COMPARING TWO PAIRING-BASED AGGREGATE SIGNATURE SCHEMES

SANJIT CHATTERJEE, DARREL HANKERSON, EDWARD KNAPP, AND ALFRED MENEZES

Abstract. In 2003, Boneh, Gentry, Lynn and Shacham (BGLS) devised the first provably-secure aggregate signature scheme. Their scheme uses bilinear pairings and their security proof is in the random oracle model. The first pairing-based aggregate signature scheme which has a security proof that does not make the random oracle assumption was proposed in 2006 by Lu, Ostrovsky, Sahai, Shacham and Waters (LOSSW). In this paper, we compare the security and efficiency of the BGLS and LOSSW schemes when asymmetric pairings derived from Barreto-Naehrig (BN) elliptic curves are employed.

1. Introduction

Beginning with the work of Joux [27] in 2000, bilinear pairings have been extensively used to design cryptographic protocols. Bilinear pairings come in two flavours – symmetric and asymmetric. If n is prime, and $\mathbb{G}$ and $\mathbb{G}_T$ are two groups of order n, then a *symmetric* pairing on $(\mathbb{G}, \mathbb{G}_T)$ is a function $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ that is bilinear, non-degenerate, and efficiently computable. Following [17], we will refer to these pairings as *Type 1* pairings. On the other hand, if $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are three groups of order n with $\mathbb{G}_1 \neq \mathbb{G}_2$, then an *asymmetric* pairing on $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ is a function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that is bilinear, non-degenerate, and efficiently computable. Following [17], asymmetric pairings for which an efficiently-computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ is known will be called *Type 2* pairings, while asymmetric pairings for which no efficiently-computable isomorphism is known either from $\mathbb{G}_1$ to $\mathbb{G}_2$ or from $\mathbb{G}_2$ to $\mathbb{G}_1$ are called *Type 3* pairings.

Boneh, Lynn and Shacham [9] were the first to observe that an efficiently-computable isomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$ can be essential to the security of a protocol. They showed that their short-signature scheme is insecure if implemented with a certain bilinear pairing for which the co-Diffie-Hellman problem (co-DHP; see §2.3 for the definition of co-DHP) is intractable but for which an isomorphism ψ is not known. Boneh, Lynn and Shacham commented that the requirement for the map ψ can be avoided by using a Type 3 pairing at "the cost of making a stronger complexity assumption", namely that co-DHP* is intractable (see §2.3 for the definition of co-DHP*). They conclude by saying that since ψ naturally exists in Type 2 pairings, there is no reason to rely on this stronger complexity assumption. It is perhaps because of the prevailing belief that hardness of co-DHP* is a stronger complexity assumption than hardness of co-DHP that most researchers who describe their protocols with asymmetric pairings use Type 2 pairings instead of Type 3 pairings.

In this paper we compare the security and efficiency of two provably-secure pairing-based signature schemes — those of Boneh, Gentry, Lynn and Shacham (BGLS) [7] and Lu, Ostrovsky, Sahai, Shacham and Waters (LOSSW) [29]. The BGLS scheme was originally described using the setting of a Type 2 pairing, and its security proof is in the random oracle model (ROM). The LOSSW scheme, on the other hand, was described in the setting of a Type 1 pairing, and its security proof did not make the random oracle assumption. The authors of [29] claimed that the LOSSW scheme "in some cases outperform[s] the most efficient random-oracle-based schemes", and in particular has faster signature verification than BGLS. The basis for the latter claim was that a BGLS signature verification requires $\ell+1$ pairing computations when ℓ signatures have been aggregated, whereas LOSSW requires only 2 pairing computations. However, the removal of the relatively-expensive pairing operations is at the expense of introducing other operations including multiplications,

Date: February 24, 2009; revised on August 15, 2009.

exponentiations, and group membership testing, and possibly requiring larger public keys and signatures. Our goal in this paper is a detailed comparison of BGLS and LOSSW using state-of-the-art BN pairings, and by using the known reductionist security arguments to guide the protocol specification and parameter selection.

