On the Efficiency and Security of Pairing-Based Protocols in the Type 1 and Type 4 Settings *

Sanjit Chatterjee, Darrel Hankerson, and Alfred Menezes

[1] Department of Combinatorics & Optimization, University of Waterloo
s2chatte@uwaterloo.ca
[2] Department of Mathematics and Statistics, Auburn University
hankedr@auburn.edu
[3] Department of Combinatorics & Optimization, University of Waterloo
ajmeneze@uwaterloo.ca

Abstract. We focus on the implementation and security aspects of cryptographic protocols that use Type 1 and Type 4 pairings. On the implementation front, we report improved timings for Type 1 pairings derived from supersingular elliptic curves in characteristic 2 and 3 and the first timings for supersingular genus-2 curves in characteristic 2 at the 128-bit security level. In the case of Type 4 pairings, our main contribution is a new method for hashing into $\mathbb{G}_2$ which makes the Type 4 setting almost as efficient as Type 3. On the security front, for some well-known protocols we discuss to what extent the security arguments are tenable when one moves to genus-2 curves in the Type 1 case. In Type 4, we observe that the Boneh-Shacham group signature scheme, the very first protocol for which the Type 4 setting was introduced in the literature, is trivially insecure, and we describe a small modification that appears to restore its security.

1 Introduction

Bilinear pairings have become an extremely useful instrument in the cryptographer's toolbox. Initial breakthroughs such as the one-round tripartite key agreement protocol of Joux [23] and a practical solution to the problem of identity-based encryption by Boneh and Franklin [5] have led to an almost exponential volume of research to find novel cryptographic applications of pairings.

At an abstract level, for three groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, a pairing is a function $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ that is bilinear and non-degenerate. For cryptographic applications we also need the pairing to be efficiently computable. In concrete settings such cryptographically suitable bilinear pairings can be realized over elliptic curves or, more generally, over hyperelliptic curves and abelian varieties. Naturally the groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ as well as the pairing function are constrained by the underlying mathematical structure over which they are defined.

* An abbreviated version of this paper appeared in the proceedings of the *WAIFI 2010* conference, Lecture Notes in Computer Science, 6087 (2010), 114-134.

However, as noted by Galbraith, Paterson and Smart [17], protocol designers sometimes treat the bilinear pairing as a "black box". As a result the designers may gloss over such important structural constraints and the subtleties they introduce in the protocols and their security arguments. This in turn may lead to erroneous or misleading claims about the efficiency and security of pairing-based protocols. For example, protocols employing bilinear pairings sometimes assume that the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ also possess some additional properties such as efficient hashing into $\mathbb{G}_2$ or the existence of an efficiently computable isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$. Different types of pairings can be realized that possess the properties required of a particular protocol, but not all protocols can be implemented using the same type of pairing.

This motivated a classification of bilinear pairings into different types based on the concrete structures of the underlying groups [17]. The focus of that work was on three types of pairings where the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are of the same prime order n. When $\mathbb{G}_1 = \mathbb{G}_2$, the pairing is said to be symmetric (called Type 1 in [17]). The pairing is asymmetric when $\mathbb{G}_1 \neq \mathbb{G}_2$. If there is an efficiently computable isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ then the pairing is said to be of Type 2; if no such isomorphism is known it is called a Type 3 pairing. In either case no efficiently computable isomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$ is known.

Symmetric pairings (Type 1) are derived from supersingular (hyper)elliptic curves whereas asymmetric pairings are derived from ordinary curves. Known examples of such pairings are the Weil and Tate pairings and their modifications such as the eta pairing [2], the ate pairing [22], and the R-ate pairing [25].

Cryptographic protocols employing pairings are usually described in the symmetric setting, allowing for a relatively simpler description of the protocol and its security argument. However, current research indicates that, at higher security levels, Type 1 pairings are expected to be slower on many platforms. So from the point of view of efficient implementation, Type 2 and Type 3 are considered better choices. And, for a protocol originally proposed in the symmetric setting, it is usually possible to translate the protocol description and the security argument to the asymmetric setting.[1]

In the asymmetric setting, current research suggests that Type 3 is overall a better choice [17]. This is because of the reduced cost of pairing evaluation and also the relatively smaller size of elements of $\mathbb{G}_2$ which in turn reduces the cost of other operations such as group operations in $\mathbb{G}_2$ or testing membership in $\mathbb{G}_2$. The major functional distinctions between the Type 3 and Type 2 settings are that, first of all, in the former it is possible to hash into $\mathbb{G}_2$, which is infeasible in the latter; and, secondly, whereas there is an efficiently computable isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ in Type 2, no such efficiently computable map is known for Type 3. Because in some cases the description of a protocol or its security argument employed the map ψ, it was earlier thought that either such protocols cannot be implemented in Type 3 [17] or a stronger complexity assumption was needed [6, 33]. Contrary to this belief it has been recently argued that any protocol or

[1] We are not aware of any protocol that has to be necessarily restricted to the symmetric setting.

security argument in Type 2 has a natural, efficient, and secure counterpart in Type 3 [11]. Hence, in the asymmetric setting there appears to be no good reason to use Type 2 instead of Type 3.

However, not all pairing-based protocols available in the literature can be implemented in Type 3 (or Type 2). For example, consider the case of the group signature scheme of Boneh and Shacham [7] with verifier-local revocation. In this protocol a random element of $\mathbb{G}_2$ is first obtained through hashing into $\mathbb{G}_2$ and then one applies the map ψ on this element to obtain the corresponding element of $\mathbb{G}_1$. As observed in [33], the protocol cannot be implemented in Type 2 because in that setting we do not have any algorithm to securely hash into $\mathbb{G}_2$, and furthermore cannot be implemented in Type 3 because in that case we do not know how to compute ψ for a random element of $\mathbb{G}_2$.

Perhaps realizing this shortcoming of Type 2 (and Type 3), Shacham in his PhD thesis [32] introduced a new kind of pairing. In this setting, while $\mathbb{G}_1$ and $\mathbb{G}_T$ are cyclic groups of prime order n, $\mathbb{G}_2$ is taken to be a group of exponent n, whose order is some power of n. This was later termed a Type 4 pairing [12, 17]. Like Type 2 and Type 3, a Type 4 pairing can be realized over ordinary elliptic or hyperelliptic curves. But unlike Type 2 or Type 3, here one can both hash into $\mathbb{G}_2$ and also have an efficiently computable homomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$. However, the hashing into $\mathbb{G}_2$ is reported to be quite expensive and there is a small probability that the pairing can be degenerate. The Boneh-Shacham group signature scheme of [7] is described in the Type 4 setting in [32] with a standard reductionist security argument. Several other protocols that use a Type 4 pairing have been proposed [29, 8] based on the Boneh-Shacham scheme. Thus, protocols that require hashing into $\mathbb{G}_2$ followed by an application of ψ can be implemented in the Type 4 setting although the protocol description may require some special care, and as noted in [12] the security argument can become cumbersome.

Protocols such as the Boneh-Shacham group signature scheme [7] can also be easily implemented in Type 1 because here $\mathbb{G}_1 = \mathbb{G}_2$ and hashing into $\mathbb{G}_1$ is very efficient. Also recall that most pairing-based protocols were originally proposed in this setting.[2] The main drawback of Type 1 is that the bitlengths of the elements of $\mathbb{G}_1$ will be larger (because of the smaller embedding degrees than what is achievable with asymmetric pairings) and, as a result, pairing computation and operations in $\mathbb{G}_1$ can be expected to be slower at high security levels. However, instructions on next-generation processors such as the forthcoming Intel machines may make Type 1 in characteristic 2 (and 3) fields an attractive choice. Some authors [2, 31] have also proposed to use genus-2 curves in the symmetric setting and use degenerate divisors to speed the pairing computation.

Our contribution. For efficient and secure implementation of the majority of pairing-based protocols it suffices to work in the Type 3 setting. However, as the preceding discussion suggests, we also need to consider the issues of efficient and secure implementation of protocols in the Type 1 and Type 4 settings.

[2] We note that not all protocols can be implemented securely in Type 1, e.g., those requiring the extended Diffie-Hellman problem (XDH or SXDH) to be hard [4, 9, 13].

While the question of efficiency does not require any additional justification, we draw attention to the question of security of a cryptographic protocol in these settings for the following reasons. A protocol described in the Type 1 setting which is implemented over a genus-2 curve may require a rewriting of the security argument to check whether the original security assurance is indeed maintained in this setting. Similarly, protocols described in the Type 4 setting may require special scrutiny because of the structure of the group $\mathbb{G}_2$, in particular its effect on the way the pairing is actually employed in the protocol. In this work we report on both these aspects of efficiency and security in the Type 1 and Type 4 settings.

Type 1. Following recent work of Beuchat et al. [3] and Aranha et al. [1], we provide improved timings for software implementation of Type 1 pairings over elliptic curves in characteristic 2 and characteristic 3 fields. We also report the first pairing timings for supersingular genus-2 curves at the 128-bit security level. We next take a look at the security arguments of some well-known protocols when implemented with these genus-2 curves and with degenerate divisors. Our analysis shows that for the Boneh-Lynn-Shacham (BLS) signature scheme [6] one needs a new hardness assumption that is trivially equivalent to the security of the scheme. In other words, the reductionist argument does not provide any meaningful assurance about the actual security of the protocol in this setting. A similar analysis is carried out for the Boneh-Franklin IBE scheme [5] and we observe that here also one needs to modify the original security assumption.

Type 4. As already mentioned, the main motivation for working in Type 4 is that it is possible to hash into $\mathbb{G}_2$. However, in terms of efficiency that appears to be a major limitation of the Type 4 setting as hashing into $\mathbb{G}_2$ has been reported to be computationally quite expensive [12, 17]. Here we propose a new technique to hash into $\mathbb{G}_2$ which is surprisingly cheap. This method is built upon the shorter representation of elements of $\mathbb{G}_2$ proposed in [10] in the context of the Type 2 setting. We also report the performance benefits that can be obtained for pairing evaluation and other operations involving elements of $\mathbb{G}_2$ in the Type 4 setting. As we have already noted, Type 4 pairings should be carefully used in cryptographic protocols. We show that the Boneh-Shacham group signature scheme as described in Shacham's thesis [32] is trivially insecure. We describe a small modification that appears to restore security. The signature now contains an element of $\mathbb{G}_2$, however with our new representation of elements of $\mathbb{G}_2$ the corresponding increase in the signature size is not very significant.

Organization. The remainder of the paper is organized as follows. In §2 we report the pairing computation times in Type 1 when using degenerate divisors in genus-2 curves over characteristic 2 fields at the 128-bit security level and also discuss the security aspects of the BLS signature and Boneh-Franklin IBE schemes in this setting. In §3 we describe the implementation aspects of Type 4 pairings derived from ordinary elliptic curves having even embedding degree

and show how one can efficiently hash into $\mathbb{G}_2$. We then show that the Boneh-Shacham group signature scheme as described [32] is insecure and how a small modification appears to restore security without significant performance penalty.

