Improved Signcryption and Broadcast Signcryption with Detachable Signatures

Edgar Elliott

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

University of Washington

2020

Committee:

Paulo Barreto

Anderson Nascimento

Program Authorized to Offer Degree:

Computer Science and Systems

University of Washington

Abstract

Improved Signcryption and Broadcast Signcryption with Detachable Signatures

Edgar Elliott

Chair of the Supervisory Committee:

Paulo Barreto

School of Engineering and Technology

With the ever-increasing amount of sensitive information being sent and received over the internet, the importance of fast secure cryptographic schemes will only continue to grow. In this paper we examine an existing but not widely known signcryption scheme, and propose modifications to it which increase its efficiency and protect it against side-channel cryptographic attacks.

1 Introduction

Signcryption schemes are a class of public-key cryptographic scheme that both encrypt the content of a message, so that only the intended recipient can access it, and assure the recipient that they can trust the source of the message. While this can be accomplished by first attaching a digital signature to the message and then encrypting the result, a scheme that performs both operations can do so much more efficiently.

Some signcryption schemes suffer from the flaw that if the recipient wants to be able to prove the identity of the sender, at some later time, the only way to do so is to save the encrypted text and perform the entire decryption step again, costing additional storage and time. To avoid this cost, it is desirable for a signature scheme to provide *detachable* signatures, where only the signature and the plaintext of the message are required for future verification. Usually it is also desirable for these signatures to be *unlinkable*, meaning that it cannot be shown that the signer and encryptor are the same user.

Some of the more efficient signcryption schemes are based on the hardness of the discrete logarithm problem, possibly coupled with related but similarly conventional security assumptions. One noteworthy signcryption scheme of this kind is the SEG-signcryption scheme (Schnorr ElGamal), which is purely based on the elliptic curve Diffie-Hellman assumption.

The SEG-signcryption protocol is arguably far less known than others of

its kind, having been published only in Libert's doctoral thesis[2] in 2006. However it is simple and efficient, and has the potential to be modified using concepts discovered since its original publication to create schemes with additional useful properties.

Furthermore, since all its components can be defined in terms of Kummer varieties, in particular those of Montgomery curves. It is possible to perform the entire scheme in this setting, further increasing its efficiency.

The remainder of this document is organized as follows. Section 2 gives an overview of the necessary background information required to understand the problem. Section 3 introduces and details the various component schemes that are combined into the final protocol. In Section 4 we describe the primary contribution of this paper, the Kummer-friendly variant of the SEG scheme. Section 5 explores further schemes which can be derived from the SEG scheme and examines their potential in the modified scheme. Section 6 reports on the experimental evaluation of the resulting protocol. We conclude in Section 7.

2 Background

Before examining the cryptographic schemes discussed in this paper it is important to lay the groundwork. This section will briefly cover some of the concepts that form the basis for the workings of the schemes discussed, as well as the motivation for some of the improvements being made.

2.1 The Diffie-Hellman Problem

All cryptographic schemes have, at their core, a mathematical problem for which no fast algorithm is known. For the schemes described here, that problem is the Diffie-Hellman Problem, which asks: Given some generator g of a group, and the values g^x and g^y, find the value of g^{xy}. If a user knows at least one of x or y then they can easily verify a solution, but any user, without knowledge of either, would find it infeasible to compute the answer, provided the numbers involved were sufficiently large.

2.2 Side-channel Attacks

Side channel attacks are a form of cryptographic attack that glean information by targeting the hardware running the cryptographic algorithms, rather than the algorithms themselves. By accessing information about the hardware such as power consumption, CPU usage, or even simply the time spent running the algorithm for different inputs, an attacker can gather potentially compromising information. The key to preventing side-channel attacks is to choose functions whose behavior does not change depending on their inputs. For example, the extended Euclidean algorithm is a common algorithm for finding multiplicative inverses in a finite field. However, the extended euclidean algorithms runtime is dependent on the values of its inputs, making it a potential leak. Instead of using the extended Euclidean algorithm to find an inverse, raising an element of a cyclic group to the power of two less than

the order of the group will return the multiplicative inverse in constant time, even if that constant time is slower than the extended Euclidean algorithm. That loss in performance can be mitigated by storing values as fractions during calculations so that division is only performed when absolutely necessary.

Another process that has the potential to be accessed in this way is scalar multiplication in a group, a central operation in the Diffie-Hellman problem. The most efficient way to compute scalar multiplication is to iterate over the binary representation of the scalar, doubling the point at each step and adding its original value again if the scalar has a set bit in that position. However, in any setting where doubling and adding are distinct operations with different computational costs, an attacker with access to power consumption or similar information can potentially determine which operation is being performed at any given moment. The sequence of operations performed during the double-and-add method can then be used to determine the binary value of the scalar involved in the calculation. There are multiple different approaches to preventing this, the relevant methods of which we will detail here.