We show that the BGLS and LOSSW schemes, as well as the BLS [9] and Waters [41] signature schemes upon which they are based, can all be described using Type 3 pairings (and that the Waters and LOSSW schemes can be described using Type 2 pairings).[1] We explain how some of these protocols have to be modified in order for the known reductionist security proofs to carry over to the different settings.[2] We argue, contrary to what can be inferred from [9], that the existing evidence suggests that Type 3 pairings offer at least as much security as Type 2 pairings when used to implement the four signature schemes under consideration. Furthermore, we compare Type 2 and Type 3 pairings derived from a certain Barreto-Naehrig (BN) elliptic curve [2] offering 128 bits of security. We show that the elements of the group $\mathbb{G}_2$ in Type 2 pairings can always be represented so that operations in $\mathbb{G}_2$ have significantly lower cost than suggested by the high-level estimates of Galbraith, Paterson and Smart [17] and the analysis of Chen, Cheng and Smart [12]. Despite these improvements, we conclude that Type 2 pairings offer no performance benefits over Type 3 pairings. Finally, we demonstrate that the BGLS scheme outperforms the LOSSW scheme in every respect — smaller public keys, smaller signatures, faster signature generation, and faster signature verification when 10 or fewer signatures have been aggregated.

The remainder of this paper is organized as follows. The construction of Type 2 and Type 3 pairings from BN curves are reviewed in §2, and their security is contrasted. In §3, we describe a particular BN curve and offer concrete estimates of the efficiency of fundamental operations in the Type 2 and Type 3 pairings derived from this curve. The security and efficiency of the BLS and Waters signature schemes are compared in §4. Finally, §5 compares the security and efficiency of the BGLS and LOSSW aggregate signature schemes.

Notation. In order to maintain consistency with the literature, $\mathbb{G}$, $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_2'$ will be additively-written cyclic groups with generators P, P_1, P_2, P_2' in §2 and §3, while in §4 and §5, $\mathbb{G}$, $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_2'$ will be multiplicatively-written cyclic groups with generators g, g_1, g_2, g_2'. The notation SIG-x is used to denote the signature scheme SIG implemented with a Type x pairing, where $\mathrm{x} \in \{1,2,3\}$.

2. Barreto-Naehrig curves

Barreto-Naehrig elliptic curves [2] are parameterized by an integer z for which both $p(z) = 36z^4 + 36z^3 + 24z^2 + 6z + 1$ and $n(z) = 36z^4 + 36z^3 + 18z^2 + 6z + 1$ are prime. For each such z, there is an ordinary elliptic curve $E : Y^2 = X^3 + b$ defined over the finite field $\mathbb{F}_p$ where $p = p(z)$, and such that $\#E(\mathbb{F}_p) = n$ where $n = n(z)$. Such an elliptic curve has embedding degree $k = 12$, this being the smallest positive integer for which n divides $p^k - 1$. If follows that $E[n] \subseteq E(\mathbb{F}_{p^{12}})$, where $E[n]$ denotes the set of all n-torsion points on E. (Recall that $E[n]$ is a finite abelian group of rank 2, whence $E[n] \cong \mathbb{Z}_n \oplus \mathbb{Z}_n$ and $E[n]$ has $n+1$ different subgroups of order n.)

2.1. Type 3 pairings from BN curves.

Let $\mathbb{G}_1 = E(\mathbb{F}_p)$, and let $\mathbb{G}_T$ denote the unique order-n subgroup of $\mathbb{F}_{p^{12}}^*$. Now, E has a sextic twist over $\mathbb{F}_{p^2}$, namely $\tilde{E}/\mathbb{F}_{p^2} : Y^2 = X^3 + b'$, such that $n \mid \#\tilde{E}(\mathbb{F}_{p^2})$ and $n^2 \nmid \#\tilde{E}(\mathbb{F}_{p^2})$ [24]. Let $\tilde{T} \in \tilde{E}(\mathbb{F}_{p^2})$ be a point of order n, and define $\tilde{\mathbb{G}}_2 = \langle \tilde{T} \rangle$. Then there is an efficiently-computable monomorphism $\phi : \tilde{E}(\mathbb{F}_{p^2}) \to E(\mathbb{F}_{p^{12}})$. Letting $T = \phi(\tilde{T})$ and $\mathbb{G}_2 = \langle T \rangle$, we have $\mathbb{G}_2 \neq \mathbb{G}_1$ and an efficiently-computable group isomorphism $\phi : \tilde{\mathbb{G}}_2 \to \mathbb{G}_2$. The group $\mathbb{G}_2$ is called the *trace-0 subgroup* of

[1] Type 1 pairings are currently viewed as being significantly slower than their Type 2 and Type 3 counterparts (see [22]) at the 128-bit security level, and therefore we will restrict our attention in this paper to Type 2 and Type 3 pairings.