Notation. In the remainder of this paper, the first component $\mathbb{G}_1$ of the domain of a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is the order-n subgroup of $E(\mathbb{F}_q)$ or $J_C(\mathbb{F}_q)$, where E is an elliptic curve defined over $\mathbb{F}_q$ and J_C is the divisor class group of a genus-2 hyperelliptic curve C defined over $\mathbb{F}_q$. If e is a Type 1 pairing, then $\mathbb{G}_2 = \mathbb{G}_1$. If e is a Type 2, Type 3 or Type 4 pairing, then $\mathbb{G}_2 = \mathbb{T}$, $\mathbb{T}_0$, $E[n]$, respectively, where $E[n]$ is the n-torsion group of E, $\mathbb{T}_0$ is the Trace-0 subgroup of $E[n]$, and $\mathbb{T}$ is any order-n subgroup of $E[n]$ different from $\mathbb{G}_1$ and $\mathbb{T}_0$; cf. §3 for further details.

2 Type 1 pairings on supersingular genus-2 curves

In this section, we give the context for performance comparisons at the 128-bit security level for pairings based on supersingular genus-2 curves defined over characteristic 2 finite fields against those built on elliptic curves. Security aspects of the BLS signature scheme and the Boneh-Franklin IBE scheme are discussed in the genus-2 setting.

2.1 Type 1 pairings

We briefly describe three specific symmetric pairings derived from supersingular elliptic and hyperelliptic curves defined over fields of small characteristic; see [2] for details. The elliptic curves E are defined over $\mathbb{F}_{2^{1223}}$ and $\mathbb{F}_{3^{509}}$, and have embedding degrees 4 and 6, respectively. The genus-2 curve C is defined over $\mathbb{F}_{2^{439}}$ and has embedding degree 12. The pairings are $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$, where $\mathbb{G}_1$ is the subgroup of prime-order n of $E(\mathbb{F}_{2^{1223}})$, $E(\mathbb{F}_{3^{509}})$ or $J_C(\mathbb{F}_{2^{439}})$, and $\mathbb{G}_T$ is the order-n subgroup of $\mathbb{F}^*_{2^{4\cdot 1223}}$, $\mathbb{F}^*_{3^{6\cdot 509}}$ or $\mathbb{F}^*_{2^{12\cdot 439}}$, respectively. These pairings attain the 128-bit security level because Pollard's rho method for computing discrete logarithms in $E(\mathbb{F}_{2^{1223}})$, $E(\mathbb{F}_{3^{509}})$ and $J_C(\mathbb{F}_{2^{439}})$ has running time at least 2^{128}, as do the index-calculus algorithms for computing discrete logarithms in the extension fields $\mathbb{F}_{2^{4\cdot 1223}}$, $\mathbb{F}_{3^{6\cdot 509}}$ and $\mathbb{F}_{2^{12\cdot 439}}$ [27].

For genus 2, we focus on the most favourable case where the pairing is on degenerate divisors, each of which is essentially a point on the curve. The pairing algorithms given in [2] for the cases under consideration are similar in the sense that there is a "Miller evaluation" loop, followed by an exponentiation in the extension field to select a canonical representative. The final exponentiation is relatively inexpensive, and so the pairing cost can be estimated by counting field multiplications in the main loop.

Elliptic curve over characteristic 2 field. Let $q = 2^{1223}$. We chose the representation $\mathbb{F}_{2^{1223}} = \mathbb{F}_2[z]/(z^{1223}+z^{255}+1)$. Squaring is inexpensive relative to multiplication, and square roots are likewise inexpensive in this representations

since $\sqrt{z} = z^{612} + z^{128}$ and $\sqrt{c} = \sum c_{2i} z^i + \sqrt{z} \sum c_{2i+1} z^i$ for $c = \sum c_i z^i \in \mathbb{F}_{2^{1223}}$. The extension field $\mathbb{F}_{q^4}$ is represented using tower extensions $\mathbb{F}_{q^2} = \mathbb{F}_q[u]/(u^2 + u + 1)$ and $\mathbb{F}_{q^4} = \mathbb{F}_{q^2}[v]/(v^2 + v + u)$.

The supersingular elliptic curve $E_1/\mathbb{F}_{2^{1223}} : y^2 + y = x^3 + x$ has embedding degree 4. We have $\#E_1(\mathbb{F}_{2^{1223}}) = 5n$ where $n = (2^{1223} + 2^{612} + 1)/5$ is a 1221-bit prime. The doubling formula is $(x, y) \mapsto (x^4 + 1, x^4 + y^4 + 1)$, and hence the cost of doubling a point is relatively small.

Barreto, Galbraith, Ó hÉigeartaigh and Scott [2] give an algorithm for computing the η_T pairing, with cost estimated as $612 \times 7 = 4284$ $\mathbb{F}_q$-multiplications. The estimate is based on the number of multiplications in the main loop, and ignores the relatively minor cost of the final exponentiation (1 inversion in $\mathbb{F}_{q^4}$, 3 multiplications in $\mathbb{F}_{q^4}$, and 612 squarings in $\mathbb{F}_q$).

Elliptic curve over characteristic 3 field. Let $q = 3^{509}$. We chose the representation $\mathbb{F}_{3^{509}} = \mathbb{F}_3[z]/(z^{509} - z^{318} - z^{191} + z^{127} + 1)$. Cubing is inexpensive relative to multiplication, and the choice of reduction polynomial enables cube roots to be computed significantly faster than an $\mathbb{F}_q$-multiplication since $z^{1/3} = z^{467} + z^{361} - z^{276} + z^{255} + z^{170} + z^{85}$ and $z^{2/3} = -z^{234} + z^{128} - z^{43}$. The extension field $\mathbb{F}_{q^6}$ is represented using tower extensions $\mathbb{F}_{q^3} = \mathbb{F}_q[u]/(u^3 - u - 1)$ and $\mathbb{F}_{q^6} = \mathbb{F}_{q^3}[v]/(v^2 + 1)$.

The supersingular elliptic curve $E_2/\mathbb{F}_{3^{509}} : y^2 = x^3 - x + 1$ has embedding degree 6. We have $\#E_2(\mathbb{F}_{3^{509}}) = 7n$ where $n = (3^{509} - 3^{255} + 1)/7$ is an 804-bit prime. The tripling formula is $(x, y) \mapsto (x^9 - 1, -y^9)$, and hence the cost of tripling a point is relatively small.

The algorithm of Barreto, Galbraith, Ó hÉigeartaigh and Scott [2] for computing the η_T pairing has a cost estimate of $255 \times 14 = 3570$ $\mathbb{F}_q$-multiplications. As in the characteristic 2 case, the relatively minor cost of the final exponentiation has been ignored.

Genus 2 curve over characteristic 2 field. Let $m = 439$ and $q = 2^m$. We chose the representation $\mathbb{F}_{2^{439}} = \mathbb{F}_2[z]/(z^{439} + z^{49} + 1)$. Squaring and square root are inexpensive relative to multiplication in this representation, with $\sqrt{z} = z^{220} + z^{25}$. The extension field $\mathbb{F}_{q^{12}}$ is represented using tower extensions $\mathbb{F}_{q^6} = \mathbb{F}_q[w]/(w^6 + w^5 + w^3 + w^2 + 1)$ and $\mathbb{F}_{q^{12}} = \mathbb{F}_{q^6}[s]/(s^2 + s + w^5 + w^3)$.

The curve $C/\mathbb{F}_{2^{439}} : y^2 + y = x^5 + x^3$ has embedding degree 12. The divisor class group J_C has $\#J_C(\mathbb{F}_q) = 2^{2m} + 2^{(3m+1)/2} + 2^m + 2^{(m+1)/2} + 1 = 13n$, where n is an 875-bit prime. The pairing is defined for divisors $D = (P_1) + (P_2) - 2(\infty)$ where P_i are points on the curve; however the computation is faster for degenerate divisors where the support consists of a single point [2, 26]. If $D = (P) - (\infty)$ is such a degenerate divisor (with $P \in C(\mathbb{F}_q)$), then it is not necessarily the case that jD is degenerate; however, $8D$ is degenerate [2]. Furthermore, this octupling is relatively inexpensive, and is given by $8D = (\phi\pi^6 P) - (\infty)$ where $\pi(x, y) = (x^2, y^2)$ and $\phi(x, y) = (x + 1, y + x^2 + 1)$. Exploiting this octupling, the algorithm in [2] for η_T on degenerate divisors has an approximate cost of $219 \cdot 69 = 15111$ $\mathbb{F}_q$-multiplications (see [30] for additional details).

Comparisons. Barreto et al. [2] give experimental data for pairing times at the "950-bit" and "1230-bit" security levels, where the level is in terms of the bitsize of the extension field $\mathbb{F}_{q^k}$. Times (in milliseconds) for the η_T pairing in the 1230-bit case on a 3 GHz Intel Pentium 4 are given in Table 1.

Table 1. Times (in milliseconds) from [2] for the η_T pairing at the "1230-bit security level" on a 3 GHz Intel Pentium 4.

		$C(\mathbb{F}_{2^{103}})$	
$E(\mathbb{F}_{2^{307}})$	$E(\mathbb{F}_{3^{127}})$	degenerate	general
3.50	5.36	1.87	6.42

Their work shows significant incentive to use genus-2 curves in the case that the pairing is on degenerate divisors. However, field multiplication for $\mathbb{F}_{2^{103}}$ exploited 128-bit single-instruction multiple-data (SIMD) registers on the Pentium 4, while the other fields used only 32-bit registers. The rationale for limiting the wide registers to the genus-2 case was that "Great potential savings can be realized if an element of the base field can be represented in a single machine word, rather than using a multi-precision representation" and a factor 2 acceleration was reported for field multiplication via the wide registers.

This difference in implementations is especially significant since the Pentium 4 is 32-bit. The techniques are perhaps less elegant when applied to larger fields, but similar acceleration can be obtained via wide registers for the fields in the other pairings. For example, [20] examined the acceleration offered by SIMD registers for pairings at a higher security level using the fields $\mathbb{F}_{2^{1223}}$ and $\mathbb{F}_{3^{509}}$, but did not consider an example from genus 2. Beuchat et al. [3] and Aranha et al. [1] subsequently demonstrated significantly faster field arithmetic on platforms considered in [20].

In short, the implementation techniques for the times in [2] favour the genus-2 curve. If the registers used in $\mathbb{F}_{2^{103}}$ were applied to $\mathbb{F}_{2^{307}}$, then we would expect that the pairing times would be significantly closer. On the other hand, we are interested in the 128-bit security level, where the higher embedding degree of the genus-2 curve is an advantage. Our intent here is to give a meaningful comparison at the 128-bit security level among the various pairings on a "reference platform" using whatever methods are believed to be fastest in each scenario. The Pentium 4 is no longer of primary interest, and so we chose the popular 64-bit Intel Core2.

Timings for our implementations appear in Table 2. Pairings for the Barreto-Naehrig (BN) curve (see §3.1) over a prime field are expected to be fastest at this security level, in part because the embedding degree is 12 and the platform possesses a relatively fast integer multiplier on 64-bit operands. Details on this timing using the MIRACL library appear in [20].

Beuchat et al. [3] discuss optimization strategies and set benchmarks for pairing times in the elliptic curve cases over characteristic 2 and 3. As in [1], a focus is on parallelizing the pairing computation, although the times for a

Table 2. Timings (in clock cycles) on an Intel Core2. Field operations in characteristic 2 and 3 use 128-bit SIMD registers and exploit shift and shuffle instructions introduced with SSSE3. The timing for the BN curve is from [20].

