}]$ construction [Barreto and Zanon 2023],

*A preliminary version of this paper was presented and received the Best Paper Award at the XXIII Brazilian Symposium on Information and Computational Systems Security – SBSeg 2023, Juiz de Fora (MG), Brazil.

which stems from zero-knowledge arguments for the possession of a signature. That construction, however, was so far only known to apply to an underlying full group structure, not for a Kummer variety.

Contribution: Our contribution in this paper is a blind signature scheme built with the *BZ* strategy adapted to the Kummer variety setting. This requires a new, biquadratic verification method which, although more complex than the Renes-Smith algorithm, is nevertheless very efficient. Namely, at the 2^κ security level, it only imposes a computational overhead of a fraction $1/(2\kappa)$ of the total processing time. The resulting scheme admits a natural constant-time implementation and turns out to be more efficient than any blind signature scheme based on elliptic curves we are aware of.

Organization: The remainder of this research document is organized as follows. Section 2 introduces basic concepts related to our proposal and the pertaining notation. Section 3 recapitulates the conventional discrete-logarithm $BZ[\mathsf{DL}]$ and $IZ[\mathsf{DL}]$ schemes. Our proposed Kummer-based protocols $BZ[\mathsf{qDL}]$ and $IZ[\mathsf{qDL}]$ are introduced and discussed in Section 4. We describe the novel biquadratic test required by these proposed protocols in Section 5. Section 6 presents a test-of-concept experimental evaluation of our proposals. We conclude in Section 7.

2. Preliminaries

We hereby follow the notation conventions of [Barreto and Zanon 2023]. This includes the color highlights to allow for more easily following the roles of the quantities involved in the protocol across different algorithms and steps.

The set of all integers modulo n (with representatives in range $[0 \ldots n-1]$) is written $\mathbb{Z}_n$. The set of all binary strings is denoted as $\{0,1\}^*$, and given any other set $\mathcal{U}$ containing an additively neutral element ϵ (e.g., $\epsilon = 0$ for the integers), $\mathcal{U}^*$ denotes the set $\mathcal{U} \setminus \{\epsilon\}$. The shorthand $(\langle \textit{condition} \rangle)$ **?** $\langle \textit{value-if-true} \rangle$ **:** $\langle \textit{value-if-false} \rangle$ denotes the choice of one out of the two specified possible values depending on the Boolean *condition*. The shorthand $x \xleftarrow{\$} \mathcal{U}$ denotes the uniformly random sampling of a value x from a set $\mathcal{U}$. When a function returns a tuple and one or more components of that tuple are to be ignored, they are indicated by an empty underscore: e.g., if $f(x)$ returns a pair (y,z), but only y will be used, the function call is written $(y,\_) \leftarrow f(x)$. Let n be a (large) prime number, and let G be a generator of the n-torsion of an elliptic curve. The *Kummer variety* $\mathfrak{G} := \langle G \rangle / \pm$ identifies every pair of opposite points $\{P, -P\} \subset \langle G \rangle$ with a single element $\pm P \in \mathfrak{G}$. It supports the notion of pseudo-multiplication by scalar, inherited in a natural fashion from the group $\langle G \rangle$: for any $k \in \mathbb{Z}_n$, $[n](\pm P) := \pm([k]P)$. Given $\pm P, \pm Q, \pm R \in \mathfrak{G}$, the *Renes-Smith test* [Renes and Smith 2017, Alg. 2] efficiently determines whether $\pm R \in \{\pm(P+Q), \pm(P-Q)\}$ (see Algorithm 30). For the Kummer variety of a Montgomery curve [Montgomery 1987], all of these operations can use only the x-coordinate of the curve points (see Appendix A). The discrete logarithm computation problem (DL) translates to a Kummer variety version: given an element $\pm P \in \langle G \rangle / \pm$, find $m \in \mathbb{Z}_n$ such that $\pm P = [m](\pm G)$. Clearly the Kummer variant is as hard as the conventional DL problem: if it were any easier, an instance $P \in \langle G \rangle$ of DL could be mapped to its Kummer version, solved for m so that $P \in \{[m]G, [-m]G\}$, and the correct value from $\pm m$ could be determined with a multiplication by scalar; the same is certainly true the other way around.

Figure 1. The canonical *BZ* scheme

Signer $BZ.S(sk)$		User $BZ.U(pk, m)$		Verifier $BZ.Ver(pk, m)$
$(\hat{u}, \hat{v}, \mathsf{st}_S) \xleftarrow{\$} BZ.S_1(sk)$	$\xrightarrow{\hat{u},\hat{v}}$			
	$\xleftarrow{\hat{c},\hat{d}}$	$(\hat{c}, \hat{d}, \mathsf{st}_U) \xleftarrow{\$} BZ.U_1(pk, \hat{u}, \hat{v}, m)$		
$\hat{w} \leftarrow BZ.S_2(\mathsf{st}_S, \hat{c}, \hat{d})$	$\xrightarrow{\hat{w}}$	$\sigma \leftarrow BZ.U_2(\mathsf{st}_U, \hat{w})$	$\xrightarrow{\sigma}$	$b \leftarrow BZ.Ver(pk, \sigma, m)$

2.1. The *BZ* blind signature methodology

Essentially, a blind signature scheme is an interactive protocol between a *signer* and a *user*: the *signer* issues a verifiable signature σ on a message chosen by the *user*, and later cannot link σ to the run of the protocol in which it was created. The following formal definitions from [Barreto and Zanon 2023] are relevant to the protocols hereby proposed.

Definition 1 (Canonical Three-move Blind-signature-from-Zero-knowledge Scheme) *A* canonical three-move Blind-signature-from-Zero-knowledge scheme *(see Figure 1) is a 5-tuple of algorithms* $BZ = (BZ.PG, BZ.KG, BZ.S = (BZ.S_1, BZ.S_2), BZ.U = (BZ.U_1, BZ.U_2), BZ.Ver)$ *where:*

- *The randomized parameter generation algorithm* $BZ.PG$ *takes a security parameter* 1^κ *as input and returns system parameters,* param.
- *The randomized key generation algorithm* $BZ.KG$ *takes* param *as input and creates a key pair* (pk, sk). *In turn, the public key defines challenge spaces* $(\mathcal{C}, \mathcal{D}) := \mathcal{CD}(pk)$, *and is known to all parties.*
- *The first, randomized signer algorithm* $BZ.S_1$ *takes as input the secret key* sk *and returns commitments* $(\hat{u}, \hat{v})$ *and the signer's state* st_S.
- *The first, randomized user algorithm* $BZ.U_1$ *takes as input the public key* pk, *commitments* $(\hat{u}, \hat{v})$, *and a message* m, *and returns challenges* $(\hat{c} \in \mathcal{C}, \hat{d} \in \mathcal{D})$ *and the user's state* st_U.
- *The second, deterministic signer algorithm* $BZ.S_2$ *takes as input the signer's state* st_S, *which includes the secret key* sk *and the commitments* $(\hat{u}, \hat{v})$, *together with challenges* $(\hat{c} \in \mathcal{C}, \hat{d} \in \mathcal{D})$, *and returns the response* $\hat{w}$.
- *The second, deterministic user algorithm* $BZ.U_2$ *takes as input the user's state* st_U, *which includes the public key* pk, *the commitments* $(\hat{u}, \hat{v})$, *the message* m, *and the challenges* $(\hat{c} \in \mathcal{C}, \hat{d} \in \mathcal{D})$, *together with response* $\hat{w}$, *and returns a signature* σ, *or* $\perp$ *in case of failure.*
- *The deterministic verification algorithm* $BZ.Ver$ *takes as input the public key* pk, *a signature* σ, *and a message* m, *and returns* $b = 1$ *to indicate acceptance or* $b = 0$ *to indicate rejection (if* $\sigma = \perp$ *the output is always* 0*).*

A secure instantiation of the BZ scheme must satisfy the following properties:

- *Perfect correctness:* for any signature σ that genuinely results from $BZ.U_2$ after a protocol session with a key pair (pk, sk) and a message m, it must hold that $BZ.Ver(pk, \sigma, m) = 1$;
- *Perfect blindness [Hauck et al. 2019, Def. 5.3]:* if $BZ.S$ chooses two messages (m_0, m_1), establishes two sessions with $BZ.U$ with transcripts (T_0, T_1), and ends up observing the resulting two signatures (σ_0, σ_1), then $BZ.S$ cannot tell whether T_0 corresponds to m_0 (and T_1 corresponds to m_1), or T_0 corresponds to m_1 (and T_1 corresponds to m_0) any better than by random guessing;

Figure 2. The canonical IZ scheme

Prover $IZ.P(sk)$		Verifier $IZ.U(pk, m)$
$(\hat{u}, \hat{v}, \mathsf{st}_P) \xleftarrow{\$} IZ.P_1(sk)$	$\xrightarrow{\hat{u},\hat{v}}$	
	$\xleftarrow{\hat{c},\hat{d}}$	$\hat{c} \xleftarrow{\$} \mathcal{C}, \hat{d} \xleftarrow{\$} \mathcal{D}$
$\hat{w} \leftarrow IZ.P_2(\mathsf{st}_P, \hat{c}, \hat{d})$	$\xrightarrow{\hat{w}}$	$b \leftarrow IZ.Ver(pk, \hat{u}, \hat{v}, \hat{c}, \hat{d}, \hat{w})$

- *One-more unforgeability (OMUF) [Hauck et al. 2019, Def. 5.4]:* if $BZ.U$ interacts with $BZ.S$ in ℓ protocol sessions, then $BZ.U$ obtains no more than ℓ blind signatures (namely, only from those sessions that reach completion).