[2] Smart and Vercauteren [40] analyzed the security of the BLS signature scheme in the Type 3 setting. Their security reduction makes a relativized assumption whereby the adversary is given oracle access to ψ. Our analysis avoids the use of such relativized assumptions.

$E[n]$ since it has the property that $\mathrm{Tr}(Z) = \sum_{i=0}^{11} \pi^i(Z) = \infty$ for all $Z \in \mathbb{G}_2$, where $\pi : (x,y) \mapsto (x^p, y^p)$ is the pth-power Frobenius map. We next define three asymmetric pairings on $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$. Since no efficiently-computable isomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$ or from $\mathbb{G}_2$ to $\mathbb{G}_1$ is known, these pairings are of Type 3.

The (*full*) *Tate pairing* $\hat{e} : E[n] \times E[n] \to \mathbb{G}_T$ can be defined as follows. Let $P, Q \in E[n]$, and let $R \in E(\mathbb{F}_{p^{12}})$ with $R \notin \{\infty, P, -Q, P-Q\}$. Then

$$\hat{e}(P,Q) = \left(\frac{f_{n,P}(Q+R)}{f_{n,P}(R)} \right)^{(p^{12}-1)/n},$$

where the *Miller function* $f_{s,P}$ is a function whose only zeros and poles in E are a zero of order s at P, a pole of order 1 at sP, and a pole of order $s-1$ at ∞. The (*restricted*) *Tate pairing* $t_n : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is defined by

$$t_n(P,Q) = (f_{n,P}(Q))^{(p^{12}-1)/n},$$

and can be computed using Algorithm 1.

Algorithm 1 (Computing the Tate pairing)

```
INPUT: $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$.
OUTPUT: $t_n(P,Q)$.
    1. Write $n$ in binary: $n = \sum_{i=0}^{L-1} n_i 2^i$.
    2. $T \leftarrow P$,   $f \leftarrow 1$.
    3. For $i$ from $L-2$ downto to 0 do:          {Miller operation}
        3.1 Let $\ell$ be the tangent line at $T$.
        3.2 $T \leftarrow 2T$,   $f \leftarrow f^2 \cdot \ell(Q)$.
        3.3 If $n_i = 1$ and $i \neq 0$ then
                Let $\ell$ be the line through $T$ and $P$.
                $T \leftarrow T + P$,   $f \leftarrow f \cdot \ell(Q)$.
    4. Return($f^{(p^{12}-1)/n}$).          {Final exponentiation}
```

The *ate pairing* $a_n : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, introduced by Hess, Smart and Vercauteren [24], is defined by

$$a_n(P,Q) = (f_{t-1,Q}(P))^{(p^{12}-1)/n}$$

where $t - 1 = p - \#E(\mathbb{F}_p) = 6z^2$. The ate pairing is generally faster to compute than the Tate pairing because the number of iterations in the Miller operation is determined by the bitlength of $t - 1 \approx \sqrt{n}$.

The *R-ate pairing* $R_n : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, introduced recently by Lee, Lee and Park [28], further decreases the number of iterations of the Miller operation. It is defined by

$$R_n(P,Q) = \left(f \cdot (f \cdot \ell_{aQ,Q}(P))^p \cdot \ell_{\pi(aQ+Q),aQ}(P)\right)^{(p^{12}-1)/n},$$

where $a = 6z + 2$, $f = f_{a,Q}(P)$, and $\ell_{A,B}$ denotes the line through A and B. There is an integer N such that $R_n(P,Q) = \hat{e}(Q,P)^N$ for all $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$ [28]. The R-ate pairing can be computed using Algorithm 2. Notice that the number of iterations in the Miller operation is now determined by the bitlength of $a \approx \sqrt{t} \approx n^{1/4}$.