2.3 Edwards Curves

Since the SEG scheme as described [2] imposes no restrictions on what group must be used, it is possible to choose a group where the operations of addition and point doubling are the same. Edwards curves, not widely known at the time of publication, provide such an environment. Edwards curve groups

are made up of points in some field K that satisfy an equation of the form $x^2 + y^2 = 1 + dx^2y^2$ and the operation has no simple geometric explanation, but is given by the formula:

$$(x_1, y_1) + (x_2 y_2) = \left(\frac{x_1 y_1 + x_2 y_2}{x_1 x_2 + y_1 y_2}, \frac{x_1 y_1 - x_2 y_2}{x_1 y_2 - y_1 x_2} \right)$$

This same formula can be used for addition even when adding a point to itself. Additionally only the least significant bit of the y coordinate of a point needs to be stored or broadcast along with the x coordinate since the point can be reconstructed from that information alone, reducing the length of the signcryptogram significantly.

2.4 Montgomery Curves and the Montgomery Ladder

Montgomery curves are another form of elliptic curve. In fact, Edwards curves and Montgomery curves have an equivalence where a curve in one form can be mapped to an equivalent curve in the other form preserving the group structure under the other curve's group law. Montgomery curves consist of points in a field K that satisfy some equation of the form $By^2 = x^3 + Ax^2 + x$. Unlike Edwards curves, the Montgomery curve group law requires a different formula for addition when the addends are the same point. To understand why this is the case requires examining the group law. In Montgomery curve groups, adding two points requires first finding the line that passes through both of those points. In the case where the two

points are the same it instead requires finding the tangent line to the curve at that point. Unfortunately, these two cases require fundamentally different formulas, which creates a potential vulnerability to side-channel attacks due to their differing computational costs.

The solution to this vulnerability is a formula called the Montgomery ladder. Like the normal double-and-add method, the Montgomery ladder iterates over the scalar one bit at a time. However, unlike the previous method it performs exactly the same operations at each step. It initializes two values $R_0 = 0$ and $R_1 = P$, and at each step it writes $R_0 + R_1$ to one value, and doubles the other. Which of R_0 or R_1 receives which resulting value depends on the bit of the scalar at that step. While this algorithm's process may be less intuitive, its constant operations make it resistant to side-channel attacks [1] while keeping the double-and-add method's logarithmic time complexity.

2.5 The Kummer Variety

The Kummer variety $\mathfrak{G}$ is not actually a group, it is the result of identifying every point in a group $\langle G \rangle$ with its inverse such that every pair of points $\{P, -P\} \subset \langle G \rangle$ uniquely correspond to a single element $\pm P \in \mathfrak{G}$. The resulting structure maintains some but not all of the properties of a group. In particular it does not have access to the group's addition function, however, instead it has access to a pseudo-addition function which takes in three inputs $\pm Q$, $\pm P$, and $\pm(Q - P)$ in order to compute the value $\pm(Q + P)$. This is

sufficient for the Montgomery ladder since thanks to the fact that at every step $R_0 - R_1 = P$. The advantage of the Kummer variety is its speed, since just the x coordinate is sufficient to perform all operations without any need to reconstruct the entire point. For an algorithm designed to use it, like qDSA (quotient Digital Signature Algorithm) [3], the Kummer variety can deliver a significant increase in efficiency.

3 Component Schemes

The final scheme outlined in this paper is most easily understood through looking at its building blocks. We will recap the details of those schemes here.

3.1 The Schnorr Signature

For the sake of completeness, we start by summarizing Schnorr's traditional signature scheme.

- **Setup:** given a security parameter k, pick a $2k$-bit prime n, a cyclic group $\mathbb{G}$ of order n, a generator $G \in \mathbb{G}$ and a hash function $\mathcal{H} : \mathbb{G} \times \mathbb{G} \times \{0,1\}^* \to \mathbb{Z}/n$. Return the parameter set $I \leftarrow (n, \mathbb{G}, G, \mathcal{H})$.
- **Keygen:** The signer picks a private key $s \xleftarrow{\$} \mathbb{Z}/n$, then computes and publishes the corresponding public key $V \leftarrow [s]G \in \mathbb{G}$.

- **Sign:** To sign a message m, the signer picks $r \xleftarrow{\$} \mathbb{Z}/n$ and computes $R \leftarrow [r]G$, $h \leftarrow \mathcal{H}(R, V, \mathsf{m})$, $z \leftarrow r + sh \pmod{n}$. The signature is the pair $\sigma \leftarrow (R, z) \in \langle G \rangle \times \mathbb{Z}/n$.

- **Verify:** To verify a signature (R, z) under the public key V, the verifier computes $h \leftarrow \mathcal{H}(R, V, \mathsf{m})$ and checks that $[z]G - [h]V = R$.

The verification step $[z]G - [h]V = R$ is equivalent to $[r + sh]G - [sh]G = [r]G$. With the assumption that the discrete logarithm problem is hard, only someone with knowledge of s could have sent this message. Similarly, h being generated from all three of R, V, and the message text means that if any of those three values change then h will change and the signature will not verify correctly.