Unforgeability and correctness of BZ instantiations build upon an underlying identification scheme IZ, in which a *prover* proves knowledge of a private key sk to a *verifier*.

Definition 2 (Canonical Three-move Identification-from-Zero-knowledge Scheme) *A* canonical three-move Identification-from-Zero-knowledge scheme *(see Figure 2) is a 4-tuple of algorithms* $IZ = (IZ.PG, IZ.KG, IZ.P = (IZ.P_1, IZ.P_2), IZ.Ver)$ *where:*

- *The randomized parameter generation algorithm* $IZ.PG$ *takes a security parameter* 1^κ *as input and returns system parameters* param.
- *The randomized key generation algorithm* $IZ.KG$ *takes* param *as input and creates a key pair* (pk, sk). *In turn, the public key defines challenge spaces* $(\mathcal{C}, \mathcal{D}) := \mathcal{CD}(pk)$, *and is known to all parties.*
- *The first, randomized prover algorithm* $IZ.P_1$ *takes as input the secret key* sk *and returns commitments* $(\hat{u}, \hat{v})$ *and the prover's state* st_P.
- *The second, deterministic prover algorithm* $IZ.P_2$ *takes as input the prover's state* st_P, *which includes the secret key* sk *and the commitments* $(\hat{u}, \hat{v})$, *together with challenges* $(\hat{c} \in \mathcal{C}, \hat{d} \in \mathcal{D})$, *and returns the response* $\hat{w}$.
- *The deterministic verification algorithm* $IZ.Ver$ *takes as input the public key* pk, *commitments* $(\hat{u}, \hat{v})$, *challenges* $(\hat{c} \in \mathcal{C}, \hat{d} \in \mathcal{D})$, *and response* $\hat{w}$, *and returns* $b = 1$ *to indicate acceptance or* $b = 0$ *to indicate rejection.*

3. The *BZ*[DL] and *IZ*[DL] schemes

The conventional $BZ[\mathsf{DL}]$ blind signature scheme and its associated identification scheme $IZ[\mathsf{DL}]$ are defined over a cryptographically strong group, typically that of a large prime order subgroup of points on an elliptic curve, where the DL problem is considered hard.

The $BZ[\mathsf{DL}]$ scheme consists of the following algorithms:

Algorithm 1 $BZ[\mathsf{DL}].PG(1^\kappa)$

Select an Abelian group $\mathbb{G} := \langle G \rangle$ of prime order $n \approx 2^{2\kappa}$ and secure hash functions $\mathcal{H} : \mathbb{G} \times \{0,1\}^* \to \mathbb{Z}_n^*$ and $\mathcal{G} : \mathbb{G} \to \mathbb{Z}_n^*$.
return $\mathsf{param} := (n, G, \mathcal{G}, \mathcal{H})$

Algorithm 2 $BZ[\mathsf{DL}].KG(\mathsf{param})$

$(n, G, \mathcal{G}, \mathcal{H}) \leftarrow \mathsf{param}$
$x \xleftarrow{\$} \mathbb{Z}_n^*, \quad Y \leftarrow [x]G$
$sk \leftarrow (x, \mathsf{param}), \quad pk \leftarrow (Y, \mathsf{param})$
return (pk, sk)

Algorithm 3 $BZ[\mathsf{DL}].S_1(sk)$

$(x, \mathsf{param}) \leftarrow sk, \quad (n, G, \mathcal{G}, \mathcal{H}) \leftarrow \mathsf{param}$
$r \xleftarrow{\$} \mathbb{Z}_n^*, \quad s \xleftarrow{\$} \mathbb{Z}_n^*, \quad \hat{U} \leftarrow [r]G, \quad \hat{V} \leftarrow [s]G, \quad \mathsf{st}_S \leftarrow (sk, r, s)$
return $(\hat{U} \in \mathbb{G}^*, \hat{V} \in \mathbb{G}^*, \mathsf{st}_S)$

Algorithm 4 $BZ[\mathsf{DL}].U_1(pk, \hat{U} \in \mathbb{G}^*, \hat{V} \in \mathbb{G}^*, m \in \{0,1\}^*)$

$(Y, \mathsf{param}) \leftarrow pk, \quad (n, G, \mathcal{G}, \mathcal{H}) \leftarrow \mathsf{param}$
$\pi \xleftarrow{\$} \mathbb{Z}_n^*, \quad \delta \xleftarrow{\$} \mathbb{Z}_n^*, \quad \rho \xleftarrow{\$} \mathbb{Z}_n^*, \quad \varepsilon \xleftarrow{\$} \mathbb{Z}_n^*$
▷ $\zeta \leftarrow \rho\pi \pmod n$, compute $\zeta^{-1} \pmod n$, $\pi^{-1} \leftarrow \rho\zeta^{-1} \pmod n$, $\rho^{-1} \leftarrow \pi\zeta^{-1} \pmod n$
$U \leftarrow [\pi]\hat{U} + [\delta]G, \quad V \leftarrow [\rho\pi]\hat{V} + [\varepsilon]G$
$c \leftarrow \mathcal{H}(U, m), \quad d \leftarrow \mathcal{G}(V), \quad \hat{c} \leftarrow c\pi^{-1} \pmod n, \quad \hat{d} \leftarrow d\rho^{-1} \pmod n$
$\mathsf{st}_U \leftarrow (pk, U, \hat{U}, \hat{V}, \pi, \delta, \rho, \varepsilon, d, \hat{c}, \hat{d})$
return $(\hat{c} \in \mathbb{Z}_n^*, \hat{d} \in \mathbb{Z}_n^*, \mathsf{st}_U)$

Algorithm 5 $BZ[\mathsf{DL}].S_2(\mathsf{st}_S, \hat{c} \in \mathbb{Z}_n^*, \hat{d} \in \mathbb{Z}_n^*)$

$(sk, r, s) \leftarrow \mathsf{st}_S, \quad (x, \mathsf{param}) \leftarrow sk, \quad (n, G, \mathcal{G}, \mathcal{H}) \leftarrow \mathsf{param}$
$z \leftarrow r - \hat{c}x \pmod n, \quad \hat{w} \leftarrow s - \hat{d}z \pmod n$
return $\hat{w} \in \mathbb{Z}_n^*$

Algorithm 6 $BZ[\mathsf{DL}].U_2(\mathsf{st}_U, \hat{w} \in \mathbb{Z}_n^*)$

$(pk, U, \hat{U}, V, \hat{V}, \pi, \delta, \rho, \varepsilon, d, \hat{c}, \hat{d}) \leftarrow \mathsf{st}_U, (Y, \mathsf{param}) \leftarrow pk, (n, G, \mathcal{G}, \mathcal{H}) \leftarrow \mathsf{param}$
▷ check that the signer is honest:
$\hat{H} \leftarrow \hat{U} - [\hat{c}]Y, \quad$ **if** $\hat{V} \neq [\hat{w}]G + [\hat{d}]\hat{H}$ **: return** $\perp$
$w \leftarrow \rho\pi\hat{w} - d\delta + \varepsilon \pmod n$
return $\sigma := (U, d, w) \in \mathbb{G}^* \times (\mathbb{Z}_n^*)^2$

Algorithm 7 $BZ[\mathsf{DL}].Ver(pk, \sigma \in \mathbb{G}^* \times (\mathbb{Z}_n^*)^2, m \in \{0,1\}^*)$

if $\sigma = \perp$ **: return** 0
$(Y, \mathsf{param}) \leftarrow pk, \quad (n, G, \mathcal{G}, \mathcal{H}) \leftarrow \mathsf{param}$
$(U, d, w) \leftarrow \sigma, \quad c \leftarrow \mathcal{H}(U, m), \quad H \leftarrow U - [c]Y, \quad V \leftarrow [w]G + [d]H$
return $(d = \mathcal{G}(V))$ **?** 1 **:** 0

The BZ protocol can be proved to be perfectly correct, perfectly blind, and one-more-unforgeable in the random oracle model under the *one-more discrete logarithm* assumption (OMDL) [Bellare and Palacio 2002], whereby an adversary is given $\ell + 1$ elements from $\langle G \rangle$, allowed to perform ℓ queries to an oracle capable of solving an instance of the DL problem in $\langle G \rangle$ (not necessarily from the set of received elements), and challenged to produce the discrete logarithms of all $\ell + 1$ group elements it has received (the OMDL assumption being precisely that this problem is unfeasible).