Algorithm 2 (Computing the R-ate pairing)

```
INPUT: $P \in G_1$ and $Q \in G_2$.
OUTPUT: $R_r(Q,P)$.
    1. Write $a = 6z + 2$ in binary: $a = \sum_{i=0}^{L-1} a_i 2^i$.
```

```
2. $T \leftarrow Q$,    $f \leftarrow 1$.
3. For $i$ from $L-2$ downto 0 do:          {Miller operation}
   3.1 Let $\ell$ be the tangent line at $T$.
   3.2 $T \leftarrow 2T$,    $f \leftarrow f^2 \cdot \ell(P)$.
   3.3 If $a_i = 1$ then
           Let $\ell$ be the line through $T$ and $Q$.
           $T \leftarrow T + Q$,    $f \leftarrow f \cdot \ell(P)$.
4. $f \leftarrow f \cdot (f \cdot \ell_{T,Q}(P))^p \cdot \ell_{\pi(T+Q),T}(P)$.
5. Return($f^{(p^{12}-1)/n}$).          {Final exponentiation}
```

Table 1 from [22] lists the costs of computing the Tate, ate, and R-ate pairings for a particular BN curve described in §3, demonstrating the superiority of the R-ate pairing. The cost estimates have been validated by experiments. For example, [22] reports timings of 81 million and 54 million clock cycles for computing the ate and R-ate pairings on a 2.8 GHz Pentium 4 machine using general purpose registers.

Pairing	Miller operation	Final exponentiation	Total
Tate	27,934m	7,246m+i	35,180m+i
ate	15,801m	7,246m+i	23,047m+i
R-ate	7,847m+i	7,246m+i	15,093m+2i

Table 1. Costs of the Tate, ate and R-ate pairings for the BN curve described in §3. Here, m and i denote multiplication and inversion in $\mathbb{F}_p$.

If the product of ℓ R-ate pairings is desired, then the steps of the individual pairing computations can be interleaved, with the product of the partial results being stored in a common accumulator f [38, 21]. In that case, the expensive operation $f \leftarrow f^2$ in step 3.2 of Algorithm 2 and the final exponentiation in step 5 can be shared by all ℓ pairing computations. It follows that the cost of computing the product of ℓ R-ate pairings for the particular BN curve is $15,093m + 2i + (\ell-1)(5507m + i)$; i.e., the incremental cost of each additional pairing is roughly one-third the cost of the first pairing.

2.2. **Type 2 pairings from BN curves.** Let $R \in E[n]$ with $R \notin \mathbb{G}_1$ and $R \notin \mathbb{G}_2$, and define $\mathbb{G}_2' = \langle R \rangle$. Then the map $e_n : \mathbb{G}_1 \times \mathbb{G}_2' \to \mathbb{G}_T$ defined by $e_n(P,Q) = \hat{e}(Q,P)^{2N}$ is an asymmetric pairing on $(\mathbb{G}_1, \mathbb{G}_2', \mathbb{G}_T)$. It is a Type 2 pairing because the trace map Tr is an efficiently-computable isomorphism from $\mathbb{G}_2'$ to $\mathbb{G}_1$.

At first glance, it may appear that evaluating $e_n(P,Q)$ may be more expensive than evaluating the Type 3 pairing R_n because the point $Q \in \mathbb{G}_2'$ has coordinates that are in $\mathbb{F}_{p^{12}}$ and not in any proper subfield. However, the following shows that the task of computing $e_n(P,Q)$ is easily reduced to the task of computing an R-ate pairing value.

Lemma 1 ([25]). *Let $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2'$. Then $e_n(P,Q) = R_n(P,\hat{Q})$, where $\hat{Q} = Q - \pi^6(Q)$.*

Proof. First note that $\hat{Q} \neq \infty$ since $Q \notin E(\mathbb{F}_{p^6})$. Moreover, $\mathrm{Tr}(\hat{Q}) = \mathrm{Tr}(Q) - \mathrm{Tr}(\pi^6(Q)) = \infty$, and hence $\hat{Q} \in \mathbb{G}_2$. Finally,

$$\begin{aligned} e_n(P,Q) &= \hat{e}(Q,P)^{2N} \\ &= \hat{e}(2Q,P)^N \\ &= \hat{e}(Q+\hat{Q}+\pi^6(Q),P)^N \\ &= \hat{e}(\hat{Q},P)^N \cdot \hat{e}(Q+\pi^6(Q),P)^N \\ &= R_n(P,\hat{Q}), \end{aligned}$$

since $Q + \pi^6(Q) \in E(\mathbb{F}_{p^6})$ whence $\hat{e}(Q+\pi^6(Q),P) = 1$ [15, Lemma IX.8]. □