The length of the signature ends up being one $2k$ bit integer plus whatever information is needed to uniquely identify the element of the group R. In the case of Edwards curves, the x-coordinate plus the least-significant bit of the y-coordinate is sufficient for the recipient to reconstruct the point.

3.2 The SEG Signcryption Scheme

The original SEG-signcryption scheme was devised by Libert [2]. It is based on Schnorr signatures (and indirectly on ElGamal encryption).

In what follows, Alice denotes the signcryption sender and Bob denotes the signcryption recipient.

- **Setup:** given a security parameter k and a plaintext length ℓ, pick a $2k$-bit prime n, a cyclic group $\mathbb{G}$ of order n, a generator $G \in \mathbb{G}$, and hash functions $\mathcal{H} : \mathbb{G} \times \{0,1\}^\ell \to \mathbb{Z}/n$, $\mathcal{F} : \mathbb{Z}/n \to \{0,1\}^{\ell+2k}$ and $\mathcal{G} : \mathbb{G}^3 \to \{0,1\}^{2k}$. Return the parameter set $I \leftarrow (n, \mathbb{G}, G, \mathcal{H}, \mathcal{F}, \mathcal{G}, \ell)$.

- **Keygen:** user U picks a private key $s_U \xleftarrow{\$} \mathbb{Z}/n$, then computes and publishes the corresponding public key $V_U \leftarrow [s_U]G \in \mathbb{G}$.

- **Sign/Encrypt:** given a message $\mathsf{m} \in \{0,1\}^\ell$, Bob's public key V_B and Alice's private key s_A, Alice does the following:

  1. picks $r \xleftarrow{\$} \mathbb{Z}/n$ and computes $R \leftarrow [r]G \in \mathbb{G}$, $h \leftarrow \mathcal{H}(R, V_A, \mathsf{m}) \in \mathbb{Z}/n$ and $z \leftarrow r + s_A h \pmod{n}$ (notice that this is essentially a Schnorr signature);
  2. computes $\Omega \leftarrow [r]V_B \in \mathbb{G}$, $\zeta \leftarrow z \oplus \mathcal{G}(R, V_B, \Omega) \in \{0,1\}^{2k}$, and $\mu \leftarrow (\mathsf{m} \,\|\, V_A) \oplus \mathcal{F}(z) \in \{0,1\}^{\ell+2k}$ (notice that the computation of Ω essentially completes an ElGamal encryption step).

  Then Alice sends the signcryptogram $C \leftarrow (R, \zeta, \mu) \in \mathbb{G} \times \{0,1\}^{2k+\ell+2k}$ to Bob.

- **Decrypt/Verify:** given $C = (R, \zeta, \mu)$, Bob does the following:

  1. computes $\Omega \leftarrow [s_B]R \in \mathbb{G}$ and $z \leftarrow \zeta \oplus \mathcal{G}(R, V_B, \Omega)$, and rejects C if $z \notin \mathbb{Z}/n$;
  2. computes $(\mathsf{m} \,\|\, V_A) \leftarrow \mu \oplus \mathcal{F}(z) \in \{0,1\}^{\ell+2k}$;

3. computes $h \leftarrow \mathcal{H}(R, V_A, \mathsf{m}) \in \mathbb{Z}/n$, and accepts m if $R = [z]G - [h]V_A$.

If m is accepted, Bob detaches Alice's signature $\sigma \leftarrow (R, z)$, or equivalently $\sigma \leftarrow (h, z)$, for the message m.

This scheme begins with the Schnorr scheme, but where that scheme ended, this one uses r and the recipient's public key to create the value Ω. This is where the Diffie-Hellman assumption comes into play, since the only values a potential attacker should have access to are R and Vb it should be infeasible for them to compute Ω. Once the recipient has computed Ω they can compute the necessary hash values to retrieve z, m, and V_a and complete the verification.

The detached signature in this scheme is not *strongly* unlinkable, since the same R is present in σ and in C, and hence they can be linked.

However, it is still *weakly* unlinkable, in the sense that Alice can deny having created C (of course, she cannot deny having signed m into (R, z) as soon as this signature is detached).

Indeed, if Alice merely signs m into (R, z) and makes the signature and her public key V_A available to Bob, he can transform them into a signcryptogram for himself by computing $\Omega \leftarrow [s_B]R \in \mathbb{G}$, $\zeta \leftarrow z \oplus \mathcal{G}(R, V_B, \Omega) \in \{0,1\}^{2k}$, $\mu \leftarrow (m \,\|\, V_A) \oplus \mathcal{F}(z) \in \{0,1\}^{\ell+2k}$, and finally $C \leftarrow (R, \zeta, \mu) \in \mathbb{G} \times \{0,1\}^{2k+\ell+2k}$.

3.3 The qDSA Signature

The qDSA signature scheme [3] operates in the so-called Kummer variety $\mathfrak{G}$.