Field, curve, and pairing	Field mult	Pairing
$E/\mathbb{F}_{p256}$, R-ate	.31	10
$E/\mathbb{F}_{3^{509}}$, η_T	3.86	15.8
$E/\mathbb{F}_{2^{1223}}$, η_T	3.84	19.0
$C/\mathbb{F}_{2^{439}}$, η_T on degenerate divisors	.86	16.4
	Units: 10^3 cycles	10^6 cycles

single-core computation were also impressive. The times in Table 2 are faster, but are consistent in the sense that characteristic 3 offers an advantage. On the other hand, this advantage is not as large as in [3], mainly due to the difference in characteristic 2 multiplication.

Compared with [2], applying the wide registers across fields has narrowed differences. Genus 2 has lost much of the performance advantage, although it may still be attractive from an implementation and keysize perspective for protocols having pairings on degenerate divisors. The gap between the pairing from the BN curve and those over characteristic 2 and 3 is perhaps narrower than expected.[3]

The experimental data in Table 1 gives a factor 3.4 penalty for a pairing on general divisors. In special cases, the cost will be less. Nondegenerate divisors $(P_1) + (P_2) - 2(\infty)$ are of two forms, either $P_i \in C(\mathbb{F}_q)$ or P_1 and P_2 are conjugates in $C(\mathbb{F}_{q^2}) \setminus C(\mathbb{F}_q)$. A pairing on divisors can be calculated as a product of pairings on points; e.g., in the case where $P_i \in C(\mathbb{F}_q)$, $\eta_T((P_1) + (P_2) - 2(\infty), (P) - (\infty)) = \eta_T(P_1, P)\eta_T(P_2, P)$ at twice the cost of a pairing on degenerate divisors. However, this approach may not be the most efficient when points lie in $C(\mathbb{F}_{q^2}) \setminus C(\mathbb{F}_q)$ [2]. Lee and Lee [26] give explicit formulas for the pairing on general divisors, with estimated cost (from field multiplications, where an $\mathbb{F}_{q^{12}}$-multiplication is counted as 45 $\mathbb{F}_q$-multiplications) as a factor 4 over the pairing on degenerate divisors.

Implementation notes. Compared with [3] and [20], the characteristic 2 multiplier (described in [1]) uses twice as much data-dependent precomputation but fewer shift operations. Some of the improvement against [20] was achieved by reducing the number of move operations (a weakness underestimated in [20]), although a portion of the acceleration was obtained by exploiting a shift operation introduced with the Supplemental Streaming SIMD Extension 3 (SSSE3).[4] The faster shift is also useful in characteristic 3 – additions are more expensive

[3] The comparison in [3] is against the slower ate pairing, which gives the timing for the BN curve as 15×10^6 cycles.

[4] SSSE3 was also exploited in [1] to obtain very fast squaring and root for a parallel implementation that performed these operations in excess; these accelerations give only minor reduction in pairing times here (e.g., 6% for the pairing over $\mathbb{F}_{2^{1223}}$).

than in characteristic 2, but field multiplication performs more shifting. Multiplication for $\mathbb{F}_{2^{1223}}$ is via one application of Karatsuba where elements are split at 616 bits (a multiple of 8 that allows fast shifting and eliminates the "fixup" required in combing on n-word input when both inputs have length greater than $nW - w$ for word-length W and comb width w). Multiplication in the 439-bit field is via combing directly on field elements. Combing uses two tables, each of 16 elements.

The strategies for characteristic 2 and 3 are similar, in part because an $\mathbb{F}_3$-element is represented as a pair (a_0, a_1) of bits and addition involves only bitwise operations. Harrison et al. [21] proposed an addition using 7 XOR ($\oplus$) and OR ($\vee$) operations via the sequence: $t \leftarrow (a_0 \vee b_1) \oplus (a_1 \vee b_0)$, $c_0 \leftarrow (a_1 \vee b_1) \oplus t$, $c_1 \leftarrow (a_0 \vee b_0) \oplus t$. The number of operations was reduced to 6 by Kawahara et al. [24] who reported 7–8% improvement in field multiplication on an AMD Opteron (a processor similar to the Intel Core2) using "non-standard" encodings of $\mathbb{F}_3$-elements. They also gave a 6-operation addition in the encoding suggested by [21] using XOR and ANDN ($x \wedge \bar{y}$).

The SIMD instruction set on the Core2 includes ANDN, although Beuchat et al. [3] reported that the 7-op addition "consistently yields a shorter computation time" than the 6-op variant with ANDN, and speculated that ANDN on the Core2 "is implemented less efficiently" than XOR and OR. However, these instructions have the same timings [14], and our experimental data is that field multiplication is faster with the 6-op variant. We suspect that the discrepancy is due to register allocation strategy in the accumulation portion of the multiplication method. A variation on the formulation in [21] is proposed in [3]: $t \leftarrow (a_0 \vee a_1) \wedge (b_0 \vee b_1)$, $c_0 \leftarrow (a_0 \vee b_0) \oplus t$, $c_1 \leftarrow (a_1 \vee b_1) \oplus t$. Specifics are not given on why this resulted in faster code, but we note that it permits simpler register tracking in the accumulation portion of field multiplication and has one operation on accumulator registers only. In this sense, the formulation in [21] and the 6-op variant for $c \leftarrow c + b$ are less pleasant. Compilers can be quite sensitive to the precise form of the code; however, the 6-op variant can be coded without increasing dependency chains, and we expect this formulation to be fastest provided that unnecessary moves are avoided. Experimentally, we observed roughly 10% faster times for field multiplication.

As in [3], we use the loop-unrolling technique of [19] along with the $\mathbb{F}_{3^{6m}}$ multiplication of [18] (requiring 15 multiplications and 67 additions in $\mathbb{F}_{3^m}$) to accelerate the pairing computation in characteristic 3. This reduces the cost from 14 to an effective 12.5 $\mathbb{F}_{3^m}$ multiplications in each iteration of the Miller loop. Some incremental accelerations noted in [3] were not implemented; for example, a few tables of precomputation in the evaluation loop of the pairing computation can be reused (a width-4 comb requires 81 elements of precomputation, although half are obtained by simple negation).

Timings were done on a 2.4 GHz Intel Core2-quad running Sun Solaris, using the GNU C 4.1 compiler with some fragments written in assembly. Most of the SIMD operations are via intrinsics, with SSSE3 instructions accessed via assembly.

2.2 Security of protocols using degenerate divisors

The principal motivation for considering hyperelliptic curves and degenerate divisors for pairing-based protocols is to speed the pairing computation at higher security levels. Naturally we need the assurance that the protocol, when implemented in this setting, maintains its original security guarantee. The question of security received some attention in [15, 2], however the main emphasis in those works is on efficient pairing computation. Here we take a closer look at the security argument of two well-known protocols when implemented in the setting of §2.1.

BLS signature scheme. We first describe the BLS signature scheme [6] using symmetric pairings (Type 1). We then present two variants of the scheme depending upon which particular elements are chosen to be degenerate and examine the resulting effect on the security argument.

Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ be a Type 1 pairing on a genus-2 curve, and let $H : \{0,1\}^* \to \mathbb{G}_1$ be a hash function. Let P_1 be a generator of $\mathbb{G}_1$. The public parameters of the system are $\langle \mathbb{G}_1, \mathbb{G}_T, H, P_1 \rangle$.

Alice's private key is an integer $x \in_R [0, n-1]$ and her corresponding public key is $X = xP_1$. To sign a message M, Alice computes $Q = H(M)$ and then $\sigma = xQ$ as her signature on M. To verify, Bob computes $Q = H(M)$ and accepts σ as a valid signature on M if and only if

$$e(\sigma, P_1) = e(Q, X). \tag{1}$$

Correctness of the verification algorithm follows because of the bilinearity property of e, i.e.,

$$e(\sigma, P_1) = e(xQ, P_1) = e(Q, xP_1) = e(Q, X).$$

Security of the scheme is based on the hardness of the computational Diffie-Hellman problem (DHP) in $\mathbb{G}_1$ assuming H to be a random oracle. Recall that the DHP in $\mathbb{G}_1 = \langle P_1 \rangle$ is the following: given X (where $X = xP_1$ for some $x \in_R [0, n-1]$) and $Q \in_R \mathbb{G}_1$, compute xQ. The essential ideas behind the reductionist security argument are as follows. Given a DHP instance (X, Q), the simulator sets the challenge public key as X and runs the BLS adversary $\mathcal{A}$. The simulator responds to all hash queries $H(M)$ made by $\mathcal{A}$, except for a randomly chosen distinguished query, by selecting $a \in_R [0, n-1]$ and setting $H(M) = aP_1$; the response to the distinguished hash query $H(M^*)$ is $H(M^*) = Q$. The simulator responds to signing queries $M \neq M^*$ by setting $\sigma = aX$. If $\mathcal{A}$ eventually produces a forged signature σ^* on M^*, then the simulator has been successful in obtaining the solution σ^* to the DHP instance (X, Q).

Recall from §2.1 that C is a supersingular genus-2 curve over $\mathbb{F}_q$, $q = 2^m$, and $\mathbb{G}_1$ is the set of n-torsion points in $J_C(\mathbb{F}_q)$ where $n \approx q^2$ is prime. Let $\mathcal{D}$ be the set of degenerate divisors in $\mathbb{G}_1$; then $\#\mathcal{D} \approx q$ since $\#C(\mathbb{F}_q) \approx q$. For efficient implementation we would like to have some (if possible all) of the elements of $\mathbb{G}_1$ used in the protocol to lie in $\mathcal{D}$. However, the choice is constrained

by how these elements are actually generated in the protocol, and whether the protocol environment can be properly simulated in the security argument. We further elaborate on these issues based on the following two versions of BLS. We use calligraphic fonts for degenerate divisors to distinguish them from general divisors.

BLS-1a: The key generation algorithm chooses a random element $\mathcal{P}_1$ of $\mathcal{D}$ as the system parameter and a hash function $H : \{0,1\}^* \to \mathcal{D}$. Both these tasks can be accomplished without any security penalty and is as efficient as working in the elliptic curve setting.[5] Then, with overwhelming probability, Alice's public key $x\mathcal{P}_1$ will *not* be a degenerate divisor. Since the range of H is $\mathcal{D}$, $\mathcal{Q} = H(M)$ is a degenerate divisor. But $\sigma = x\mathcal{Q}$ will likely be non-degenerate (with overwhelming probability). As a result, one of the arguments in each of the two pairing computations in the verification equation (1) is a degenerate divisor while the other is non-degenerate. This still makes the pairing computation faster compared to the case when both arguments are general divisors.

Next we investigate to what extent the original security argument of BLS is applicable in the case of BLS-1a. The DHP (with respect to the generator $\mathcal{P}_1$ of $\mathbb{G}_1$) is the problem of determining xQ, given $X = x\mathcal{P}_1 \in \mathbb{G}_1$ for some unknown $x \in_R [0, n-1]$ and $Q \in_R \mathbb{G}_1$. The natural choice would be to argue security of BLS-1a based on the hardness of the following variant of DHP. Given $X = x\mathcal{P}_1 \in \mathbb{G}_1$ for some unknown $x \in_R [0, n-1]$ and $\mathcal{Q} \in_R \mathcal{D}$, compute $x\mathcal{Q}$ — we call this problem DHP*. The following shows that DHP and DHP* are computationally equivalent.