For the $IZ[\mathsf{DL}]$ scheme, algorithms $IZ.PG$ and $IZ.KG$ are, respectively, identical

to $BZ.PG$ and $BZ.KG$, except that the hash functions are ignored and omitted. The remaining algorithms are as follows.

Algorithm 8 $IZ[\mathsf{DL}].P_1(sk)$

1: $(x, \mathsf{param}) \leftarrow sk$, $(n, G) \leftarrow \mathsf{param}$, $r \xleftarrow{\$} \mathbb{Z}_n^*$, $s \xleftarrow{\$} \mathbb{Z}_n^*$, $\hat{U} \leftarrow [r]G$, $\hat{V} \leftarrow [s]G$, $\mathsf{st}_P \leftarrow (sk, r, s)$
2: **return** $(\hat{U} \in \mathbb{G}^*, \hat{V} \in \mathbb{G}^*, \mathsf{st}_P)$

Algorithm 9 $IZ[\mathsf{DL}].P_2(\mathsf{st}_P, \hat{c} \in \mathbb{Z}_n^*, \hat{d} \in \mathbb{Z}_n^*)$

1: $(sk, r, s) \leftarrow \mathsf{st}_P$, $(x, \mathsf{param}) \leftarrow sk$, $(n, G) \leftarrow \mathsf{param}$, $z \leftarrow r - \hat{c}x \pmod n$, $\hat{w} \leftarrow s - \hat{d}z \pmod n$
2: **return** $\hat{w} \in \mathbb{Z}_n^*$

Algorithm 10 $IZ[\mathsf{DL}].Ver(pk, \hat{U} \in \mathbb{G}^*, \hat{V} \in \mathbb{G}^*, \hat{c} \in \mathbb{Z}_n^*, \hat{d} \in \mathbb{Z}_n^*, \hat{w} \in \mathbb{Z}_n^*)$

1: $(Y, \mathsf{param}) \leftarrow pk$, $(n, G) \leftarrow \mathsf{param}$, $\hat{H} \leftarrow \hat{U} - [\hat{c}]Y$
2: **return** $\left(\hat{V} = [\hat{w}]G + [\hat{d}]\hat{H}\right)$ **?** 1 **:** 0

The $IZ[\mathsf{qDL}]$ scheme cannot attain the standard meddler-in-the-middle (MitM) security [Barreto and Zanon 2023, Thm. 3], but it can be shown to achieve impersonation security against concurrent attacks (IMP-CA) [Bellare and Palacio 2002] under the OMDL assumption [Barreto and Zanon 2023, Thm. 5], in the sense that breaking $BZ[\mathsf{DL}]$ OMUF implies breaking $IZ[\mathsf{DL}]$ IMP-CA, and this in turn implies solving OMDL.

4. The *BZ*[qDL] and *IZ*[qDL] schemes

The Kummer-based $BZ[\mathsf{qDL}]$ blind signature scheme we propose consists of the following algorithms. The protocol structure closely follows the plain elliptic version $BZ[\mathsf{DL}]$, with similar roles being indicated by the same color, but there are some significant differences due to the absence of a full group structure in the Kummer variety.

Certain operations that look similar require particular attention: in the comments withing the algorithms, we describe what Kummer-specific computations correspond to the formal specifications (e.g., the extended Montgomery ladder computes additional information, essentially for free, that is necessary for further protocol processing):

Algorithm 11 $BZ[\mathsf{qDL}].PG(1^\kappa)$

1: Select a Kummer variety $\mathfrak{G}$ from an underlying group of prime order $n \approx 2^{2\kappa}$, a generator $\pm G \in \mathfrak{G}$, and secure hash functions $\overline{\mathcal{H}} : \mathfrak{G} \times \{0,1\}^* \to \mathbb{Z}_n^*$ and $\overline{\mathcal{G}} : \mathfrak{G} \to \mathbb{Z}_n^*$.
2: **return** $\mathsf{param} := (n, \pm G, \overline{\mathcal{G}}, \overline{\mathcal{H}})$

Algorithm 12 $BZ[\mathsf{qDL}].KG(\mathsf{param})$

1: $(n, \pm G, \overline{\mathcal{G}}, \overline{\mathcal{H}}) \leftarrow \mathsf{param}$
2: $x \xleftarrow{\$} \mathbb{Z}_n^*, \quad \pm Y \leftarrow \pm[x]G \quad \triangleright \pm Y \leftarrow \mathsf{xML}(x, \pm G)$
3: $sk \leftarrow (x, \mathsf{param}), \quad pk \leftarrow (\pm Y, \mathsf{param})$
4: **return** (pk, sk)

Algorithm 13 $BZ[\mathsf{qDL}].S_1(sk)$

1: $(x, \mathsf{param}) \leftarrow sk, \quad (n, \pm G, \overline{\mathcal{G}}, \overline{\mathcal{H}}) \leftarrow \mathsf{param}$
2: $r \xleftarrow{\$} \mathbb{Z}_n^*, \quad s \xleftarrow{\$} \mathbb{Z}_n^*$
$\triangleright$ $(\pm\hat{\Delta}_U, \pm\hat{U}) \leftarrow \mathsf{xML}^+(r-1, \pm G), \quad (\pm\hat{\Delta}_V, \pm\hat{V}) \leftarrow \mathsf{xML}^+(s-1, \pm G)$
3: $\pm\hat{U} \leftarrow \pm[r]G, \quad \pm\hat{\Delta}_U \leftarrow \pm(\hat{U} - G), \quad \pm\hat{V} \leftarrow \pm[s]G, \quad \pm\hat{\Delta}_V \leftarrow \pm(\hat{V} - G), \quad \mathsf{st}_S \leftarrow (sk, r, s)$
4: **return** $(\pm\hat{U} \in \mathfrak{G}^*, \pm\hat{\Delta}_U \in \mathfrak{G}^*, \pm\hat{V} \in \mathfrak{G}^*, \pm\hat{\Delta}_V \in \mathfrak{G}^*, \mathsf{st}_S)$

Algorithm 14 $BZ[\mathsf{qDL}].U_1(pk, \pm\hat{U} \in \mathfrak{G}^*, \pm\hat{\Delta}_U \in \mathfrak{G}^*, \pm\hat{V} \in \mathfrak{G}^*, \pm\hat{\Delta}_V \in \mathfrak{G}^*, m \in \{0,1\}^*)$

1: $(\pm Y, \mathsf{param}) \leftarrow pk, \quad (n, \pm G, \overline{\mathcal{G}}, \overline{\mathcal{H}}) \leftarrow \mathsf{param}$
$\triangleright$ check that $\pm\hat{\Delta}_U = \pm\hat{U} \pm G$ and $\pm\hat{\Delta}_V = \pm\hat{V} \pm G$:
$\triangleright$ **if not** $\mathsf{xRS}(\pm\hat{U}, \pm\hat{\Delta}_U, \overline{\pm G})$ **or not** $\mathsf{xRS}(\pm\hat{V}, \pm\hat{\Delta}_V, \overline{\pm G})$ **:**
2: **if** $\pm\hat{\Delta}_U \neq \pm\hat{U} \pm G$ **or** $\pm\hat{\Delta}_V \neq \pm\hat{V} \pm G$ **: return** $\perp$
3: $\pi \xleftarrow{\$} \mathbb{Z}_n^*, \quad \delta \xleftarrow{\$} \mathbb{Z}_n^*, \quad \rho \xleftarrow{\$} \mathbb{Z}_n^*, \quad \varepsilon \xleftarrow{\$} \mathbb{Z}_n^*$
$\triangleright$ $\zeta \leftarrow \rho\pi \pmod n$, compute $\zeta^{-1} \pmod n$, $\pi^{-1} \leftarrow \rho\zeta^{-1} \pmod n$, $\rho^{-1} \leftarrow \pi\zeta^{-1} \pmod n$
$\triangleright$ $\pm U \leftarrow \mathsf{xML2D}(\pi, \pi^{-1}, \delta, \pm\hat{U}, \pm G, \pm\hat{\Delta}_U), \quad \pm V \leftarrow \mathsf{xML2D}(\zeta, \zeta^{-1}, \varepsilon, \pm\hat{V}, \pm G, \pm\hat{\Delta}_V)$
4: $\pm U \leftarrow \pm[\pi]\hat{U} \pm [\delta]G, \quad \pm V \leftarrow \pm[\rho\pi]\hat{V} \pm [\varepsilon]G \quad \triangleright$ **if** $\pm U = O$ **or** $\pm V = O$ **: restart at** step 3
5: $c \leftarrow \overline{\mathcal{H}}(\pm U, m), \quad d \leftarrow \overline{\mathcal{G}}(\pm V), \quad \hat{c} \leftarrow c\pi^{-1} \pmod n, \quad \hat{d} \leftarrow d\rho^{-1} \pmod n$
6: $\mathsf{st}_U \leftarrow (pk, \pm\hat{U}, \pm\hat{V}, \pm U, \pm V, \pi, \delta, \rho, \varepsilon, d, \hat{c}, \hat{d}, \zeta)$
7: **return** $(\hat{c} \in \mathbb{Z}_n^*, \hat{d} \in \mathbb{Z}_n^*, \mathsf{st}_U)$