For an underlying Montgomery curve over $\mathbb{F}_p$ given by the affine equation $E_M : By^2 = x^3 + Ax^2 + x$ with $A \notin \{0, \pm 2\}$ and $B \neq 0$, this means that, for any $P = (x, y) \in E_M$, $\pm P$ can be conveniently represented by the affine x-coordinate $\overline{\pm P} := x \in \mathbb{F}_p$ in the protocol, or else (during computations) by a pair $(X : Z) \in \mathbb{F}_p^2$ of projective coordinates such that $\overline{\pm P} = X/Z = XZ^{p-2}$ (with the reasonable convention of denoting the point at infinity by 0, since the affine point $(0, 0)$ on the Montgomery curve has order 2 and thus is not in a group of cryptographic relevance).

- The signer's key pair is $(s \in \mathbb{Z}/n, \overline{\pm V} \leftarrow \overline{\pm [s]G})$.
- To sign a message m, the signer picks a signing formal key pair $(r \xleftarrow{\$} \mathbb{Z}/n, \overline{\pm R} \leftarrow \overline{\pm [r]G})$ and computes $h \leftarrow \overline{\mathcal{H}}(\overline{\pm R}, \overline{\pm V}, \mathsf{m})$, $z \leftarrow r + sh \pmod{n}$. The signature is the pair $(\overline{\pm R}, z) \in \mathbb{F}_p \times \mathbb{Z}/n$.
- To verify a signature $(\overline{\pm R}, s)$, the verifier computes $h \leftarrow \overline{\mathcal{H}}(\overline{\pm R}, \overline{\pm V}, \mathsf{m})$ and checks that $\pm R \in \{\pm([z]G + [h]V), \pm([z]G - [h]V)\}$. This is carried out efficiently for Montgomery curves by checking $\mathsf{RS}(\pm [z]G, \pm [h]V, \overline{\pm R})$, using the Renes-Smith technique (Algorithm B.1).

Other than using the Kummer variety there isn't much difference between qDSA signatures and Schnorr signatures. The biggest change is that the verification step must take into account the ambiguity associated with addition

and subtraction, which the algorithm described by Renes and Smith [3] does efficiently.

4 The Kummer Friendly SEG Scheme

Now we outline the final scheme, as with qDSA and Schnorr signatures, the structure parallels the SEG scheme almost exactly, substituting group operations for the Kummer variety equivalents where necessary.

- **Setup:** given a security parameter k and a plaintext length ℓ, pick a $2k$-bit prime n, Kummer variety $\mathfrak{G} := \mathbb{G}/\pm$ where $\mathbb{G}$ is a cyclic group of order n, a generator $G \in \mathbb{G}$, and hash functions $\overline{\mathcal{H}} : \mathbb{F}_p \times \{0,1\}^\ell \to \mathbb{Z}/n$, $\mathcal{F} : \mathbb{Z}/n \to \{0,1\}^{\ell+2k}$ and $\overline{\mathcal{G}} : \mathbb{F}_p^3 \to \{0,1\}^{2k}$. Return the parameter set $I \leftarrow (n, \mathbb{G}, G, \overline{\mathcal{H}}, \mathcal{F}, \overline{\mathcal{G}}, \ell)$.

- **Keygen:** user U picks a private key $s_U \xleftarrow{\$} \mathbb{Z}/n$, then computes and publishes the corresponding public key $\overline{\pm V_U} \leftarrow \overline{\pm [s_U]G} \in \mathbb{F}_p$.

- **Sign/Encrypt:** given a message $\mathsf{m} \in \{0,1\}^\ell$, Bob's public key $\overline{\pm V_B}$ and Alice's private key s_A, Alice does the following:

  1. picks $r \xleftarrow{\$} \mathbb{Z}/n$ and computes $\overline{\pm R} \leftarrow \overline{\pm [r]G} \in \mathbb{F}_p$, $h \leftarrow \overline{\mathcal{H}}(\overline{\pm R}, \overline{\pm V_A}, \mathsf{m}) \in \mathbb{Z}/n$ and $z \leftarrow r + s_A h \pmod{n}$ (notice that this is essentially a qDSA signature);
  2. computes $\overline{\pm \Omega} \leftarrow \overline{\pm [r]V_B} \in \mathbb{F}_p$, $\zeta \leftarrow z \oplus \overline{\mathcal{G}}(\overline{\pm R}, \overline{\pm V_B}, \overline{\pm \Omega}) \in \{0,1\}^{2k}$, and $\mu \leftarrow (\mathsf{m} \,\|\, \overline{\pm V_A}) \oplus \mathcal{F}(z) \in \{0,1\}^{\ell+2k}$ (notice that

the computation of $\overline{\pm\Omega}$ essentially completes a Kummer-based ElGamal encryption step).