Lemma 1. *The DHP and DHP* problems are computationally equivalent.*

Proof. It is clear that DHP* reduces to DHP. To prove the converse, suppose that we are given a DHP instance (X, Q) and an oracle for solving DHP*. If $Q \in \mathcal{D}$ then the DHP*-oracle can be used to compute xQ. If $Q \notin \mathcal{D}$, say $Q = (P_1)+(P_2)-2(\infty)$, there are two cases to consider. Let us say that Q is of type A if $P_1, P_2 \in C(\mathbb{F}_q)$ and of type B if $P_1, P_2 \in C(\mathbb{F}_{q^2}) \setminus C(\mathbb{F}_q)$.

Suppose first that Q is of type A. This case can be recognized because the Mumford representation (see [28]) of Q will take the form (a, b), where $a, b \in \mathbb{F}_q[x]$ with $\deg(a) = 2$, $\deg(b) \leq 1$, and where the roots of a belong to $\mathbb{F}_q$. More explicitly, if $a(x) = (x - u_1)(x - u_2)$ with $u_1, u_2 \in \mathbb{F}_q$, then $P_1 = (u_1, v_1)$ and $P_2 = (u_2, v_2)$ where $v_1 = b(u_1)$ and $v_2 = b(u_2)$. Thus, we can efficiently write $Q = \mathcal{Q}_1 + \mathcal{Q}_2$, where $\mathcal{Q}_1 = (P_1) - (\infty)$ and $\mathcal{Q}_2 = (P_2) - (\infty)$ are degenerate divisors. The DHP*-oracle can then be used to compute $x\mathcal{Q}_1$ and $x\mathcal{Q}_2$, from which $xQ = x\mathcal{Q}_1 + x\mathcal{Q}_2$ is immediately obtained.

[5] In [16, §7] the concern was raised that hashing to the set of degenerate divisors in $J_C(\mathbb{F}_q)$ instead of to the set of general divisors can lead to a loss security. This is because hash collisions in the former case can be found in $O(q^{1/2})$ time using generic algorithms, whereas collision finding in the latter case takes $O(q)$ time. However, the concern is not an issue in our setting with $q = 2^{439}$ because then $\sqrt{q} \approx 2^{219}$ which is significantly greater than the target security level of 2^{128}.

Suppose now that Q is of type B. In this case, we can multiply Q by randomly-selected integers $\ell \in [1, n-1]$ until the resulting divisor Q' is of type A. The expected number of trials is 2, since the number of type A divisors in $J_C(\mathbb{F}_q)$ is approximately $q^2/2$, as is the number of type B divisors in $J_C(\mathbb{F}_q)$ (this follows because $\#C(\mathbb{F}_q) \approx q$ and $\#C(\mathbb{F}_{q^2}) \approx q^2$). As above, one can then compute xQ' and hence $xQ = \ell^{-1}(xQ')$. $\square$

Now, given a DHP* instance $(X, \mathcal{Q})$, the simulator sets the challenge public key as X and interacts with the BLS-1a adversary $\mathcal{A}$. To properly answer $\mathcal{A}$'s signing query on a message M, the simulator has to "program" the random oracle in such a way that it outputs some $\mathcal{H} \in_R \mathcal{D}$ for which the simulator knows the discrete log with respect to $\mathcal{P}_1$. Recall that this was trivially accomplished for the original protocol — the simulator first chose $a \in_R [0, n-1]$ and then returned $a\mathcal{P}_1$. However, to apply this strategy in the simulation of BLS-1a, the simulator must satisfy the additional constraint that $a\mathcal{P}_1$ lies in $\mathcal{D}$.

The simulator could easily satisfy this condition if given some fixed $\mathcal{P} \in \mathcal{D}$ she has some mechanism to choose a random a such that $a\mathcal{P}$ also belongs to $\mathcal{D}$. The only known way to guarantee this in our genus-2 setting is to choose a to be a power of 8, i.e., if $\mathcal{P} \in \mathcal{D}$ then $8^i\mathcal{P}$ is also a degenerate divisor for any integer i. However, as the following lemma indicates, the hash output will then be confined to an extremely small subset of $\mathcal{D}$ (and thus the simulation will fail).

Lemma 2. *Let $\mathcal{P}$ be a degenerate divisor of order n in $J_C(\mathbb{F}_q)$, where C is the supersingular genus-2 curve over $\mathbb{F}_q$ (with $q = 2^m$) defined in §2.1. Then there are exactly $4m$ degenerate divisors of the form $8^i\mathcal{P}$.*

Proof. We have $q^{12} \equiv 1 \pmod{n}$, and so $8^{4m} \equiv 1 \pmod{n}$. Hence, the order of 8 modulo n is in $\{1, 2, 4, m, 2m, 4m\}$. Since $n > 8^4$, $8^m \not\equiv 1 \pmod{n}$ and $8^{2m} \not\equiv 1 \pmod{n}$, the order of 8 modulo n must be $4m$. $\square$

One way to circumvent this problem is to define a new problem which we call DHP* with oracle access and denote by DHP$^*_{\mathcal{O}}$. In addition to the DHP* instance $X = x\mathcal{P}_1$ and $\mathcal{Q}$, the solver (i.e., the BLS-1a simulator in the present context) is given access to an oracle $\mathcal{O}$. Each time it is invoked, the oracle $\mathcal{O}$ returns a random $\mathcal{P} \in \mathcal{D}$ along with $x\mathcal{P}$. It is easy to argue that the security of BLS-1a is equivalent to the hardness of DHP$^*_{\mathcal{O}}$. For example, when reducing DHP$^*_{\mathcal{O}}$ to the problem of breaking BLS-1a, the simulator returns $\mathcal{P}$ when $\mathcal{A}$ queries the random oracle on some message M, and subsequently $x\mathcal{P}$ in response to a signature query on M (where the simulator obtains $(\mathcal{P}, x\mathcal{P})$ from its oracle $\mathcal{O}$). At some point, $\mathcal{A}$ returns a valid forgery on some message M^* whose hash value has been set to $\mathcal{Q}$. The simulator returns this signature as the solution to the given DHP$^*_{\mathcal{O}}$ instance.

However, there is a circularity in the whole argument — the assumption that it is hard to solve DHP$^*_{\mathcal{O}}$ is nothing but a rephrasing of the assertion that it is hard to forge a BLS-1a signature. Currently we do not know any way out of this circularity based on the known security argument for BLS. Neither is there any evidence to suggest that BLS-1a is insecure.

Remark 1. Let $\mathcal{P}$ be a degenerate divisor of order n in $J_C(\mathbb{F}_q)$ with $n \approx q^2$, and let $\mathcal{D}$ denote the set of all degenerate divisors in $\langle \mathcal{P} \rangle$. The set of *degeneracy-preserving multipliers* is $\mathrm{DPM} = \{\ell \in [0, n-1] : \ell\mathcal{P} \in \mathcal{D}\}$. Lemma 2 says that $\{8^i \bmod n : 0 \leq i < 4m\} \subseteq \mathrm{DPM}$, while one would expect that $\#\mathrm{DPM} \approx n/q$. The interesting question is whether one can efficiently select an integer uniformly at random from DPM. If yes, then it easy to see that DHP^* and $\mathrm{DHP}^*_{\mathcal{O}}$ are computationally equivalent, and thus the security of BLS-1a can be proven based on the assumptions that DHP is hard and that H is a random function.

BLS-1b: Alternatively, we can keep the range of the hash function H to be $\mathbb{G}_1$ and choose only the fixed system parameter $\mathcal{P}_1$ to be a degenerate divisor. With this modification, we still make some efficiency gains in the verification algorithm namely in the evaluation of $e(\sigma, \mathcal{P}_1)$. The known security argument for BLS with respect to DHP can now be easily adapted for BLS-1b.

Boneh-Franklin identity-based encryption scheme. The situation is similar for the BF-IBE scheme [5]. We assume the reader is familiar with the basic idea of the protocol. Suppose that the public parameters are $\langle \mathbb{G}_1, \mathbb{G}_T, H, P_1 \rangle$, the Key Generation Centre's public key is $D_{\mathsf{pub}} \in \mathbb{G}_1$, and the public key corresponding to an arbitrary identity ID is obtained as $Q_{\mathsf{ID}} = H(\mathsf{ID})$. Encryption involves the computation of a pairing value $e(D_{\mathsf{pub}}, Q_{\mathsf{ID}})$. The hardness of BF-IBE is based on the so-called bilinear Diffie-Hellman (BDH) problem — given aP_1, bP_1, cP_1 for $a, b, c \in_R [0, n-1]$, compute $e(P_1, P_1)^{abc}$.

In [2], Barreto et al. suggest that without loss of security it is possible to choose both D_{pub} and Q_{ID} to be degenerate divisors so that encryption involves pairing of two degenerate divisors. We note that the known security argument of BF-IBE suffers from the same problem that we encountered in BLS-1a, namely it is not possible to simulate the random oracle H with range $\mathcal{D}$. The security argument does however go through if one chooses only D_{pub} to be degenerate. Now in the encryption algorithm only one of the arguments to the pairing function is a degenerate divisor while the other is a general divisor. In this case, the corresponding instance of the BDH problem contains one degenerate divisor and two general divisors. As was done in Lemma 1, one can prove that this variant of BDH is equivalent to the original BDH problem.

Remark 2. We have not found any pairing-based protocols in the literature that can be implemented so that both arguments to one or more of the pairing functions are degenerate divisors, and where the original security argument in the elliptic curve setting can be carried over to the genus-2 setting. Thus, the speed benefits of using pairings in the genus-2 setting where both arguments are degenerate divisors do not seem to be directly applicable to known protocols.

3 Type 4 pairings

Let E be an ordinary elliptic curve defined over the finite field $\mathbb{F}_q$. Let n be a prime divisor of $\#E(\mathbb{F}_q)$ satisfying $\gcd(n, q) = 1$, and let k (the embedding

degree) be the smallest positive integer such that $n \mid q^k - 1$. We will assume that k is even. Since $k > 1$, we have $E[n] \subseteq E(\mathbb{F}_{q^k})$. We will further assume that $n^3 \nmid \#E(\mathbb{F}_{q^k})$. Let $\mathbb{G}_T$ be the order-n subgroup of $\mathbb{F}_{q^k}^*$. The (full) Tate pairing is a non-degenerate bilinear function $\hat{e} : E[n] \times E[n] \to \mathbb{G}_T$ and can be defined as follows:

$$\hat{e}(P, Q) = \left(\frac{f_{n,P}(Q + R)}{f_{n,P}(R)} \right)^{(q^k - 1)/n}, \tag{2}$$

where $R \in E(\mathbb{F}_{q^k})$ with $R \notin \{\infty, P, -Q, P - Q\}$, and where the *Miller function* $f_{n,P}$ is a function whose only zeros and poles in E are a zero of order n at P and a pole of order n at ∞.