Algorithm 15 $BZ[\mathsf{qDL}].S_2(\mathsf{st}_S, \hat{c} \in \mathbb{Z}_n^*, \hat{d} \in \mathbb{Z}_n^*)$

1: $(sk, r, s) \leftarrow \mathsf{st}_S, \quad (x, \mathsf{param}) \leftarrow sk, \quad (n, \pm G, \overline{\mathcal{G}}, \overline{\mathcal{H}}) \leftarrow \mathsf{param}$
$\triangleright$ ensure that *both* possible signatures are consistent:
2: **if** $r \pm \hat{c}x \equiv 0 \pmod n$ **or** $s \pm \hat{d}z \equiv 0 \pmod n$ **: return** $\perp$ $\quad \triangleright$ Kummer variety constraint violation
3: $z \leftarrow r - \hat{c}x \pmod n, \quad \hat{w} \leftarrow s - \hat{d}z \pmod n$
4: **return** $\hat{w} \in \mathbb{Z}_n^*$

Algorithm 16 $BZ[\mathsf{qDL}].U_2(\mathsf{st}_U, \hat{w} \in \mathbb{Z}_n^*)$

1: $(pk, \pm\hat{U}, \pm\hat{V}, \pm U, \pm V, \pi, \delta, \rho, \varepsilon, d, \hat{c}, \hat{d}, \zeta) \leftarrow \mathsf{st}_U, \quad (\pm Y, \mathsf{param}) \leftarrow pk, \quad (n, \pm G, \overline{\mathcal{G}}, \overline{\mathcal{H}}) \leftarrow \mathsf{param}$
$\triangleright$ check that the signer is honest:
$\triangleright$ **if not** $\mathsf{xBQ}(\mathsf{xML}(\hat{w}, \pm G), \mathsf{xML}(\hat{d}, \pm\hat{U}), \mathsf{xML}(-\hat{c}\hat{d} \pmod n, \pm Y), \pm\hat{V})$ **:**
2: **if** $\pm\hat{V} \neq \pm[\hat{w}]G \pm [\hat{d}]\hat{U} \pm [-\hat{c}\hat{d}]Y$ **: return** $\perp$
$\triangleright$ ensure that *all* possible signatures are consistent (otherwise the protocol needs to restart from the beginning):
3: **if** $\zeta\hat{w} \pm d\delta \pm \varepsilon = 0 \pmod n$ **: return** $\perp$
4: $w \leftarrow \zeta\hat{w} - d\delta + \varepsilon \pmod n$
5: **return** $\sigma := (\pm U, \pm V, w) \in (\mathfrak{G}^*)^2 \times \mathbb{Z}_n^*$

Algorithm 17 $BZ[\mathsf{qDL}].Ver(pk, \sigma \in (\mathfrak{G}^*)^2 \times \mathbb{Z}_n^*, m \in \{0,1\}^*)$

1: **if** $\sigma = \perp$ **: return** 0
2: $(\pm Y, \mathsf{param}) \leftarrow pk, \quad (n, \pm G, \overline{\mathcal{G}}, \overline{\mathcal{H}}) \leftarrow \mathsf{param}, \quad (\pm U, \pm V, w) \leftarrow \sigma, \quad c \leftarrow \overline{\mathcal{H}}(\pm U, m), \quad d \leftarrow \overline{\mathcal{G}}(\pm V)$
$\triangleright$ $(\mathsf{xBQ}(\mathsf{xML}(w, \pm G), \mathsf{xML}(d, \pm U), \mathsf{xML}(-cd \pmod n, \pm Y), \pm V))$ **?** 1 **:** 0
3: **return** $(\pm V = \pm[w]G \pm [d]U \pm [-cd]Y)$ **?** 1 **:** 0

The security properties of this protocol are a direct restatement of the DL version,

with the OMDL problem replaced by the version induced by the Kummer variety, the so-called OMKDL assumption. This will be formulated in detail in Section 4.2.

For the associated identification scheme $IZ[\mathsf{qDL}]$, algorithms $IZ[\mathsf{qDL}].PG$ and $IZ[\mathsf{qDL}].KG$ are, respectively, identical to $BZ[\mathsf{qDL}].PG$ and $BZ[\mathsf{qDL}].KG$, except that the hash functions are ignored and omitted. The remaining algorithms are as follows.

Algorithm 18 $IZ[\mathsf{qDL}].P_1(sk)$

$(x, \mathsf{param}) \leftarrow sk$, $(n, \pm G) \leftarrow \mathsf{param}$, $r \xleftarrow{\$} \mathbb{Z}_n^*$, $s \xleftarrow{\$} \mathbb{Z}_n^*$
$\pm\hat{U} \leftarrow \pm[r]G$, $\pm\hat{V} \leftarrow \pm[s]G$, $\mathsf{st}_P \leftarrow (sk, r, s)$ ▷ $\pm U \leftarrow \mathsf{xML}(r, \pm G)$, $\pm V \leftarrow \mathsf{xML}(s, \pm G)$
return $(\pm\hat{U} \in \mathfrak{G}^*, \pm\hat{V} \in \mathfrak{G}^*, \mathsf{st}_P)$

Algorithm 19 $IZ[\mathsf{qDL}].P_2(\mathsf{st}_P, \hat{c} \in \mathbb{Z}_n^*, \hat{d} \in \mathbb{Z}_n^*)$

$(sk, r, s) \leftarrow \mathsf{st}_P$, $(x, \mathsf{param}) \leftarrow sk$, $(n, \pm G) \leftarrow \mathsf{param}$
▷ ensure that *both* possible signatures are consistent:
if $r \pm \hat{c}x \equiv 0 \pmod n$ **or** $s \pm \hat{d}z \equiv 0 \pmod n$ **: return** $\perp$ ▷ Kummer variety constraint violation
$z \leftarrow r - \hat{c}x \pmod n$, $\hat{w} \leftarrow s - \hat{d}z \pmod n$
return $\hat{w} \in \mathbb{Z}_n^*$

Algorithm 20 $IZ[\mathsf{qDL}].Ver(pk, \pm\hat{U} \in \mathfrak{G}^*, \pm\hat{V} \in \mathfrak{G}^*, \hat{c} \in \mathbb{Z}_n^*, \hat{d} \in \mathbb{Z}_n^*, \hat{w} \in \mathbb{Z}_n^*)$

$(\pm Y, \mathsf{param}) \leftarrow pk$, $(n, \pm G) \leftarrow \mathsf{param}$
return $\left(\pm\hat{V} = \pm[\hat{w}]G \pm [\hat{d}]\hat{U} \pm [-\hat{c}\hat{d}]Y\right)$ **? 1 : 0** ▷ $\left(\mathsf{xBQ}\left(\pm[\hat{w}]G, \pm[\hat{d}]\hat{U}, \pm[-\hat{c}\hat{d} \pmod n]Y, \pm\hat{V}\right)\right)$ **? 1 : 0**

4.1. Differences between the DL and qDL protocols

We see that there is a communication difference between the blind signature and identification protocols, namely, the signature scheme needs to transmit extra points $\pm\hat{\Delta}_U$ and $\pm\hat{\Delta}_V$ to enable laddering. This does not create a leakage opportunity, since the extra points only differ from the essential points $\pm\hat{U}$ and $\pm\hat{V}$ by an offset $\pm G$, and could be recomputed from public data. This, however, would incur a substantial computational overhead, since adding or subtracting the offset requires the full point coordinates to be completed, defeating the goal of keeping only x-coordinates throughout the whole scheme.

Thus, the extra points are only transmitted to allow for an efficiency trade-off. Yet, they must still be checked for consistency (that is, they must have the required offset form), and this is why the Renes-Smith test must be applied to them. This check is still far more efficient than recovering the full coordinates.