Then Alice sends the signcryptogram $C \leftarrow (\overline{\pm R}, \zeta, \mu) \in \mathbb{F}_p \times \{0,1\}^{2k+\ell+2k}$ to Bob.

- **Decrypt/Verify:** given $C = (\overline{\pm R}, \zeta, \mu)$, Bob does the following:

  1. computes $\overline{\pm\Omega} \leftarrow \overline{\pm[s_B]R} \in \mathbb{G}$ and $z \leftarrow \zeta \oplus \overline{\mathcal{G}}(\overline{\pm R}, \overline{\pm V_B}, \overline{\pm\Omega})$, and rejects C if $z \notin \mathbb{Z}/n$;
  2. computes $(\mathsf{m} \,\|\, \overline{\pm V_A}) \leftarrow \mu \oplus \mathcal{F}(z) \in \{0,1\}^{\ell+2k}$;
  3. computes $h \leftarrow \overline{\mathcal{H}}(\overline{\pm R}, \overline{\pm V_A}, \mathsf{m}) \in \mathbb{Z}/n$
  4. accepts m if $\mathsf{RS}(\pm[z]G, \pm[h]V, \overline{\pm R})$ is true.

  If m is accepted, Bob detaches Alice's qDSA signature $\sigma \leftarrow (\overline{\pm R}, z)$ for the message m.

As it happens for the original Schnorr-based scheme, the detached signature here is not strongly unlinkable (the same $\overline{\pm R}$ is present in σ and in C) but it is weakly unlinkable in the same sense as before: if Alice merely signs m into $(\overline{\pm R}, z)$ and makes the signature and her public key $\overline{\pm V_A}$ available to Bob, he can transform them into a signcryptogram for himself by computing $\overline{\pm\Omega} \leftarrow \overline{\pm[s_B]R} \in \mathbb{F}_p$, $\zeta \leftarrow z \oplus \overline{\mathcal{G}}(\overline{\pm R}, \overline{\pm V_B}, \overline{\pm\Omega}) \in \{0,1\}^{2k}$, $\mu \leftarrow (m \,\|\, \overline{\pm V_A}) \oplus \mathcal{F}(z) \in \{0,1\}^{\ell+2k}$, and finally $C \leftarrow (\overline{\pm R}, \zeta, \mu) \in \mathbb{F}_p \times \{0,1\}^{2k+\ell+2k}$.

5 Derived Schemes

5.1 Broadcast Signcryption

We describe this scheme in the Kummer version.

Alice can sign/encrypt a message to N recipients at once with smaller effort than repeating the full signcryption process N times. To achieve this, she $r \xleftarrow{\$} \mathbb{Z}/n$ and computes $\overline{\pm R} \leftarrow \overline{\pm[r]G}$, signs m into $(\overline{\pm R}, z)$, and encrypts m (and $\overline{\pm V_A}$) into μ as before. Then for each recipient (whose public key is $\overline{\pm V_i}$) she computes $\zeta_i \leftarrow z \oplus \overline{\mathcal{G}}(\overline{\pm R}, \overline{\pm V_i}, \overline{\pm[r]V_i}) \in \{0,1\}^k$ and completes the broadcast signcryptogram $C \leftarrow (\overline{\pm R}, \zeta_1, \ldots, \zeta_N, \mu) \in \mathbb{G} \times \{0,1\}^{2kN} \times \{0,1\}^{\ell+2k}$.

5.1.1 Complexity analysis

The core computational cost to sign/encrypt is $1 + N$ multiplications by scalar. This contrasts with a cost of $2N$ when a separate signcryptogram is created for each recipient. Thus, for large N the processing cost is nearly halved.

The space taken by all signcryptograms is $2k+2kN+\ell+2k = 2k(N+2)+\ell$, which is much smaller than the collective space of $(4k + \ell)N$ bits needed by separate signcryptograms. In particular, a single copy of the signcrypted message itself is needed. Although this does not decrease transmission bandwidth since each recipient still needs to get a triple $(\overline{\pm R}, \zeta_i, \mu)$, it does reduce storage requirements by a factor roughly N when the message is much longer

than the overhead of an individual signcryptogram, that is, when $\ell \gg 4k$. This is particularly relevant when the signcryptograms must be kept in a server for distribution.

The cost for each recipient to decrypt/verify is unchanged, namely, 3 multiplications by scalar.

5.1.2 Security

We argue that the broadcast signcryption scheme remains secure from Alice's point of view despite the common signcryption nonce $\overline{\pm R}$.

The reason is the very fact that, given only Alice's public key $\overline{\pm V_A}$, the (public) message m, and a single valid signature $(\overline{\pm R}, z)$ for it, any recipient could prepare an individual signcryptogram $C_i \leftarrow (\overline{\pm R}, \zeta_i, \mu)$ on their own (with a ζ_i of their choosing, computed exactly the same way as in the weak unlinkability discussion for detached signatures), each sharing the same R and the same μ. After all of these signcryptograms are published, they can be compressed into the form $C \leftarrow (\overline{\pm R}, \zeta_1, \ldots, \zeta_N, \mu)$.