2.3. **Security.** Recall that $\mathbb{G}_1 = E(\mathbb{F}_p)$, $\mathbb{G}_2$ is the trace-0 subgroup of $E[n]$, $\mathbb{G}_2'$ is any order-n subgroup of $E[n]$ different from $\mathbb{G}_1$ and $\mathbb{G}_2$, and $\mathbb{G}_T$ is the order-n subgroup of $\mathbb{F}_{p^{12}}^*$. BN curves are especially well suited for the 128-bit security level because if p is a 256-bit prime (whence n is also a 256-bit prime) then Pollard's rho method [34] for computing discrete logarithms in $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_2'$ or $\mathbb{G}_T$ has running time at least 2^{128}, as does the number field sieve algorithm for computing discrete logarithms in the extension field $\mathbb{F}_{p^{12}}$ [19, 35, 36]. Schirokauer [37] has shown that there are cases where discrete logarithms in prime fields $\mathbb{F}_p$ and degree-two extensions $\mathbb{F}_{p^2}$ of prime fields can be computed significantly faster than standard versions of the number field sieve if the prime p has low Hamming weight. However, Schirokauer's method is not known to be effective for computing discrete logarithms in $\mathbb{F}_{p^{12}}$ for BN primes p, even if the BN parameter z is chosen so that p has relatively small Hamming weight. Thus, the fastest algorithms presently known for computing discrete logarithms in $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_2'$ and $\mathbb{G}_T$ have running times at least 2^{128}.

It should be noted that while the discrete logarithm problems in the groups $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_2'$ and $\mathbb{G}_T$ are equivalent in practice, such an equivalence has never been *proven*. Let us denote the discrete logarithm problem in a group $\mathbb{G}$ by $\mathrm{DLP}_{\mathbb{G}}$. The Tate pairing can be used to efficiently reduce the DLP in any order-n subgroup of $E[n]$ to the DLP in $\mathbb{G}_T$. Namely, if $P \in E[n]$ and $Q \in \langle P \rangle$, then one has $\log_P Q = \log_\alpha \beta$ where $\alpha = \hat{e}(P,R)$, $\beta = \hat{e}(Q,R)$, and $R \in E[n]$ is chosen so that $\hat{e}(P,R) \neq 1$. Now, the trace map $\mathrm{Tr} : \mathbb{G}_2' \to \mathbb{G}_1$ is an efficiently-computable isomorphism and so $\mathrm{DLP}_{\mathbb{G}_2'} \leq \mathrm{DLP}_{\mathbb{G}_1}$. Furthermore, if $Q \in \mathbb{G}_2'$ then $Q_2 = Q - \frac{1}{12}\mathrm{Tr}(Q) \in \mathbb{G}_2$ since $Q_2 \in E[n]$ and $\mathrm{Tr}(Q_2) = \mathrm{Tr}(Q) - \mathrm{Tr}(Q) = \infty$. Hence the map

$$\rho : \mathbb{G}_2' \to \mathbb{G}_2, \qquad Q \mapsto Q - \frac{1}{12}\mathrm{Tr}(Q) \tag{1}$$

is an efficiently-computable isomorphism, whence $\mathrm{DLP}_{\mathbb{G}_2'} \leq \mathrm{DLP}_{\mathbb{G}_2}$. Figure 1 summarizes the known relationships between the DLP in $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_2'$ and $\mathbb{G}_T$.

$$\begin{tikzcd}
 & & & \mathrm{DLP}_{\mathbb{G}_1} \arrow[dlll] \arrow[dl] & \\
\text{co-DHP*} \arrow[r, leftrightarrow] & \text{co-DHP} & \mathrm{DLP}_{\mathbb{G}_2'} \arrow[l] & & \mathrm{DLP}_{\mathbb{G}_T} \arrow[ul] \arrow[dl] \\
 & & & \mathrm{DLP}_{\mathbb{G}_2} \arrow[ulll] \arrow[ul] &
\end{tikzcd}$$

FIGURE 1. Relative difficulty of the discrete logarithm problems in $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_2'$, $\mathbb{G}_T$, and the co-DHP and co-DHP* problems. The notation $A \leftarrow B$ means that there is an efficient reduction from A to B. The equivalence of co-DHP* and co-DHP holds if the generators P_1, P_2, P_2' are suitably chosen (cf. Lemma 2).