4.2. Security

The proposed schemes satisfy analogous security properties as the $BZ[\mathsf{DL}]$ and $IZ[\mathsf{DL}]$ schemes upon which they are based, under the Kummer-based assumptions corresponding to their discrete-logarithm counterparts.

Namely, the $BZ[\mathsf{qDL}]$ protocol can be proved to be perfectly correct, perfectly blind in the random oracle model, and one-more-unforgeable (OMUF) under the *one-more Kummer discrete-logarithm assumption* (OMKDL), captured in Definition 3.

Definition 3 *Let* $\mathfrak{G} := \langle G \rangle / \pm$ *be a Kummer variety. Let an adversary be given* $\ell + 1$ *elements from* $\mathfrak{G}$*, then allowed to perform* ℓ *queries to an oracle capable of solving an instance of the Kummer DL problem in* $\mathfrak{G}$ *(not necessarily from the set of received*

elements), and finally challenged to produce the Kummer discrete logarithms of all $\ell + 1$ *group elements it has received. The* one-more Kummer discrete-logarithm *assumption (OMKDL) is that properly responding to this challenge is unfeasible.*

As pointed out in Section 2, OMKDL and OMDL are equivalent.

The proofs are straightforward, word-for-word restatements of Theorems 1 (for perfect correctness), 2 (for perfect blindness), 4 and 5 (for OMUF of $BZ[\mathsf{qDL}]$ and IMP-CA of $IZ[\mathsf{qDL}]$) from [Barreto and Zanon 2023], changing all instances of the discrete logarithm problem by the corresponding Kummer instances, and changing the OMDL assumption by OMKDL. Correspondingly, the $IZ[\mathsf{qDL}]$ scheme cannot attain standard MitM security, but it can be shown to achieve IMP-CA security under the OMKDL assumption, in the sense that breaking the OMUF security of $BZ[\mathsf{qDL}]$ implies breaking the IMP-CA security of $IZ[\mathsf{qDL}]$, and this in turn implies solving OMKDL. These properties stem, as before, from word-for-word restatements of Theorems 3 and 5 from [Barreto and Zanon 2023], respectively.

One of the more remarkable properties of the BZ methodology, which is also present in the proposed Kummer version, is its resistance against ROS attacks [Benhamouda et al. 2021]. More specifically, given a prime number p and access to a random oracle H_{ros} with range in $\mathbb{Z}_p$, the ROS problem (in dimension ℓ) asks to find $(\ell + 1)$ vectors $\hat{\rho}_i \in \mathbb{Z}_p^\ell$ for $i \in [\ell + 1]$, and a vector $\boldsymbol{c} = (c_1, \ldots, c_\ell)$ such that: $H_{ros}(\hat{\boldsymbol{\rho}}_i) = \langle \hat{\boldsymbol{\rho}}_i, \boldsymbol{c} \rangle$ for all $i \in [\ell + 1]$. In practice, using a solver for the ROS problem, many blind signature schemes can be attacked in polynomial time if the attacker can interact with the signer via $\ell > \lg p$ concurrent sessions. However, at best, such attacks appear to be *harder* to mount against $BZ[\mathsf{qDL}]$ than in the conventional $BZ[\mathsf{DL}]$ case. This is a consequence of the sign ambiguity in the Kummer version: linear combinations involving a number ℓ of terms in the DL case branch into 2^ℓ terms that differ in the sign, and the attacker would have to guess somehow which sign assignments are correct. It is not clear at this time how (or whether) this particular obstacle can be overcome, but even if it can, the result is still the same situation as in $BZ[\mathsf{DL}]$, which is argued to be infeasible in [Barreto and Zanon 2023, Appendix A].

For brevity (and sheer lack of space), those lengthy proofs are omitted here. We refer the interested reader to the original proofs as indicated above.

Finally, from the point of view of *concrete* (practical) security there is no more than a 1-bit decrease compared to qDSA due to the need to accept twice as many signature verification relations as equally valid. Like qDSA, there is no more than a further 1-bit decrease compared to a conventional Schnorr-like signature in a setting that supports full group arithmetic.

5. The biquadratic test

At the core of the Kummer-based versions of IZ and BZ is the requirement to determine if $\pm T = \pm P \pm Q \pm R$ for given points $\pm P$, $\pm Q$, $\pm R$, and $\pm T$, which generalizes the Renes-Smith algorithm to perform the related check $\pm R = \pm P \pm Q$.

Let $P = [X_P : Z_P]$, $Q = [X_Q : Z_Q]$, $R = [X_R : Z_R]$, and $T = [X_T : Z_T]$. The Renes-Smith idea to test whether $aX_R^2 - 2bX_RZ_R + cZ_R^2 = 0$, where $a := (X_PZ_Q -$

$Z_P X_Q)^2$, $b := (X_P X_Q + Z_P Z_Q)(X_P Z_Q + Z_P X_Q) + 2A X_P Z_P X_Q Z_Q$, and $c := (X_P X_Q - Z_P Z_Q)^2$.

The generalized biquadratic check, which can be written $\pm R \pm T = \pm P \pm Q$, could be performed by applying the Renes-Smith technique twice if the coordinates of $\pm S := \pm(R+T)$ and $\pm D := \pm(R-T)$ were known, but obtaining both (as opposed to computing only one) would incur the additional computational effort of a full multiplication by scalar. Yet, one can proceed along this line and look for a biquadratic formula corresponding to the combined test $(aX_S^2 - 2bX_S Z_S + cZ_S^2)(aX_D^2 - 2bX_D Z_D + cZ_D^2) = 0$, so as to check if either the sum or the difference between $\pm R$ and $\pm T$ corresponds to $\pm P \pm Q$.

Starting with normalized coordinates $\pm S = [x_S : 1]$, $\pm D = [x_D : 1]$, $R = [x_R : 1]$, and $T = [x_T : 1]$ (to be homogenized later), and assuming provisionally that $\pm O \neq \pm R \neq \pm T \neq \pm O$, the biquadratic formula reads:

$$a^2(x_S x_D)^2 + 4b^2(x_S x_D) - 2b(x_S + x_D)(a(x_S x_D) + c) + ac(x_S^2 + x_D^2) + c^2 = 0. \quad (1)$$

Applying the Montgomery group law yields[1]

$$\begin{aligned} x_S x_D &= (x_R x_T - 1)^2/(x_R - x_T)^2, \\ x_S + x_D &= 2((x_R x_T + 1)(x_R + x_T) + 2A x_R x_T)/(x_R - x_T)^2, \\ x_S^2 + x_D^2 &= 2(4x_R x_T(2A(A x_R x_T + (x_R x_T + 1)(x_R + x_T)) + (x_R x_T - 1)^2) \\ &\quad + ((x_R x_T + 3)^2 - 8)(x_R + x_T)^2)/(x_R - x_T)^4. \end{aligned}$$

These expressions can now be plugged into Eq. 1 and the common denominator $(x_R - x_T)^4$ can be dropped. Homogenizing back by substituting $x_R := X_R/Z_R$ and $x_T := X_T/Z_T$ and then dropping a further common denominator $(Z_R Z_T)^6$, we finally get the desired biquadratic formula:

$$a^2 v^4 + 4b^2 v^2 z^2 - 4b(uw + 2Ade)(av^2 + cz^2) + act + c^2 z^4 = 0 \quad (2)$$

where $d := X_R X_T$, $e := Z_R Z_T$, $u := d + e$, $v := d - e$, $f := X_R Z_T$, $g := X_T Z_R$, $w := f + g$, $z := f - g$, and $t := 2(4de(2A(Ade + uw) + v^2) + ((u + 2e)^2 - 8e^2)w^2)$.

As it happens for the Renes-Smith test, this formula holds in full generality, including when $\pm R = \pm O$, $\pm T = \pm O$, or $\pm R = \pm T$, and it vanishes precisely when $\pm T = \pm P \pm Q \pm R$. This is summarized in Algorithm 31 in Appendix B.

5.1. Analysis

Unsurprisingly, the biquadratic test above clearly looks much more involved than the basic Renes-Smith test, since it must cope with one more implicit differential addition and twice as many equivalent acceptance conditions.

Yet, implementing it costs no more than 20 general multiplications and 9 squarings in the underlying finite field (plus a few much lighter operations like additions/subtractions and shifts). This represents about four times the cost of the Renes-Smith test[2] which, at 5 general multiplications and 2 squarings, is itself slightly less than

[1] Remark: a computationally simpler expression for $x_S^2 + x_D^2$ may conceivably exist.

[2] In general, the biquadratic test costs about three times as much as Renes-Smith, but the latter can be optimized in the context of BZ[qDL].

the cost of one joint pseudo-doubling and differential addition (xDBLADD) on the curve, namely 6 general multiplications and 4 squarings.