This same observation holds for the original scheme over Abelian varieties and for the proposed variant over Kummer varieties alike. The broadcast protocol does not involve any secret information besides that recipient's private key. In particular, it only involves Alice's *public* information: hence, if the result were insecure, the underlying Schnorr signature scheme would be itself insecure.

5.2 Joint-Broadcast Signcryption

In broadcast signcryption, either recipient is able to retrieve the message (and its detached signature) independently. But one could also define joint broadcast signcryption where decryption/verification is only enabled when all recipients collaborate without setting up a secret sharing scheme in advance.

It turns out that the following protocol, which addresses the above scenario, easily solves the problem with the plain SEG-signcryption scheme, but is hard to adapt to a Kummer variant, as we will explain afterward.

Alice computes $V \leftarrow \sum_{i=1}^{N} V_i$, $\Omega \leftarrow [r]V$, $\zeta \leftarrow z \oplus \mathcal{G}(R, V, \Omega)$, and completes the signcryptogram $C \leftarrow (R, \zeta, \mu)$.

To decrypt/verify, each recipient computes $\Omega_i \leftarrow [s_i]R \in \mathbb{G}$, then they all jointly compute $V \leftarrow \sum_{i=1}^{N} V_i$, $\Omega \leftarrow \sum_{i=1}^{N} \Omega_i$ and $z \leftarrow \zeta \oplus \mathcal{G}(R, V, \Omega)$, thereby being finally able to recover and verify m and detach its signature.

The obstacle to obtain a Kummer variant of this protocol lies in the need to compute *sums* of group elements. Although it is straightforward to compute $V \leftarrow \sum_{i=1}^{N} V_i$ and $\Omega \leftarrow \sum_{i=1}^{N} \Omega_i$, computing the corresponding quantities $\overline{\pm V} \leftarrow \sum_{i=1}^{N} \overline{\pm V_i}$ and $\overline{\pm \Omega} \leftarrow \sum_{i=1}^{N} \overline{\pm \Omega_i}$ is not only algorithmically non-trivial, but it also leads to complications in its very interpretation: due to the sign ambiguity in a Kummer variety, the obtained $\overline{\pm V}$ and $\overline{\pm \Omega}$ may correspond to any group elements of form $\pm V_1 \pm \cdots \pm V_N$ and $\pm \Omega_1 \pm \cdots \pm \Omega_N$ respectively, meaning an exponential increase in ambiguity, i.e. there would be 2^{2N-2} different possibilities for ζ, and decryption would become infeasible.

6 Experimental Evaluation

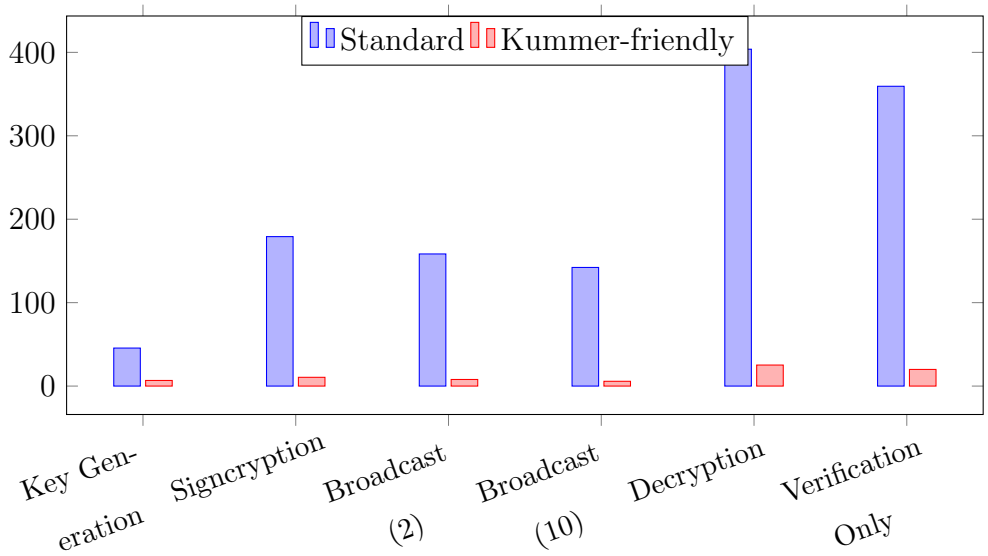
Both the unmodified SEG scheme and the modified version were implemented in Java using the elliptic curve group E_521 and $KMACXOF256$ hash function.

Testing consisted of one hundred trials each of Public key generation, message signcryption, message decryption and verification, signature re-verification, and broadcast signcryption to groups of 2, and 10 recipients. Each key was used on the same 512 byte random plaintext. Each step was timed and the average times in milliseconds are present in table 1. The times for the broadcast versions are divided by the number of recipients to provide meaningful comparison with the single-recipient version.