Let $\mathbb{G}_1 = E(\mathbb{F}_q)[n]$. If the first component of the domain of $\hat{e}$ is restricted to $\mathbb{G}_1$, then the definition of $\hat{e} : \mathbb{G}_1 \times E[n] \to \mathbb{G}_T$ simplifies to $\hat{e}(P, Q) = (f_{n,P}(Q))^{(q^k - 1)/n}$. Such a mapping $\hat{e}$ is called a *Type 4 pairing* [32] because the second component of the domain of $\hat{e}$ is the full n-torsion group $E[n]$. The Trace function Tr defined by $\mathrm{Tr}(P) = \sum_{i=0}^{k-1} \pi^i(P)$, where π denotes the q-th power Frobenius, is an efficiently-computable homomorphism from $E[n]$ to $\mathbb{G}_1$. The kernel of Tr, called the *Trace-0 group*, is an order-n subgroup of $E[n]$. Hashing onto $\mathbb{G}_1$ can be efficiently computed by first hashing to an x-coordinate of $E(\mathbb{F}_q)$, then solving a quadratic equation over $\mathbb{F}_q$ to find the corresponding y-coordinate, and finally multiplying the resulting point by the cofactor $h_1 = \#E(\mathbb{F}_q)/n$ to obtain an n-torsion point. Hashing onto $E[n]$ can be accomplished in a similar fashion, by first hashing onto a random point in $E(\mathbb{F}_{q^k})$ and then multiplying by the cofactor $h_k = \#E(\mathbb{F}_{q^k})/n^2$. However, hashing onto $E[n]$ is considerably more expensive than hashing onto $\mathbb{G}_1$ since computations now take place in the larger field $\mathbb{F}_{q^k}$ instead of in $\mathbb{F}_q$, and moreover the cofactor $h_k \approx q^{k-2}$ can be quite large. For the case of BN curves, Chen, Cheng and Smart [12] estimated that the cost of hashing onto $E[n]$ is about 540 times that of performing a point multiplication in $\mathbb{G}_1$ (and estimated the cost of hashing onto $\mathbb{G}_1$ as "free"). This expensive hashing is a major drawback of Type 4 pairings. In the next section, we show that the representation for $E[n]$ introduced in [10] can be used to speed hashing into $E[n]$. In particular, for the case of BN curves, we estimate that our new method for hashing into $E[n]$ is less than 3 times as costly as a point multiplication in $\mathbb{G}_1$.

3.1 On efficient implementation

Following [17], we denote by D the CM discriminant of E and set

$$e = \begin{cases} \gcd(k, 6), & \text{if } D = -3, \\ \gcd(k, 4), & \text{if } D = -4, \\ 2, & \text{if } D < -4, \end{cases} \tag{3}$$

and $d = k/e$. Then E has a unique degree-e twist $\tilde{E}$ defined over $\mathbb{F}_{q^d}$ such that $n \mid \#\tilde{E}(\mathbb{F}_{q^d})$ [22]. Let $\tilde{P}_2 \in \tilde{E}(\mathbb{F}_{q^d})$ be a point of order n, and let $\tilde{\mathbb{T}}_0 = \langle \tilde{P}_2 \rangle$. Then there is a monomorphism $\phi : \tilde{\mathbb{T}}_0 \to E(\mathbb{F}_{q^k})$ such that $P_2 = \phi(\tilde{P}_2) \notin \mathbb{G}_1$.

The group $\mathbb{T}_0 = \langle P_2 \rangle$ is the Trace-0 subgroup of $E[n]$. The monomorphism ϕ can be defined so that $\phi : \tilde{\mathbb{T}}_0 \to \mathbb{T}_0$ can be efficiently computed in both directions; therefore we can identify $\tilde{\mathbb{T}}_0$ and $\mathbb{T}_0$, and consequently $\mathbb{T}_0$ can be viewed as having coordinates in $\mathbb{F}_{q^d}$ (instead of in the larger field $\mathbb{F}_{q^k}$).

We have $E[n] \cong \mathbb{G}_1 \times \mathbb{T}_0$. Define the homomorphism $\psi : E[n] \to \mathbb{G}_1$ by $\psi(Q) = \frac{1}{k}\mathrm{Tr}(Q)$. Then it is easy to verify that $Q - \psi(Q) \in \mathbb{T}_0$ for all $Q \in E[n]$ and consequently the map $\rho : Q \mapsto Q - \psi(Q)$ is a homomorphism from $E[n]$ onto $\mathbb{T}_0$. Thus, the map $\phi : E[n] \to \mathbb{G}_1 \times \mathbb{T}_0$ defined by $\phi(Q) = (\psi(Q), \rho(Q))$ is an efficiently-computable isomorphism, whose inverse, given by $(Q_1, Q_2) \mapsto Q_1 + Q_2$, is also efficiently computable. Hence, without loss of generality, the elements of $E[n]$ can be represented as pairs of points (Q_1, Q_2), where $Q_1 \in \mathbb{G}_1$ and $Q_2 \in \mathbb{T}_0$.

With this representation for $E[n]$, hashing onto $E[n]$ can be defined as $H(m) = (H_1(m), H_2(m))$, where H_1 and H_2 are hash functions with ranges $\mathbb{G}_1$ and $\mathbb{T}_0$, respectively. This is expected to be faster than the conventional hashing method outlined in the beginning of this section because hashing onto $\mathbb{G}_1$ and $\mathbb{T}_0$ requires arithmetic in $\mathbb{F}_q$ and $\mathbb{F}_{q^d}$, respectively, rather than in $\mathbb{F}_{q^k}$. Observe that if H_1 and H_2 are modeled as random oracles, then H is also a random oracle.

The ate [22] and R-ate [25] pairings are fast Type 3 pairings from $\mathbb{G}_1 \times \mathbb{T}_0$ to $\mathbb{G}_T$ defined by $e_3(P, Q) = \hat{e}(Q, P)^N$ for some fixed integer N. Now, define the Type 4 pairing $e_4 : \mathbb{G}_1 \times E[n] \to \mathbb{G}_T$ by $e_4(P, Q) = e_3(P, \hat{Q})$, where $\hat{Q} = Q - \pi^{k/2}(Q)$. Note that if $Q = (Q_1, Q_2)$, then $\hat{Q} = (\infty, 2Q_2)$. Thus, e_4 is a bilinear pairing and can be computed in essentially the same time as the Type 3 pairing e_3. The pairing e_4 is non-degenerate in the sense that (i) for each $P \in \mathbb{G}_1 \setminus \{\infty\}$, there exists $Q \in E[n]$ such that $e_4(P, Q) \neq 1$; and (ii) for each $Q \in E[n] \setminus \mathbb{G}_1$, there exists $P \in \mathbb{G}_1$ such that $e_4(P, Q) \neq 1$.

Remark 3. In cryptographic applications of Type 4 pairings, one can ensure that hash values $H(m) = (H_1(m), H_2(m))$ do not lie in $\mathbb{G}_1$ or $\mathbb{T}_0$ by defining H_1 and H_2 to have ranges $\mathbb{G}_1 \setminus \{\infty\}$ and $\mathbb{T}_0 \setminus \{\infty\}$, respectively. This ensures that $\psi(H(m)) \neq \infty$ and $e_4(P, H(m)) \neq 1$ for $P \neq \infty$.

For concreteness, we consider the BN curve $E/\mathbb{F}_p : Y^2 = X^2 + 3$ with BN parameters $z =$ `6000000000001F2D` that was studied in [10]. This curve has the property that $n = \#E(\mathbb{F}_p)$ is a 256-bit prime, and the embedding degree is $k = 12$. For this particular curve, Table 3 lists the bitlengths of elements in $\mathbb{G}_1$, $\mathbb{T}_0$, $E[n]$ and $\mathbb{G}_T$, and the estimated costs of performing essential operations in these groups; for detailed explanations see [10]. Table 3 demonstrates that Type 4 pairings have very similar performance attributes as Type 3 pairings.

3.2 On secure implementation

As we have observed earlier, the only motivation to consider the Type 4 setting for implementation of a protocol is when the protocol requires hashing into the second component $\mathbb{G}_2$ of the pairing's domain followed by an application of ψ on the hash output. However, for Type 4 pairings, $\mathbb{G}_2 = E[n]$ has order n^2, which

Table 3. Bitlengths of elements in $\mathbb{G}_1$, $\mathbb{T}_0$, $E[n]$ and $\mathbb{G}_T$, and estimated costs (in terms of $\mathbb{F}_p$ multiplications) of basic operations for Type 3 and Type 4 pairings derived from a particular BN elliptic curve.

	Type 3	Type 4
Bitlength of elements in $\mathbb{G}_1$	257	257
Bitlength of elements in $\mathbb{T}_0/E[n]$	513	770
Bitlength of elements in $\mathbb{G}_T$	1,024	1,024
Compressing elements in $\mathbb{G}_1$	free	free
Compressing elements in $\mathbb{T}_0/E[n]$	free	free
Decompressing elements in $\mathbb{G}_1$	$315m$	$315m$
Decompressing elements in $\mathbb{T}_0/E[n]$	$674m$	$989m$
Addition in $\mathbb{G}_1$	$11m$	$11m$
Doubling in $\mathbb{G}_1$	$7m$	$7m$
Addition in $\mathbb{T}_0/E[n]$	$30m$	$41m$
Doubling in $\mathbb{T}_0/E[n]$	$17m$	$24m$
Exponentiation in $\mathbb{G}_1$	$1{,}533m$	$1{,}533m$
Exponentiation in $\mathbb{T}_0/E[n]$	$3{,}052m$	$4{,}585m$
Fixed-base exponentiation in $\mathbb{T}_0$	$718m$	$718m$
Fixed-base exponentiation in $\mathbb{T}_0/E[n]$	$1{,}906m$	$2{,}624m$
Hashing into $\mathbb{G}_1$	$315m$	$315m$
Hashing into $\mathbb{T}_0/E[n]$	$3{,}726m$	$4{,}041m$
e_n/R_n Pairing	$15{,}175m$	$15{,}175m$
Testing membership in $\mathbb{G}_1$	free	free
Testing membership in $\mathbb{T}_0/E[n]$	$3{,}052m$	$3{,}052m$

can be a fundamental distinction affecting the functionality and security of a protocol in the Type 4 setting. We demonstrate this with the very first protocol for which the Type 4 setting was introduced in the literature.

Boneh-Shacham group signature scheme. In a group signature scheme every member of the group has a secret key but there is a single public key for the entire group. The signer-anonymity property of such a scheme finds application, for example, in privacy preserving attestation [7]. Revocation of a user may be critical for such an application, e.g., when the user's secret key is compromised.

Boneh and Shacham proposed a short group signature scheme [7] with an interesting property that given a list of revoked users a verifier can locally check whether the signature has been generated by one of them. This is called a verifier-local revocation (VLR) group signature. They defined a security model for VLR group signature, proposed a construction based on asymmetric pairings, provided a security proof, and discussed the efficiency of the scheme in the elliptic curve setting.

The original description [7] of the Boneh-Shacham group signature scheme (BS-VLR) includes a hash function whose range is $\mathbb{G}_2 \times \mathbb{G}_2$ and also employs the map ψ on the components of the outputs of this hash function. As noted elsewhere [33, 11], this protocol cannot be implemented in either the Type 2 or

the Type 3 setting. Later in his Ph.D. thesis [32], Shacham introduced Type 4 pairings and reproduced the BS-VLR scheme in that setting without further modification.

Here we take another look at the BS-VLR scheme from [32] and demonstrate that the protocol as described does not achieve its desired functionality and in fact is *not* secure. The protocol description is quite involved and so is its security proof. We recall only those parts of the protocol that are relevant to our discussion. Interested readers are referred to §7.4 of Shacham's thesis [32] as well as the original paper of Boneh and Shacham [7] for the elaborate details.