Besides, the biquadratic test is only ever invoked accompanied by three laddering operations (in Algorithms 16, 17, and 20). Since the total number of xDBLADD operations performed within each laddering matches the scalar bitlength (i.e. twice the security bit-level κ), the runtime overhead of the biquadratic test is no more than a fraction about $1/(2\kappa)$. For instance, at the 128-bit security level the biquadratic test is only responsible for a fraction about $1/256$ of the runtime, and even less at higher security levels. Thus, despite its complexity, it is lightweight in the context of the proposed protocols.

6. Experimental assessment

We developed a proof-of-concept implementation of the proposed blind signature protocol $BZ[\mathsf{qDL}]$ in Python. With it, we conducted tests on the sample Montgomery curve $E : y^2 = x^3 - 61370x^2 + x$ over $\mathbb{F}_{2^{256}-189}$ (the so-called `ed-256-mers` curve from [Bos et al. 2016]). For comparison, we have also implemented the conventional $BZ[\mathsf{DL}]$ protocol on the same curve `ed-256-mers` in Edwards form, with similarly constant-time laddering.

The computational costs are summarized on Table 1, in the form of operation counts for each of the algorithms of a canonical three-move blind-signature-from-zero-knowledge scheme except PG (since picking a suitable curve and hash functions transcends the scope of a particular protocol that uses them). The smaller costs between $BZ[\mathsf{DL}]$ and $BZ[\mathsf{qDL}]$ are highlighted in **boldface**. These counts were obtained using a constant-time and uniform-access implementation, where all scalar factors are constrained to have the same bitlength.

We limit the statistics to the most relevant arithmetic operations: finite field multiplications (`M`), finite field squarings (`S`), and multiplications by small constants A, $2A$ or $(A+2)/4$ (`C`). We also include finite field inversions (`I`), since they are required when normalizing curve points $[X : Z]$ as $[X/Z : 1]$, specifically for serialization as byte arrays and transmission. We tally multiplications and inversions $\bmod p$ and $\bmod n$ together, as these moduli are always of comparable size.

Conventional $BZ[\mathsf{DL}]$ was already more efficient than the main proposal known to resist ROS attacks [Benhamouda et al. 2021], namely the general-purpose, DL-based blind signature scheme by Tessaro-Zhu BS_3 [Tessaro and Zhu 2022] (whose signatures are $2\times$ shorter than Abe's scheme [Kastner et al. 2022], for example). Hence, we conjecture that our proposal is the most efficient scheme known heretofore in the same category.

Table 1. Operation counts for BZ[DL] and BZ[qDL] per protocol execution

	M		S		C		I	
algorithm	DL	qDL	DL	qDL	DL	qDL	DL	qDL
KG	3319	**1267**	1024	**1014**	255	**254**	0	0
S_1	6644	**2538**	2048	**2028**	510	**508**	**2**	4
S_2	**2**	4	0	0	0	0	0	0
U_1	13322	**6113**	4098	**4068**	1024	**1018**	10	**3**
U_2	9984	**4079**	3072	**3051**	767	**765**	**1**	2
Ver	9994	**4077**	3073	**3051**	768	**765**	7	**2**

7. Conclusion

We have described a blind signature scheme based on the Kummer variety of Montgomery curves. Our proposal follows the BZ methodology, in the sense that the signature obtained from the signer is replaced by a zero-knowledge argument of its possession, namely, a one-time signature created using (part of) that obtained signature as a private key. The scheme can be proven secure under the OMKDL assumption and resist the ROS attack that caused the demise of most elliptic-curve blind signature schemes. To support our proposal, we have tailored arithmetic algorithms and developed new ones that enable efficient implementation. Experimentally, the result turns out to be one of the most efficient among the few blind signature proposals that are still known to offer strong security guarantees.

References

Aranha, D. F., Novaes, F. R., Takahashi, A., Tibouchi, M., and Yarom, Y. (2020). LadderLeak: Breaking ECDSA with less than one bit of nonce leakage. In *ACM SIGSAC Conference on Computer and Communications Security (CCS 2020)*, pages 225–242. Association for Computing Machinery. DOI:10.1145/3372297.3417268.

Barreto, P. L. and Zanon, G. H. M. (2023). Blind signatures from Zero-knowledge arguments. Cryptology ePrint Archive, Paper 2023/067. https://eprint.iacr.org/2023/067.

Bellare, M. and Palacio, A. (2002). GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In *Advances in Cryptology – CRYPTO 2002*, pages 162–177. Springer. DOI:10.1007/3-540-45708-9_11.

Benhamouda, F., Lepoint, T., Loss, J., Orrù, M., and Raykova, M. (2021). On the (in)security of ROS. In *Advances in Cryptology – EUROCRYPT 2021*, pages 33–53. Springer. DOI:10.1007/978-3-030-77870-5_2.

Bos, J. W., Costello, C., Longa, P., and Naehrig, M. (2016). Selecting elliptic curves for cryptography: an efficiency and security analysis. *Journal of Cryptographic Engineering*, 6:259–286. 10.1007/s13389-015-0097-y.

Hauck, E., Kiltz, E., and Loss, J. (2019). A modular treatment of blind signatures from identification schemes. In *Advances in Cryptology – EUROCRYPT 2019*, pages 345–375. Springer. DOI:10.1007/978-3-030-17659-4_12.

Kastner, J., Loss, J., and Xu, J. (2022). On pairing-free blind signature schemes in the algebraic group model. In *IACR International Conference on Public-Key Cryptography*, pages 468–497. Springer.

Montgomery, P. L. (1987). Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of computation*, 48(177):243–264. DOI:10.1090/S0025-5718-1987-0866113-7.

Renes, J. and Smith, B. (2017). qDSA: Small and secure digital signatures with curve-based Diffie–Hellman key pairs. In *Advances in Cryptology – ASIACRYPT 2017*, pages 273–302. Springer. DOI:10.1007/978-3-319-70697-9_10.

Schnorr, C. P. (1990). Efficient identification and signatures for smart cards. In *Advances in Cryptology — CRYPTO' 89*, pages 239–252. Springer. DOI:10.1007/0-387-34805-0_22.

Tessaro, S. and Zhu, C. (2022). Short pairing-free blind signatures with exponential security. In *Advances in Cryptology – EUROCRYPT 2022*, pages 782–811. Springer. DOI:10.1007/978-3-031-07085-3_27.

A. Arithmetic of Montgomery curves

For convenience of reference, we recapitulate here the core arithmetic algorithms for Montgomery curves. We include the optimizations enabled within $BZ[\mathsf{qDL}]$.

The laddering multiplication-by-scalar Algorithms 21, 22, 23 and 24 use three functions, xADD for pseudo-addition, xDBL for pseudo-doubling and xDBLADD for joint pseudo-doubling and differential addition on a Montgomery curve (Algorithms 25, 26, and 27), as well as uniform conditional scalar swapping cSWP and point swapping xSWP (Algorithms 28 and 29).

We ensure that all scalars have the same bitlength $\ell := \lceil \lg n \rceil$ of the underlying group order, which is essential to attain a constant-time ladder, by flipping their sign modulo n when they are smaller than $\lceil n/2 \rceil$. This technique is tailored to Kummer varieties, and is slightly simpler and more efficient than the one suggested in [Aranha et al. 2020], which would require all scalars to be normalized to bitlength $\ell + 2$ instead.

Algorithm 21 The (extended) Montgomery ladder, $\mathsf{xML}^+(k, \pm P)$

INPUT: $k := \sum_{i=0}^{\ell-1} k_i 2^i$ with $0 \leq k \leq n$ for $n := 2|\mathfrak{G}| - 1$, $\ell := \lceil \lg n \rceil$, and $\pm P \in \mathfrak{G} \setminus \{O, T\}$, $T := (0, 0)$.
OUTPUT: $\pm[k]P \in \mathfrak{G}$ and $\pm[k+1]P \in \mathfrak{G}$.
COST: 1 call to xDBL, $\ell - 1$ calls to xDBLADD, $\ell - 1$ calls to xSWP.

$(k, \_) \leftarrow \mathsf{cSWP}(k, n - k, k \leq \lfloor n/2 \rfloor)$ ▷ make sure k is an ℓ-bit scalar
$(\pm R_0, \pm R_1) \leftarrow (\mathsf{xDBL}(\pm P), \pm P)$
for $i \leftarrow \ell - 2$ **downto** 0 **do**
    $(\pm R_0, \pm R_1) \leftarrow \mathsf{xSWP}(\pm R_0, \pm R_1, k_i \oplus k_{i+1})$
    $(\pm R_0, \pm R_1) \leftarrow \mathsf{xDBLADD}(\pm R_0, \pm R_1, \pm P)$
end for
$(\pm R_0, \pm R_1) \leftarrow \mathsf{xSWP}(\pm R_0, \pm R_1, k_0)$
return $(\pm R_0, \pm R_1)$

Algorithm 22 The Montgomery ladder, $\mathsf{xML}(k, \pm P)$

INPUT: $k := \sum_{i=0}^{\ell-1} k_i 2^i$ with $0 \le k \le n$ for $n := 2|\mathfrak{G}| - 1$, $\ell := \lceil \lg n \rceil$, and $\pm P \in \mathfrak{G} \setminus \{O, T\}$, $T := (0, 0)$.
OUTPUT: $\pm[k]P \in \mathfrak{G}$.