The modified version of the scheme using the Kummer variety is significantly faster in all regards. The speed of key generation, the least crucial step to optimize, is performed in less than a sixth of the time. While the encryption and decryption experience far greater increases in speed. Broadcast signcryption to multiple recipients experienced the greatest increase, with the modified version of the scheme being more than twenty times the speed of its standard counterpart.

	Standard	Kummer-friendly
Key Generation	45.56	6.73
Signcryption	179.15	10.51
Broadcast (2 recipients)	158.37	7.91
Broadcast (10 recipients)	142.19	5.74
Decryption	403.79	25.21
Re-verification	359.35	19.95

Table 1: Average Times (ms)



Changing the length of the message plaintext had a negligable effect on the performance of the algorithm. The hash function F must be set to output a hash long enough to cover both the message and the sender's public key, however adjusting the output length of F also had little effect on the efficiency, with the group operations even in the modified case being a much larger portion of the runtime.

7 Conclusion

In conclusion, the SEG signcryption scheme put forth by Libert was already an efficient scheme which had the benefit of detachable signatures and the flexibility to work with different types of elliptic curve groups, including Edwards curves which have inherent resistance to side-channel attacks. It can also be further modified for greater efficiency in certain contexts such as broadcast signcryption. Furthermore, the qDSA provides a signature scheme that can be implemented without access to the full group structure, allowing the entire scheme to be implemented with only Kummer arithmetic. This allows the modified SEG signcryption scheme put forth in this paper a significant advantage in speed over its unmodified counterpart, while still maintaining resistance to side-channel attacks.

References

[1] Éric Brier and Marc Joye. Weierstraß elliptic curves and side-channel attacks. In David Naccache and Pascal Paillier, editors, *Public Key Cryptography*, pages 335–345, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[2] Benôt Libert. New secure applications of bilinear maps in cryptography. 2006.

[3] Joost Renes and Benjamin Smith. qDSA: Small and secure digital signatures with curve-based Diffie-Hellman key pairs. Cryptology ePrint Archive, Report 2017/518, 2017. https://eprint.iacr.org/2017/518.

A The Montgomery ladder

Laddering multiplication by scalar (Algorithms A.1 and A.2) uses two functions xADD (Algorithm A.3) for differential addition and xDBL (Algorithm A.4) for doubling on a Montgomery curve, and also isochronous conditional swapping xSWP (Algorithm A.5) to ensure uniformity of access.

Algorithm A.1 The (extended) Montgomery ladder, $\mathsf{ML}^+(k, \pm P)$

INPUT: $k = \sum_{i=0}^{\ell} k_i 2^i$ with $k_\ell = 1$, and $\pm P = (X_P : Z_P) \in \mathfrak{G}$.
OUTPUT: $\pm[k]P = (X_k : Z_k) \in \mathfrak{G}$ if $\pm P \neq O$, otherwise $Z_k = 0$; and also $\pm[k+1]P = (X_{k+1} : Z_{k+1}) \in \mathfrak{G}$ if $\pm P \neq O$, otherwise $Z_{k+1} = 0$.
COST: ℓ calls to xADD, $\ell + 1$ calls to xDBL, $\ell + 1$ calls to xSWP.

$(\pm R_0, \pm R_1) \leftarrow (\pm P, \mathsf{xDBL}(\pm P))$
$swap \leftarrow 0$
for $i \leftarrow \ell - 1$ **downto** 0 **do**
    $(\pm R_0, \pm R_1) \leftarrow \mathsf{xSWP}(\pm R_0, \pm R_1, k_i \oplus swap)$
    $(\pm R_0, \pm R_1) \leftarrow (\mathsf{xDBL}(\pm R_0), \mathsf{xADD}(\pm R_0, \pm R_1, \pm P))$
    $swap \leftarrow k_i$
end for
$(\pm R_0, \pm R_1) \leftarrow \mathsf{xSWP}(\pm R_0, \pm R_1, swap)$
return $(\pm R_0, \pm R_1)$

Algorithm A.2 The Montgomery ladder, $\mathsf{ML}(k, \pm P)$

INPUT: $k = \sum_{i=0}^{\ell} k_i 2^i$ with $k_\ell = 1$, and $\pm P = (X_P : Z_P) \in \mathfrak{G}$.
OUTPUT: $\pm[k]P = (X_k : Z_k) \in \mathfrak{G}$ if $\pm P \neq O$, otherwise $Z_k = 0$.
COST: ℓ calls to xADD, $\ell + 1$ calls to xDBL, $\ell + 1$ calls to xSWP.