BS-VLR group signature scheme: The protocol employs a Type 4 pairing $e : \mathbb{G}_1 \times E[n] \to \mathbb{G}_T$. In order to maintain consistency with [7, 32], we use multiplicative notation for $\mathbb{G}_1$ and $E[n]$. The group public key is $gpk = (g_1, g_2, w)$, where $g_2 \in_R E[n]$, $g_1 = \psi(g_2)$, and $w = g_2^{\gamma}$ for some $\gamma \in_R [1, n-1]$. Suppose that the group consists of N members. The private key of the ith member is $gsk[i] = (A_i, x_i)$ where $x_i \in_R [1, n-1]$ and $A_i = g_1^{1/(\gamma + x_i)}$. The revocation token corresponding to this private key is A_i which is made public in a revocation list (RL) when membership of i is revoked from the group.

The signer i computes, among other items, two elements $T_1, T_2 \in \mathbb{G}_1$ in the following way:

1. $(\hat{u}, \hat{v}) \leftarrow H_0(gpk, M, r)$ where M is the message to be signed, $r \in_R [1, n-1]$ is a nonce, and H_0 is a hash function with range $E[n] \times E[n]$ (treated as a random oracle in the security proof); cf. Remark 3.
2. $u \leftarrow \psi(\hat{u})$ and $v \leftarrow \psi(\hat{v})$.
3. $T_1 \leftarrow u^{\alpha}$ and $T_2 \leftarrow A_i v^{\alpha}$, where $\alpha \in_R [1, n-1]$.

T_1, T_2 and r are then sent as part of the group signature σ on M. Note that given r, the verifier can easily obtain $\hat{u}$ and $\hat{v}$.

Verification is a two-step procedure — signature check and revocation check. The signature is accepted as valid only if both these checks are successful. We do not describe the first step where the verifier performs the standard check for validity of the signature σ on M under the group public key gpk. The insecurity of the protocol lies in the revocation check step, which we reproduce verbatim from [32].

> **Revocation check.** For each element $A \in \mathsf{RL}$, check whether A is encoded in (T_1, T_2) by checking if
>
> $$e(T_2/A, \hat{u}) \stackrel{?}{=} e(T_1, \hat{v}). \tag{4}$$
>
> If no element of RL is encoded in (T_1, T_2), the signer of σ has not been revoked.

In other words, suppose the group member i who generated the signature has already been revoked, i.e., $A_i \in \mathsf{RL}$. Then for $A = A_i$, the left side of (4) becomes

$$e(T_2/A_i, \hat{u}) = e(v^{\alpha}, \hat{u}) = e(v, \hat{u})^{\alpha},$$

while the right side becomes

$$e(T_1, \hat{v}) = e(u^\alpha, \hat{v}) = e(u, \hat{v})^\alpha.$$

It is assumed that $e(v, \hat{u})^\alpha$ and $e(u, \hat{v})^\alpha$ are equal, in which case equation (4) holds. As a result the verifier can link the signature to the revoked member i and hence reject it.

In fact, for a signature generated by a revoked user, equation (4) trivially holds if we are in the Type 2 or Type 3 settings where $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic groups of the same prime order n. (Simply write $\hat{u} = \hat{v}^x$ for some $x \in [0, n-1]$, and note that $u = \psi(\hat{u}) = \psi(\hat{v}^x) = v^x$.) But recall that the protocol is now described in the Type 4 setting where $\mathbb{G}_2 = E[n]$ is a group of order n^2. Notice that $E[n]$ has $n+1$ different subgroups of order n, two of which are $\mathbb{G}_1$ and $\mathbb{T}_0$. Suppose that $\mathbb{T}$ is any order-n subgroup of $E[n]$ other than $\mathbb{G}_1$ and $\mathbb{T}_0$. In the Type 4 setting, if $\hat{u}$ and $\hat{v}$ are in the same subgroup $\mathbb{T}$, then equation (4) holds. Conversely, if $\hat{u}$ and $\hat{v}$ are in different subgroups $\mathbb{T}$, then equation (4) only holds with negligible probability. However, $\hat{u}$ and $\hat{v}$ are obtained through hashing to random points in $E[n]$, and so the probability that they both belong to the same subgroup $\mathbb{T}$ is negligible. In fact, the inability to deterministically hash to a particular subgroup $\mathbb{T}$ is the sole reason to describe the protocol in the Type 4 setting instead of Type 2. So with an overwhelming probability equation (4) will not be satisfied and a signature generated by the revoked member i will pass the revocation check.

The security definition of the BS-VLR signature scheme requires that the protocol must satisfy the correctness, traceability, and selfless-anonymity properties, of which the first two are relevant to our discussion. Informally speaking, correctness means that every properly generated signature is accepted as valid if and only if the corresponding signer is not revoked, whereas traceability means that an adversary should not be able to forge a signature that cannot be traced to a revoked user. Neither of these is satisfied for the BS-VLR signature scheme as we have already explained. The wrong assertion in Theorem 7.4.4 of [32] regarding the correctness of the scheme renders the proof of Theorem 7.4.8 regarding traceability meaningless.

Modified BS-VLR signature scheme in Type 4: We now describe a small modification to the protocol that appears to restore security. One apparent drawback is that the signature in the modified protocol contains an element of $E[n]$ and may no longer be considered as "short", which was one of the original motivations of the construction. Fortunately, our new representation of $E[n]$ as discussed in §3.1 (cf. Table 3) helps to maintain the relatively small signature length.

To begin with, we note that the problem with the original protocol [32] does not stem from any intrinsic structural weakness. Rather, it is because of a technical issue related to the structure of $\mathbb{G}_2$. For example, it is possible to securely implement the protocol in the Type 1 setting (where $\mathbb{G}_2 = \mathbb{G}_1$) though the signature length increases. To keep this length short one has to work in the asymmetric setting; and since the protocol requires both hashing into $\mathbb{G}_2$ and the

map ψ, the only known option is Type 4. However, that means $\mathbb{G}_2$ is no longer a cyclic group of prime order n, but a group of order n^2. Hence, we cannot expect that two (or more) randomly generated elements will lie in the same order-n subgroup $\mathbb{T}$ of $\mathbb{G}_2$.

Keeping this in mind, the problem of the BS-VLR signature scheme in Type 4 can be easily fixed with a simple modification. The essential idea is the following. For $\hat{u}, \hat{v} \in_R E[n]$, even though in general one cannot expect that $e(\psi(\hat{v}), \hat{u})$ will be equal to $e(\psi(\hat{u}), \hat{v})$, bilinearity of e ensures that

$$e(\psi(\hat{v})^\alpha, \hat{u}) = e(\psi(\hat{v}), \hat{u}^\alpha) \tag{5}$$

for all $\alpha \in [0, n-1]$. So we make the following changes to the protocol.

1. The key generation algorithm remains unchanged. But note that g_2 is a random order-n element of $E[n]$ which can be obtained by hashing into $E[n]$ as discussed in §3.1.
2. The hash function H_0 has range $E[n] \times \mathbb{G}_1$ (instead of $E[n] \times E[n]$).
3. In the signing algorithm, compute $(\hat{u}, v) = H_0(gpk, M, r)$ and $\hat{T}_1 = \hat{u}^\alpha$. Then use $\hat{T}_1$ and $\hat{u}$ to compute the helper value $R_3 = \hat{T}_1^{r_x} \cdot \hat{u}^{-r_\delta} \in E[n]$, and use $\hat{T}_1$ to compute the challenge value c. Send $\hat{T}_1$ (*not* T_1) as part of the signature.
4. In the verification algorithm, use $\hat{T}_1$ and $\hat{u}$ (instead of T_1 and u) to rederive $R_3 \in E[n]$, use $T_1 = \psi(\hat{T}_1)$ to rederive R_1 in the signature check process, and use $\hat{T}_1$ to rederive the challenge value c. Use $\hat{T}_1$ (*not* T_1) in the revocation check step, i.e., for each $A \in \mathsf{RL}$, determine whether A is encoded in $(\hat{T}_1, T_2)$ by checking if
$$e(T_2/A, \hat{u}) \stackrel{?}{=} e(v, \hat{T}_1). \tag{6}$$

The only noticeable differences with the original scheme is that the signature now contains $\hat{T}_1 \in E[n]$ instead of $T_1 \in \mathbb{G}_1$ and the revocation check is performed based on $\hat{T}_1$. We briefly analyze the resulting effect on the security.

It is easy to check that the modified scheme satisfies the correctness property. For a signature generated by an honest user, the original argument of Theorem 7.4.4 in [32] applies when $\hat{T}_1$ (and not T_1) is sent as part of the signature. In particular, for a signature generated by a revoked member, equation (6) is exactly in the form of (5) and hence that signature will not pass the revocation check and will be rejected.

The selfless-anonymity and traceability properties of the original scheme are established through involved reductionist security arguments in Lemma 7.4.7 and Theorem 7.4.8 of [32]. Recall that the traceability property is violated because the correctness property does not hold for the original scheme. In fact we do not find any flaw per se in the proofs of these two theorems if we assume that $\mathbb{G}_2$ is a group of prime order n.

However, in the Type 4 setting $\mathbb{G}_2$ is the set of all n-torsion points $E[n]$, which is a group of order n^2. Still it is possible to carry over the original security arguments with some modifications. We do not reproduce the complete arguments here but only emphasize that in the selfless-anonymity game (Lemma 7.4.7) the following variant of the Decision Linear problem should be used: Given a 6-tuple

$(u_0, u_1, h_0 = u_0^a, h_1 = u_1^b, v, Z)$, where $u_0, u_1 \in_R E[n]$, $a, b \in_R [1, n-1]$, $v \in_R \mathbb{G}_1$, and either $Z = v^{a+b}$ or $Z \in_R \mathbb{G}_1$, decide whether $Z = v^{a+b}$. Furthermore, the elements u_0 and u_1 have to be appropriately randomized when answering signature queries on behalf of users i_0 and i_1. This randomization is possible because elements of $E[n]$ can be represented as described in §3.1. A complete description of the modified protocol and arguments for its security are provided in the Appendix.

We have identified two other protocols in the literature that extend or apply the idea of the BS-VLR signature scheme. These are the VLR signature with backward unlinkability due to Nakanishi and Funabiki [29] and the remote biometric authentication protocol due to Bringer et al. [8]. All our observations regarding the BS-VLR scheme apply to these protocols as well.

Protocols that employ asymmetric pairings and which utilize hashing into $\mathbb{G}_2$ followed by an application of the map ψ can only be instantiated in the Type 4 setting. In fact, to the best of our understanding, one should only resort to Type 4 for these kinds of protocols, since any other protocol employing an asymmetric pairing can be more efficiently instantiated in the Type 3 setting. However, when describing a protocol in the Type 4 setting or arguing its security, protocol designers should be cautious of the fact that $\mathbb{G}_2$ is no longer a prime-order group like $\mathbb{G}_1$ or $\mathbb{G}_T$. Not doing so may critically affect the functionality and security of the protocol as illustrated by the examination of BS-VLR.

4 Concluding remarks

We presented the first timings for Type 1 pairings derived from supersingular genus-2 curves in characteristic 2 at the 128-bit level, and showed that hashing to the group $\mathbb{G}_2$ in Type 4 pairings is not nearly as costly as previously believed. Furthermore, we demonstrated some pitfalls that can arise when designing protocols and formulating reductionist security arguments in the Type 1 and Type 4 settings.