1: $(\pm R_0, \_) \leftarrow \mathsf{xML}^+(k, \pm P)$
2: **return** $\pm R_0$

Algorithm 23 The 3-point Montgomery ladder, $\mathsf{xML3}(m, \pm P, \pm Q, \pm(Q - P))$

INPUT: $k := \sum_{i=0}^{\ell-1} k_i 2^i$ with $0 \le k \le n$ for $n := 2|\mathfrak{G}| - 1$, $\ell := \lceil \lg n \rceil$, $\pm P \in \mathfrak{G}$, $\pm Q \in \mathfrak{G}$, and $\pm(Q - P) \in \mathfrak{G}$.
OUTPUT: $\pm(P + [k]Q) \in \mathfrak{G}$.

1: $(\pm R_0, \pm R_1, \pm R_2, \pm R_3) \leftarrow (\pm Q, \pm P, \pm(Q - P), \mathsf{xADD}(\pm P, \pm Q, \pm(Q - P)))$ ▷ $R_3 = \pm(Q + P)$
2: $(\pm R_2, \pm R_3) \leftarrow \mathsf{xSWP}(\pm R_2, \pm R_3, k \le \lfloor n/2 \rfloor)$, $(k, \_) \leftarrow \mathsf{cSWP}(k, n - k, k \le \lfloor n/2 \rfloor)$ ▷ make sure k is an ℓ-bit scalar
3: $k_{-1} \leftarrow 0$ ▷ not an actual bit from k: defined for notation convenience
4: **for** $i \leftarrow 0$ **to** $\ell - 1$ **do**
5: $\quad (\pm R_1, \pm R_2) \leftarrow \mathsf{xSWP}(\pm R_1, \pm R_2, k_i \oplus k_{i-1})$
6: $\quad (\pm R_0, \pm R_2) \leftarrow \mathsf{xDBLADD}(\pm R_0, \pm R_2, \pm R_1)$
7: **end for**
8: $(\pm R_1, \pm R_2) \leftarrow \mathsf{xSWP}(\pm R_1, \pm R_2, k_{\ell-1})$
9: **return** $\pm R_1$

Algorithm 24 The 2-dimensional Montgomery ladder, $\mathsf{xML2D}(\alpha, \alpha^{-1}, \beta, \pm P, \pm Q, \pm(Q - P))$

INPUT: $0 < \alpha < n$, $0 < \alpha^{-1} < n$, $0 \le \beta < n$ for $n := 2|\mathfrak{G}| - 1$, $\pm P \in \mathfrak{G}$, $\pm Q \in \mathfrak{G}$, and $\pm(Q - P) \in \mathfrak{G}$.
OUTPUT: $\pm[\alpha]P \pm [\beta]Q \in \mathfrak{G}$ (computed as $\pm[\alpha](\pm P \pm [\alpha^{-1}\beta]Q)$).

1: $\pm S \leftarrow \mathsf{xML3}(\alpha^{-1}\beta \pmod n, \pm P, \pm Q, \pm(Q - P))$, $\pm R \leftarrow \mathsf{xML}(\alpha, \pm S)$
2: **return** $\pm R$

Algorithm 25 Montgomery differential addition, $\mathsf{xADD}(\pm P, \pm Q, \pm(Q - P))$

INPUT: $\pm P := [X_P : Z_P], \pm Q := [X_Q : Z_Q], \pm(Q - P) := [X_- : Z_-] \in \mathfrak{G}$.
OUTPUT: $\pm(P + Q) := [X_+ : Z_+] \in \mathfrak{G}$ if $Q - P \notin \{O, (0, 0)\}$, otherwise $X_+ = Z_+ = 0$.
COST: $\mathtt{4M} + \mathtt{2S} + \mathtt{3a} + \mathtt{3s}$, or $\mathtt{3M} + \mathtt{2S} + \mathtt{3a} + \mathtt{3s}$ if Z_- is normalized to 1.

1: $V_0 \leftarrow X_P + Z_P$, $V_1 \leftarrow X_Q - Z_Q$, $V_1 \leftarrow V_1 \cdot V_0$, $V_0 \leftarrow X_P - Z_P$, $V_2 \leftarrow X_Q + Z_Q$, $V_2 \leftarrow V_2 \cdot V_0$
2: $V_0 \leftarrow V_1 + V_2$, $V_0 \leftarrow V_0^2$, $V_1 \leftarrow V_1 - V_2$, $V_1 \leftarrow V_1^2$, $X_+ \leftarrow Z_- \cdot V_0$, $Z_+ \leftarrow X_- \cdot V_1$
3: **return** $[X_+ : Z_+]$

Algorithm 26 Montgomery pseudo-doubling, $\mathsf{xDBL}(\pm P)$

INPUT: $\pm P := [X_P : Z_P] \in \mathfrak{G}$, $(A + 2)/4$: precomputed constant.
OUTPUT: $\pm[2]P := [X_2 : Z_2] \in \mathfrak{G}$ if $P \ne O$, otherwise $Z_2 = 0$.

1: $V_0 \leftarrow X_P + Z_P$, $V_0 \leftarrow V_0^2$, $V_1 \leftarrow X_P - Z_P$, $V_1 \leftarrow V_1^2$, $X_2 \leftarrow V_0 \cdot V_1$
2: $V_0 \leftarrow V_0 - V_1$, $V_2 \leftarrow ((A + 2)/4) \cdot V_0$, $V_2 \leftarrow V_2 + V_1$, $Z_2 \leftarrow V_0 \cdot V_2$
3: **return** $[X_2 : Z_2]$

Algorithm 27 Montgomery joint pseudo-doubling and differential addition $\mathsf{xDBLADD}(\pm P, \pm Q, \pm(Q - P))$

INPUT: $\pm P := [X_P : Z_P]$, $\pm Q := [X_Q : Z_Q]$, $\pm(Q - P) := [X_- : Z_-] \in \mathfrak{G}$, $(A + 2)/4$: precomputed constant.
OUTPUT: $\pm[2]P := [X_2 : Z_2] \in \mathfrak{G}$ and $\pm(P + Q) := [X_+ : Z_+] \in \mathfrak{G}$.
COST: $\mathtt{6M} + \mathtt{4S} + \mathtt{1C} + \mathtt{4a} + \mathtt{4s}$, or $\mathtt{5M} + \mathtt{4S} + \mathtt{1C} + \mathtt{4a} + \mathtt{4s}$ if Z_- is normalized to 1.

1: $V_0 \leftarrow X_P + Z_P$, $V_1 \leftarrow X_P - Z_P$, $V_2 \leftarrow V_0^2$, $V_3 \leftarrow V_1^2$, $X_2 \leftarrow V_2 \cdot V_3$, $V_2 \leftarrow V_2 - V_3$
2: $X_+ \leftarrow ((A + 2)/4) \cdot V_2$, $V_3 \leftarrow V_3 + X_+$, $Z_2 \leftarrow V_2 \cdot V_3$, $V_2 \leftarrow X_Q + Z_Q$, $V_3 \leftarrow X_Q - Z_Q$, $V_0 \leftarrow V_0 \cdot V_3$
3: $V_1 \leftarrow V_1 \cdot V_2$ $V_2 \leftarrow V_0 + V_1$, $V_2 \leftarrow V_2^2$, $V_3 \leftarrow V_0 - V_1$, $V_3 \leftarrow V_3^2$, $X_+ \leftarrow Z_- \cdot V_2$, $Z_+ \leftarrow X_- \cdot V_3$
4: $(X_+, Z_+) \leftarrow ((Z_P \mid X_P) = 0)$ **?** $\mathsf{cSWP}(X_Q, Z_Q, X_P = 0)$ **:** $\big((Z_Q \mid X_Q) = 0\big)$ **?** $\mathsf{cSWP}(X_P, Z_P, X_Q = 0)$ **:** $((Z_- \mid X_-) = 0)$ **?** $\mathsf{cSWP}(X_2, Z_2, X_- = 0)$ **:** O
▷ adjust (in isochronous fashion)
5: **return** $([X_2 : Z_2], [X_+ : Z_+])$

Algorithm 28 Isochronous conditional swapping, $\mathsf{cSWP}(u, v, swap)$

INPUT: $u, v \in \mathbb{Z}/2^m$; $swap \in \mathbb{Z}/2^m$. ▷ all quantities viewed as m-bit integers in two's complement
OUTPUT: (v, u) if $swap = 1$, otherwise (u, v).