1: $(\pm R_0, \pm R_1) \leftarrow \mathsf{ML}^+(k, \pm P)$
2: **return** $\pm R_0$

Algorithm A.3 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 $P + Q \neq O$, 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: $V_0 \leftarrow X_P + Z_P$
2: $V_1 \leftarrow X_Q - Z_Q$
3: $V_1 \leftarrow V_1 \cdot V_0$
4: $V_0 \leftarrow X_P - Z_P$
5: $V_2 \leftarrow X_Q + Z_Q$
6: $V_2 \leftarrow V_2 \cdot V_0$
7: $V_0 \leftarrow V_1 + V_2$
8: $V_0 \leftarrow V_0^2$
9: $V_1 \leftarrow V_1 - V_2$
10: $V_1 \leftarrow V_1^2$
11: $X_+ \leftarrow Z_- \cdot V_0$ // no multiplication when $Z_- = 1$
12: $Z_+ \leftarrow X_- \cdot V_1$
13: **return** $(X_+ : Z_+)$

Algorithm A.4 Montgomery pseudo-doubling, $\mathsf{xDBL}(\pm P)$

INPUT: $\pm P = (X_P : Z_P) \in \mathbb{F}_q^2$.
OUTPUT: $\pm[2]P = (X_2 : Z_2) \in \mathfrak{G}$ if $P \neq O$, otherwise $Z_2 = 0$.
COST: $\mathtt{2M} + \mathtt{2S} + \mathtt{1c} + \mathtt{2a} + \mathtt{1s}$

1: $V_0 \leftarrow X_P + Z_P$
2: $V_0 \leftarrow V_0^2$
3: $V_1 \leftarrow X_P - Z_P$
4: $V_1 \leftarrow V_1^2$
5: $X_2 \leftarrow V_0 \cdot V_1$
6: $V_0 \leftarrow V_0 - V_1$
7: $V_2 \leftarrow ((A+2)/4) \cdot V_0$ // $(A+2)/4$ assumed to be a precomputed constant
8: $V_2 \leftarrow V_2 + V_1$
9: $Z_2 \leftarrow V_0 \cdot V_2$
10: **return** $(X_2 : Z_2)$

Algorithm A.5 Isochronous conditional swapping, $\mathsf{xSWP}(\pm P, \pm Q, swap)$

INPUT: $\pm P = (X_P : Z_P), \pm Q = (X_Q : Z_Q) \in \mathbb{Z}/2^m \times \mathbb{Z}/2^m$; $swap \in \mathbb{Z}/2^m$. NB: all quantities viewed as m-bit signed integers for some m (in two's complement).
OUTPUT: $(\pm Q, \pm P)$ if $swap = 1$, otherwise $(\pm P, \pm Q)$.

1: $mask \leftarrow -swap$ // $0 \mapsto 0^m$, $1 \mapsto 1^m$
2: $X_\Delta \leftarrow (X_P \oplus X_Q) \,\&\, mask$
3: $Z_\Delta \leftarrow (Z_P \oplus Z_Q) \,\&\, mask$
4: **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

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 B.1. The cost indications refer to operations in $\mathbb{F}_p$ as follows: `M`: generic multiplication; `S`: squaring; `C`: multiplication by small

constant ($2A$); $\mathtt{a}$: addition; $\mathtt{s}$: subtraction.

Remark 1 *The authors of [Renes-Smith] claim a complexity of* $8\mathtt{M} + 3\mathtt{S} + 1\mathtt{C} + 8\mathtt{a} + 4\mathtt{s}$ *for this algorithm, but it can be implemented at the slightly smaller cost* $8\mathtt{M} + 2\mathtt{S} + 1\mathtt{C} + 4\mathtt{a} + 4\mathtt{s}$ *as indicated below.*

Algorithm B.1 The Renes-Smith test $\mathsf{RS}(\pm P, \pm Q, \overline{\pm R})$ (detailed)

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: $8\mathtt{M} + 2\mathtt{S} + 1\mathtt{C} + 4\mathtt{a} + 4\mathtt{s}$.

1: $V_0 \leftarrow X_P \cdot X_Q$
2: $V_1 \leftarrow X_P \cdot Z_Q$
3: $V_2 \leftarrow Z_P \cdot X_Q$
4: $V_3 \leftarrow Z_P \cdot Z_Q$
5: $V_4 \leftarrow V_0 - V_3$
6: $V_4 \leftarrow V_4^2$
7: $V_5 \leftarrow V_1 - V_2$
8: $V_5 \leftarrow V_5^2$
9: $V_5 \leftarrow V_5 \cdot \overline{\pm R}$
10: $V_1 \leftarrow V_1 + V_2$
11: $V_2 \leftarrow V_0 \cdot V_3$
12: $V_2 \leftarrow 2A \cdot V_2$
13: $V_0 \leftarrow V_0 + V_3$
14: $V_0 \leftarrow V_0 \cdot V_1$
15: $V_0 \leftarrow V_0 + V_2$
16: $V_5 \leftarrow V_5 - V_0$
17: $V_5 \leftarrow V_5 - V_0$
18: $V_5 \leftarrow V_5 \cdot \overline{\pm R}$
19: $V_0 \leftarrow V_5 + V_4$
20: **return** $(V_0 = 0)$? true : false