References

1. D. Aranha, J. López, and D. Hankerson. "High-speed parallel software implementation of the η_T pairing", *RSA Cryptographers' Track (CT-RSA 2010)*, LNCS 5985 (2010), 89-105.
2. P. Barreto, S. Galbraith, C. Ó hÉigeartaigh, and M. Scott, "Efficient pairing computation on supersingular abelian varieties", *Designs, Codes and Cryptography*, 42 (2007), 239–271.
3. J.-L. Beuchat, E. López-Trejo, L. Martínez-Ramos, S. Mitsunari, and F. Rodríguez-Henríquez, "Multi-core implementation of the Tate pairing over supersingular elliptic curves", *Cryptology and Network Security (CANS 2009)*, LNCS 5888 (2009), 413–432. See also http://eprint.iacr.org/2009/276.
4. D. Boneh, X. Boyen, and H. Shacham, "Short group signatures", *Advances in Cryptology – CRYPTO 2004*, LNCS 3152 (2004), 41–55.

5. D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing", *SIAM Journal on Computing*, 32 (2003), 586–615.
6. D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing", *Journal of Cryptology*, 17 (2004), 297–319.
7. D. Boneh and H. Shacham, "Group signatures with verifier-local revocation", *11th ACM Conference on Computer and Communications Security – CCS 2004*, 168–177, 2004.
8. J. Bringer, H. Chabanne, D. Pointcheval, and S. Zimmer, "An application of the Boneh and Shacham group signature scheme to biometric authentication", *International Workshop on Security – IWSEC 2008*, LNCS 5312 (2008), 219–230.
9. J. Camenisch, S. Hohenberger, and A. Lysyanskaya, "Compact E-cash", *Advances in Cryptology – EUROCRYPT 2005*, LNCS 3494 (2005), 302–321.
10. S. Chatterjee, D. Hankerson, E. Knapp, and A. Menezes. "Comparing two pairing-based aggregate signature schemes", *Designs, Codes and Cryptography*, 55 (2010), 141-167.
11. S. Chatterjee and A. Menezes, "On cryptographic protocols employing asymmetric pairings – the role of ψ revisited", Cryptology ePrint Archive, Report 2009/480, 2009.
12. L. Chen, Z. Cheng, and N. Smart, "Identity-based key agreement protocols from pairings", *International Journal of Information Security*, 6 (2007), 213–241.
13. C. Delerablée and D. Pointcheval, "Dynamic fully anonymous short group signatures", *Progress in Cryptology – VIETCRYPT 2006*, LNCS 4341 (2006), 193–210.
14. A. Fog, *Instruction tables: Lists of instruction latencies, throughputs and micro-operation breakdowns for Intel, AMD and VIA CPUs*, http://www.agner.org/optimize/, 2009.
15. G. Frey and T. Lange, "Fast bilinear maps from the Tate-Lichtenbaum pairing on hyperelliptic curves", *Algorithmic Number Theory: 7th International Symposium, ANTS-VII*, LNCS 4076 (2006), 466–479.
16. S. Galbraith, F. Hess, and F. Vercauteren, "Hyperelliptic pairings", *Pairing-Based Cryptography – Pairing 2007*, LNCS 4575 (2007), 108–131.
17. S. Galbraith, K. Paterson, and N. Smart, "Pairings for cryptographers", *Discrete Applied Mathematics*, 156 (2008), 3113–3121.
18. E. Gorla, C. Puttmann, and J. Shokrollahi, "Explicit formulas for efficient multiplication in $\mathbb{F}_{3^{6m}}$", *Selected Areas in Cryptography (SAC 2007)*, LNCS 4876 (2007), 173–183.
19. R. Granger, D. Page, and M. Stam, "On small characteristic algebraic tori in pairing-based cryptography", *LMS Journal of Computation and Mathematics*, 9 (2006), 64–85.
20. D. Hankerson, A. Menezes, and M. Scott, "Software implementation of pairings", In M. Joye and G. Neven, editors, *Identity-Based Cryptography*, IOS Press, 2008.
21. K. Harrison, D. Page, and N. P. Smart, "Software implementation of finite fields of characteristic three, for use in pairing-based cryptosystems", *LMS Journal of Computation and Mathematics*, 5 (200), 181–193.
22. F. Hess, N. Smart, and F. Vercauteren, "The eta pairing revisited", *IEEE Trans. Information Theory*, 52 (2006), 4595–4602.
23. A. Joux, "A one round protocol for tripartite Diffie-Hellman", *Journal of Cryptology*, 17 (2004), 263–276.
24. Y. Kawahara, K. Aoki, and T. Takagi, "Faster implementation of η_T pairing over $GF(3^m)$ using minimum number of logical instructions for $GF(3)$-addition", *Pairing-Based Cryptography – Pairing 2008*, LNCS 5209 (2008), 282–296.

25. E. Lee, H. Lee, and C. Park, "Efficient and generalized pairing computation on abelian varieties", *IEEE Trans. Information Theory*, 55 (2009), 1793–1803.
26. E. Lee and Y. Lee, "Tate pairing computation on the divisors of hyperelliptic curves of genus 2", *Journal of the Korean Mathematical Society*, 45 (2008), 1057–1073.
27. A. Lenstra, "Unbelievable security: Matching AES security using public key systems", *Advances in Cryptology – ASIACRYPT 2001*, LNCS 2248 (2001). 67–86.
28. A. Menezes, Y. Wu, and R. Zuccherato, "An elementary introduction to hyperelliptic curves', Appendix in *Algebraic Aspects of Cryptography*, Springer, 1998.
29. T. Nakanishi and N. Funabiki, "A short verifier-local revocation group signature scheme with backward unlinkability", *International Workshop on Security – IWSEC 2006*, LNCS 4266 (2006), 17–32.
30. C. Ó hÉigeartaigh, *Pairing computation on hyperelliptic curves of genus 2*, PhD thesis, Dublin City University, 2006.
31. C. Ó hÉigeartaigh and M. Scott, "Pairing calculation on supersingular genus 2 curves", *Selected Areas in Cryptography (SAC 2006)*, LNCS 4356 (2007), 302–316.
32. H. Shacham, *New paradigms in signature schemes*, PhD thesis, Stanford University, 2005.
33. N. Smart and F. Vercauteren, "On computable isomorphisms in efficient pairing-based systems", *Discrete Applied Mathematics*, 155 (2007), 538–547.

A Appendix

A.1 The modified Boneh-Shacham group signature protocol

The protocol employs a Type 4 pairing $e : \mathbb{G}_1 \times E[n] \rightarrow \mathbb{G}_T$ and two hash functions H_0, H where $H_0 : \{0,1\}^* \rightarrow E[n] \times \mathbb{G}_1$ and $H : \{0,1\}^* \rightarrow [1, n-1]$. Note that an element $Q \in E[n]$ is represented as a pair of points $(Q_1, Q_2) \in \mathbb{G}_1 \times \mathbb{T}_0$, where $\mathbb{T}_0$ is the Trace-0 group of $E[n]$ and $\psi(Q) = Q_1$, which means ψ can be evaluated free of cost.

KeyGen(N): The algorithm takes as input N, the number of members in the group and proceeds as follows.

1. Select $g_2 \in_R E[n]$ and set $g_1 = \psi(g_2)$.
2. Select $\gamma \in_R [1, n-1]$ and set $w = g_2^{\gamma}$.
3. For each user i, generate the private key (A_i, x_i) by selecting $x_i \in_R [1, n-1]$ such that $\gamma + x_i \neq 0$ and then setting $A_i = g_1^{1/\gamma + x_i}$.

The group public key is $gpk = (g_1, g_2, w)$ while user i's private key is the tuple $\mathbf{gsk}[i] = (A_i, x_i)$. The revocation token corresponding to $\mathbf{gsk}[i]$ is $\mathbf{grt}[i] = A_i$. The algorithm outputs $(gpk, \mathbf{gsk}, \mathbf{grt})$ where $\mathbf{gsk}$ (resp. $\mathbf{grt}$) is the set of private keys (resp. revocation tokens) of the N members of the group. Note that the secret value γ is known only to the private key issuer.

Sign($gpk, \mathbf{gsk}[i], M$): The algorithm proceeds as follows.

1. Pick a random nonce $r \in [1, n-1]$ and compute $(\hat{u}, v) = H_0(gpk, M, r)$ and set $u = \psi(\hat{u})$.

2. Compute $\hat{T}_1 = \hat{u}^{\alpha}$ and $T_2 = A_i v^{\alpha}$, where $\alpha \in_R [1, n-1]$.
3. Set $\delta = x_i \alpha \bmod n$ and select blinding values $r_\alpha, r_x, r_\delta \in_R [1, n-1]$.
4. Compute $R_1 = u^{r_\alpha}$, $\hat{R}_3 = \hat{T}_1^{r_x} \hat{u}^{-r_\delta}$ and

$$R_2 = e(T_2, g_2)^{r_x} e(v, w)^{-r_\alpha} e(v, g_2)^{-r_\delta}.$$

5. Compute $c = H(gpk, M, r, \hat{T}_1, T_2, R_1, R_2, \hat{R}_3)$.
6. Compute $s_\alpha = r_\alpha + c\alpha$, $s_x = r_x + c x_i$ and $s_\delta = r_\delta + c\delta \bmod n$.

The algorithm outputs the signature $\sigma = (r, \hat{T}_1, T_2, c, s_\alpha, s_x, s_\delta)$.

Verify$(gpk, \sigma, M,$ RL*):* Note that RL is the set of revocation tokens (each an element of $\mathbb{G}_1$). The algorithm proceeds in two phases.

1. **Signature Check:** Check that σ is a valid signature on M as follows.
   (a) Compute $(\hat{u}, v) = H_0(gpk, M, r)$, which also gives $u = \psi(\hat{u})$.
   (b) Rederive $R_1, R_2, \hat{R}_3$ as $\tilde{R}_1 = u^{s_\alpha} / T_1^c$ where $T_1 = \psi(\hat{T}_1)$, $\tilde{R}_3 = \hat{T}_1^{s_x} \hat{u}^{-s_\delta}$ and

$$\tilde{R}_2 = e(T_2, g_2)^{s_x} e(v, w)^{-s_\alpha} e(v, g_2)^{-s_\delta} \left(e(T_2, w)/e(g_1, g_2)\right)^c.$$

   (c) Check that the challenge c is correct:

$$c \stackrel{?}{=} H(gpk, M, r, \hat{T}_1, T_2, \tilde{R}_1, \tilde{R}_2, \tilde{R}_3).$$

   If it is accept, otherwise, reject.
2. **Revocation check:** For each element $A \in$ RL, regard A as encoded in $(\hat{T}_1, T_2)$ if

$$e(T_2/A, \hat{u}) = e(v, \hat{T}_1).$$

   If no element of RL is encoded in $(\hat{T}_1, T_2)$, the signer of σ has not been revoked.

The algorithm outputs valid if both phases accept, otherwise it outputs invalid.

A.2 Security

Security of a VLR group signature scheme requires that the protocol must satisfy three properties: correctness, selfless-anonymity and traceability. (See [7, 32] for the exact definition of security.) We discuss how each of these properties is maintained by the modified protocol as described in Appendix A.1.