1: $mask \leftarrow -swap$, $\Delta \leftarrow (u \oplus v)$ & $mask$
2: **return** $(u \oplus \Delta, v \oplus \Delta)$

Algorithm 29 Isochronous conditional swapping, xSWP$(\pm P, \pm Q, swap)$

INPUT: $\pm P = [X_P : Z_P]$, $\pm Q = [X_Q : Z_Q] \in \mathbb{Z}/2^m \times \mathbb{Z}/2^m$; $swap \in \mathbb{Z}/2^m$.
OUTPUT: $(\pm Q, \pm P)$ if $swap = 1$, otherwise $(\pm P, \pm Q)$.

1: $mask \leftarrow -swap$, $X_\Delta \leftarrow (X_P \oplus X_Q)\,\&\,mask$, $Z_\Delta \leftarrow (Z_P \oplus Z_Q)\,\&\,mask$
2: **return** $([X_P \oplus X_\Delta : Z_P \oplus Z_\Delta], [X_Q \oplus X_\Delta : Z_Q \oplus Z_\Delta])$

B. The Renes-Smith algorithm and the biquadratic test

Verification of Kummer pseudo-addition can be carried out with a method proposed by Renes and Smith [Renes-Smith, Proposition 3]). It is summarized in Algorithm 30.

Algorithm 31 summarizes the proposed biquadratic method that extends the Renes-Smith test as required for our Kummer-based blind signature and identification scheme. The costs refer to operations in $\mathbb{F}_p$ as follows: `M`: generic multiplication; `S`: squaring; `C`: multiplication by small constant (A or $2A$); `a`: addition; `s`: subtraction; `d`: left-shift (by 1 or 2 positions, i.e.a doubling or a quadrupling). We remark that, although the authors of [Renes and Smith 2017] indicate a complexity of $8\mathtt{M} + 3\mathtt{S} + 1\mathtt{C} + 8\mathtt{a} + 4\mathtt{s}$ for their description this algorithm, it can be implemented at the slightly smaller cost $8\mathtt{M} + 2\mathtt{S} + 1\mathtt{C} + 4\mathtt{a} + 4\mathtt{s}$ even when none of the points is normalized, as indicated below. In the context of BZ[qDL], points $\pm P$ and $\pm Q$ are always normalized ($Z_P = Z_Q = 1$), so the actual cost is only $5\mathtt{M} + 2\mathtt{S} + 1\mathtt{C} + 4\mathtt{a} + 4\mathtt{s}$.

Algorithm 30 The Renes-Smith test xRS$(\pm P, \pm Q, \overline{\pm R})$

INPUT: $\pm P = [X_P : Z_P]$, $\pm Q = [X_Q : Z_Q] \in \mathfrak{G}$, $\overline{\pm R} \in \mathbb{F}_q$ such that $\pm R \in \mathfrak{G}$.
OUTPUT: Whether $\pm R \in \{\pm(P+Q), \pm(P-Q)\}$.
COST: $5\mathtt{M} + 2\mathtt{S} + 1\mathtt{C} + 4\mathtt{a} + 4\mathtt{s}$ (up to $3\mathtt{M}$ more if $Z_P \neq 1$ and/or $Z_Q \neq 1$).

1: $V_0 \leftarrow X_P \cdot X_Q$, $V_1 \leftarrow (Z_Q \neq 1)$ **?** $X_P \cdot Z_Q$ **:** X_P, $V_2 \leftarrow (Z_P \neq 1)$ **?** $Z_P \cdot X_Q$ **:** X_Q
2: $V_3 \leftarrow (Z_P = 1)$ **?** Z_Q **:** $(Z_Q = 1)$ **?** Z_P **:** $Z_P \cdot Z_Q$
3: $V_4 \leftarrow V_0 - V_3$, $V_4 \leftarrow V_4^2$, $V_5 \leftarrow V_1 - V_2$, $V_5 \leftarrow V_5^2$, $V_5 \leftarrow V_5 \cdot \overline{\pm R}$
4: $V_1 \leftarrow V_1 + V_2$, $V_2 \leftarrow V_0 \cdot V_3$, $V_2 \leftarrow 2A \cdot V_2$, $V_0 \leftarrow V_0 + V_3$, $V_0 \leftarrow V_0 \cdot V_1$, $V_0 \leftarrow V_0 + V_2$
5: $V_5 \leftarrow V_5 - V_0$, $V_5 \leftarrow V_5 - V_0$, $V_5 \leftarrow V_5 \cdot \overline{\pm R}$, $V_0 \leftarrow V_5 + V_4$
6: **return** $(V_0 = 0)$? true : false

Algorithm 31 The Biquadratic test xBQ$(\pm P, \pm Q, \pm R, \pm T)$

INPUT: $\pm P = [X_P : Z_P]$, $\pm Q = [X_Q : Z_Q]$, $\pm R = [X_R : Z_R]$, $\pm T = [X_T : Z_T] \in \mathfrak{G}$.
OUTPUT: Whether $\pm T \in \{\pm P \pm Q \pm R\}$.
COST: $20\mathtt{M} + 9\mathtt{S} + 3\mathtt{C} + 14\mathtt{a} + 6\mathtt{s} + 6\mathtt{d}$ ($2\mathtt{M}$ more if $Z_T \neq 0$).

▷ Renes-Smith factors:
1: $V_0 \leftarrow X_P \cdot X_Q$, $V_1 \leftarrow X_P \cdot Z_Q$, $V_2 \leftarrow Z_P \cdot X_Q$, $V_3 \leftarrow Z_P \cdot Z_Q$
2: $a \leftarrow V_1 - V_2$, $a \leftarrow a^2$, $c \leftarrow V_0 - V_3$, $c \leftarrow c^2$
3: $b \leftarrow V_0 \cdot V_3$, $b \leftarrow 2A \cdot b$, $V_0 \leftarrow V_0 + V_3$, $V_1 \leftarrow V_1 + V_2$, $V_2 \leftarrow V_0 \cdot V_1$, $b \leftarrow b + V_2$
▷ Biquadratic factors:
4: $V_0 \leftarrow X_R \cdot X_T$, $V_1 \leftarrow (Z_T \neq 1)$ **?** $X_R \cdot Z_T$ **:** X_R, $V_2 \leftarrow Z_R \cdot X_T$, $V_3 \leftarrow (Z_T \neq 1)$ **?** $Z_R \cdot Z_T$ **:** Z_R
5: $V_4 \leftarrow V_1 + V_2$, $V_1 \leftarrow V_1 - V_2$, $V_5 \leftarrow V_0 + V_3$, $V_2 \leftarrow V_0 - V_3$, $V_0 \leftarrow V_0 \cdot V_3$, $V_1 \leftarrow V_1^2$ $V_2 \leftarrow V_2^2$,
6: $V_6 \leftarrow A \cdot V_0$, $V_7 \leftarrow V_5 \cdot V_4$, $V_8 \leftarrow V_6 + V_7$, $V_8 \leftarrow 2A \cdot V_8$, $V_8 \leftarrow V_8 + V_2$, $V_8 \leftarrow V_0 \cdot V_8$, $V_8 \leftarrow V_8 \ll 2$
7: $V_3 \leftarrow V_3 \ll 1$, $V_0 \leftarrow V_5 + V_3$, $V_0 \leftarrow V_0^2$, $V_3 \leftarrow V_3^2$, $V_3 \leftarrow V_3 \ll 1$, $V_0 \leftarrow V_0 - V_3$
8: $V_4 \leftarrow V_4^2$, $V_0 \leftarrow V_0 \cdot V_4$, $V_8 \leftarrow V_8 + V_0$, $V_8 \leftarrow V_8 \ll 1$, $V_0 \leftarrow a \cdot V_2$
9: $V_3 \leftarrow c \cdot V_1$, $V_6 \leftarrow V_6 \ll 1$, $V_6 \leftarrow V_6 + V_7$, $V_7 \leftarrow V_0 + V_3$, $V_0 \leftarrow V_0^2$
10: $V_1 \leftarrow V_1 \cdot V_2$, $V_1 \leftarrow V_1 \cdot b$, $V_2 \leftarrow V_6 \cdot V_7$, $V_1 \leftarrow V_1 - V_2$, $V_1 \leftarrow V_1 \cdot b$, $V_1 \leftarrow V_1 \ll 2$
11: $V_2 \leftarrow a \cdot c$, $V_2 \leftarrow V_2 \cdot V_8$, $V_3 \leftarrow V_3^2$, $V_0 \leftarrow V_0 + V_1$, $V_0 \leftarrow V_0 + V_2$, $V_0 \leftarrow V_0 + V_3$
12: **return** $(V_0 = 0)$? true : false