Correctness. This property mandates that, for all $(gpk, \mathbf{gsk}, \mathbf{grt})$ generated by KeyGen algorithm, every signature generated by Sign algorithm should verify as valid, except when the signer is revoked. Formally,

$$\text{Verify}(gpk, \text{Sign}(gpk, \mathbf{gsk}[i], M), M, \mathsf{RL}) = \mathsf{valid} \iff \mathbf{grt}[i] \notin \mathsf{RL}.$$

In the modified protocol, a signature is accepted as valid in the first verification phase if the output of H equals c. A signature includes all the inputs to H except $R_1, R_2, \hat{R}_3$ which are rederived by the verifier as follows.

$$\tilde{R}_1 = u^{s_\alpha}/T_1^c = u^{r_\alpha + c\alpha}/u^{c\alpha} = u^{r_\alpha} = R_1$$

$$\tilde{R}_3 = \hat{T}_1^{s_x} \hat{u}^{-s_\delta} = \hat{T}_1^{r_x + cx_i} \hat{u}^{-(r_\delta + c\delta)} = \hat{R}_3 \hat{T}_1^{cx_i} \hat{u}^{c\delta} = \hat{R}_3$$

$$\begin{aligned}
\tilde{R}_2 &= e(T_2, g_2)^{s_x} e(v, w)^{-s_\alpha} e(v, g_2)^{-s_\delta} \left(e(T_2, w)/e(g_1, g_2)\right)^c \\
&= R_2 e(T_2, g_2)^{cx_i} e(v, w)^{-c\alpha} e(v, g_2)^{-cx_i\alpha} \left(e(T_2, w)/e(g_1, g_2)\right)^c \\
&= R_2 \left(\frac{e(T_2 v^{-\alpha}, w g_2^{x_i})}{e(g_1, g_2)}\right)^c \\
&= R_2 \left(\frac{e(A_i, w g_2^{x_i})}{e(g_1, g_2)}\right)^c \\
&= R_2.
\end{aligned}$$

So only a valid signature will be accepted in the first phase (except with a negligible probability of hash collision).

Now, a signature is rejected in the second phase when $e(T_2/A, \hat{u}) = e(v, \hat{T}_1)$ for some $A \in \mathsf{RL}$. This happens only if A is used in the generation of the signature component T_2.

Selfless-anonymity. The selfless-anonymity property is established through a game between a challenger and an adversary. In this game, the adversary's goal is to determine which of two users of her choosing generated a signature. She is not given access to the private key of any of these two users.

The selfless-anonymity property of the modified protocol can be established through a reductionist argument. In particular, given an adversary $\mathcal{A}$ that wins the selfless-anonymity game one can construct an algorithm $\mathcal{B}$ that solves the following variant of the Decision Linear problem in $E[n]$ and $\mathbb{G}_1$.

$\mathcal{B}$ is given a 6-tuple $(u_0, u_1, h_0 = u_0^a, h_1 = u_1^b, v_0, Z)$, where $u_0, u_1 \in_R E[n]$, $a, b \in_R [1, n-1]$, $v_0 \in_R \mathbb{G}_1$, and either $Z = v_0^{a+b}$ or $Z \in_R \mathbb{G}_1$. $\mathcal{B}$'s task is to decide whether $Z = v_0^{a+b}$.

Given this Decision Linear problem instance, $\mathcal{B}$ simulates the protocol environment for $\mathcal{A}$ to play the selfless-anonymity game.

Setup:

1. $\mathcal{B}$ picks $g_2 \in_R E[n]$ and sets $g_1 = \psi(g_2)$. $\mathcal{B}$ also picks $\gamma \in_R [1, n-1]$ and sets $w = g_2^{\gamma}$ and gives $\mathcal{A}$ the group public key $gpk = (g_1, g_2, w)$.
2. $\mathcal{B}$ selects two random users $i_0, i_1 \in [1, \ldots, N]$ and keeps i_0, i_1 secret. For all the other users it uses the standard key generation algorithm to generate the corresponding private key.

In the simulation $\mathcal{B}$ pretends as if the revocation token for i_0 is $A_{i_0} = Z/v_0^a$ and that for i_1 is $A_{i_1} = v_0^b$. Note that, $\mathcal{B}$ cannot compute A_{i_0} or A_{i_1} since it does not know v_0^a or v_0^b. Also note that $A_{i_0} = A_{i_1}$ when $Z = v_0^{a+b}$.

Random oracle queries: Each query to the random oracle H_0 or H is answered with random values from the appropriate set while maintaining consistency.

Phase 1: $\mathcal{A}$ can issue signing, corruption or revocation queries for any user i. If $i \neq i_0, i_1$ then $\mathcal{B}$ responds with an appropriate answer based on the corresponding private key $\mathbf{gsk}[i]$. A query corresponding to i_0 or i_1 is answered as follows.

- Signing query: Given a message $M \in \{0,1\}^*$ and user $i \in \{i_0, i_1\}$ $\mathcal{B}$ proceeds as follows.
  - To generate a signature on M under the private key of i_0:
    1. $\mathcal{B}$ picks random $s, t, l, c \in [1, n-1]$.
    2. Let $u_0 = (u_{0_1}, u_{0_2}) \in \mathbb{G}_1 \times \mathbb{T}_0$; then $h_0 = u_0^a = (u_{0_1}^a, u_{0_2}^a)$. $\mathcal{B}$ makes the following assignments: $\bar{u}_0 = (u_{0_1}, u_{0_2}^c)$, $\bar{h}_0 = (u_{0_1}^a, u_{0_2}^{ac}) = \bar{u}_0^a$.
    3. From $v_0, Z, \bar{u}_0 = (\bar{u}_{0_1}, \bar{u}_{0_2})$ and $\bar{h}_0 = (\bar{h}_{0_1}, \bar{h}_{0_2})$, $\mathcal{B}$ derives $\hat{u} = \bar{u}_0^l$, $\hat{T}_1 = \bar{h}_0 \bar{u}_0^s$, $v = (v_0 \bar{u}_{0_1}^t)^l$ and $T_2 = Z v_0^s \bar{h}_{0_1}^t \bar{u}_{0_1}^{st}$.

    Letting $\alpha = (a+s)/l$, we have $\hat{T}_1 = \hat{u}^{\alpha}$ and $T_2 = A_{i_0} v^{\alpha}$.
  - To generate a signature on M under the private key of i_1:
    1. $\mathcal{B}$ picks random $s, t, l, c \in [1, n-1]$.
    2. Let $u_1 = (u_{1_1}, u_{1_2}) \in \mathbb{G}_1 \times \mathbb{T}_0$; then $h_1 = u_1^b = (u_{1_1}^b, u_{1_2}^b)$. $\mathcal{B}$ makes the following assignments: $\bar{u}_1 = (u_{1_1}, u_{1_2}^c)$, $\bar{h}_1 = (u_{1_1}^a, u_{1_2}^{ac}) = \bar{u}_1^a$.
    3. From $v_0, \bar{u}_1 = (\bar{u}_{1_1}, \bar{u}_{1_2})$ and $\bar{h}_1 = (\bar{h}_{1_1}, \bar{h}_{1_2})$, $\mathcal{B}$ derives $\hat{u} = \bar{u}_1^l$, $\hat{T}_1 = \bar{h}_1 \bar{u}_1^s$, $v = (\bar{u}_{1_1}^t / v_0)^l$ and $T_2 = \bar{h}_{1_1}^t \bar{u}_{1_1}^{st} / v_0^s$.

    Letting $\alpha = (b+s)/l$, we have $\hat{T}_1 = \hat{u}^{\alpha}$ and $T_2 = A_{i_1} v^{\alpha}$.

  Thus in both cases we have $\hat{T}_1 = \hat{u}^{\alpha}$ and $T_2 = A_i v^{\alpha}$ for some $\alpha \in_R [1, n-1]$ and random and independent $\hat{u} \in E[n]$, $v \in \mathbb{G}_1$. $\mathcal{B}$ now picks random values $r, c, s_{\alpha}, s_x, s_{\delta} \in [1, n-1]$ and computes $R_1, R_2, \hat{R}_3$. If $\mathcal{A}$ has already queried random oracle H_0 with the input (gpk, M, r) or H with $(gpk, M, r, \hat{T}_1, T_2, R_1, R_2, \hat{R}_3)$ then $\mathcal{B}$ aborts the game with failure. Note that the probability of this is negligibly small as r is chosen at random from $[1, n-1]$. Otherwise, $\mathcal{B}$ makes the assignments $H_0(gpk, M, r) = (\hat{u}, v)$ and $H(gpk, M, r, \hat{T}_1, T_2, R_1, R_2, \hat{R}_3) = c$.
  $\mathcal{B}$ provides $\sigma = (r, \hat{T}_1, T_2, c, s_{\alpha}, s_x, s_{\delta})$ as the signature on M under the private key of user $i \in \{i_0, i_1\}$.
- Corruption and revocation queries: $\mathcal{B}$ aborts the game with failure if $\mathcal{A}$ issues a corruption or revocation query on i_0 or i_1.

Challenge: At this stage, $\mathcal{A}$ outputs a message $M \in \{0,1\}^*$ and two target users i_0^*, i_1^*. If $\{i_0^*, i_1^*\} \neq \{i_0, i_1\}$ then $\mathcal{B}$ aborts with failure. Otherwise, suppose $i_0^* = i_0$ and $i_1^* = i_1$. $\mathcal{B}$ picks a bit β uniformly at random and generates a signature σ^* under the private key of i_β^* using the signature generation method for i_0 or i_1 as illustrated above and gives it to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ makes additional queries as in Phase 1 with the restriction that it cannot ask for the corruption or revocation of i_0^* and i_1^*.

Output: Finally, $\mathcal{A}$ outputs its guess β' of β. $\mathcal{B}$ outputs 0 if $\beta = \beta'$ implying Z is a random element of $\mathbb{G}_1$. Otherwise, $\mathcal{B}$ outputs 1 implying $Z = v_0^{a+b}$.

For simplicity assume that $\mathcal{B}$ does not abort the game. We obtain a perfect simulation of the selfless-anonymity game when Z is a random element of $\mathbb{G}_1$. On the other hand, when $Z = v_0^{a+b}$ both i_0^* and i_1^* have the same private key and so the bit β is information theoretically hidden from $\mathcal{A}$. Hence any adversary $\mathcal{A}$ with an advantage ϵ in the selfless-anonymity game can be used to construct an algorithm $\mathcal{B}$ that solves the Decision Linear problem with advantage $\epsilon/2$.

Traceability. A VLR group signature scheme is traceable if no adversary can win the traceability game as defined in [7, 32]. Informally speaking, the adversary's goal in this game is to produce a valid nontrivial signature that cannot be traced to one of the users in her coalition (i.e., users corrupted by the adversary).

The traceability property is established in [7, 32] by an involved reductionist argument that proceeds through several stages. First an interaction framework is described for simulating the protocol environment for an adversary $\mathcal{A}$ that wins the traceability game. Next it is shown how one can instantiate this framework appropriately to obtain a solution to an instance of the underlying strong Diffie-Hellman (SDH) problem. We note that the original argument carries over for the modified protocol when the interaction framework is described to simulate the modified protocol. In particular, each query to the random oracle H_0 returns $\hat{u}, v \in E[n] \times \mathbb{G}_1$ and the signature component $T_1 \in \mathbb{G}_1$ in the original argument is replaced by $\hat{T}_1 \in E[n]$ in the simulation for the modified protocol.