Suppose now that P_1, P_2, P_2' are fixed generators of $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_2'$, respectively. Security of the signature schemes considered in §4 and §5 is based on a variant of the Diffie-Hellman problem (DHP). (Recall that DHP in a group $\mathbb{G} = \langle P \rangle$ is the problem of determining xyP given xP and yP.) If a Type 2 pairing is employed, then security of the signature schemes is based on the hardness of *co-DHP*: Given $Q \in \mathbb{G}_1$ and $zP_2' \in \mathbb{G}_2'$, compute zQ. The intractability of $\mathrm{DLP}_{\mathbb{G}_2'}$ is a necessary condition for hardness of co-DHP. On the other hand, if a Type 3 pairing is employed, then security is based on the hardness of *co-DHP**: Given Q, $zP_1 \in \mathbb{G}_1$ and $zP_2 \in \mathbb{G}_2$, compute zQ. The intractability of $\mathrm{DLP}_{\mathbb{G}_1}$ and $\mathrm{DLP}_{\mathbb{G}_2}$ are both necessary for the hardness of co-DHP*.

Note that our definition of co-DHP is with respect to *fixed* generators of $\mathbb{G}_1$ and $\mathbb{G}_2'$, while our definition of co-DHP* is with respect to *fixed* generators of $\mathbb{G}_1$ and $\mathbb{G}_2$. This is appropriate because these generators are regarded as fixed, public parameters in the signatures schemes under consideration. Moreover, unlike the case of the DLP and DHP in a prime-order group, it has not been proven that the hardness of co-DHP and co-DHP* is independent of the choice of generators.

It is not known whether $\text{DLP}_{\mathbb{G}_1} \le$ co-DHP*, or $\text{DLP}_{\mathbb{G}_2} \le$ co-DHP*, or $\text{DLP}_{\mathbb{G}_2'} \le$ co-DHP. This is unlike the case of the DHP in an order-n group $\mathbb{G}$, where there is substantial evidence that $\text{DLP}_{\mathbb{G}} \le \text{DHP}_{\mathbb{G}}$. For example, den Boer [5] proved that $\text{DLP}_{\mathbb{G}} \le \text{DHP}_{\mathbb{G}}$ if the group order n has the property that $n-1$ is smooth. Furthermore, den Boer's result was extended by Boneh, Lipton and Maurer [8, 30] to the case where an elliptic curve over $\mathbb{Z}_n$ of smooth order is known. Consequently, concerns that $\text{DHP}_{\mathbb{G}}$ might be easier than $\text{DLP}_{\mathbb{G}}$ can be alleviated by selecting an order-n group $\mathbb{G}$ for which the appropriate elliptic curves over $\mathbb{Z}_n$ are known [31]. The techniques of den Boer, Boneh, Lipton and Maurer do not appear to extend to the case of co-DHP and co-DHP*, and consequently there is presently no evidence (in the form of a reduction) that these problems are equivalent to the DLP in $\mathbb{G}_1$, $\mathbb{G}_2$ or $\mathbb{G}_2'$.

The following shows that co-DHP and co-DHP* are equivalent if the generators P_1, P_2, P_2' are suitably chosen.

Lemma 2. *Let $P_2' \in E[n]$ be an arbitrary point with $P_2' \notin \mathbb{G}_1$ and $P_2' \notin \mathbb{G}_2$, and let $\mathbb{G}_2' = \langle P_2' \rangle$. Let $c \in [1, n-1]$ be an arbitrary integer, and define $P_2 = c^{-1}\rho(P_2')$ and $P_1 = \frac{1}{12}\operatorname{Tr}(P_2')$. If c is known, then co-DHP $\le$ co-DHP* and co-DHP* $\le$ co-DHP.*

Proof. Note that $P_2' = P_1 + cP_2$. It can easily be checked that $P_1 \in \mathbb{G}_1 \setminus \{\infty\}$ and $P_2 \in \mathbb{G}_2 \setminus \{\infty\}$.

Now, given a co-DHP instance (Q, zP_2'), we compute $c^{-1}\rho(zP_2') = zP_2$ and $zP_2' - czP_2 = zP_1$. A co-DHP* solver is then used to find the solution zQ of the co-DHP* instance (Q, zP_1, zP_2), thus also obtaining the solution to the original co-DHP instance. This shows that co-DHP $\le$ co-DHP*.

Conversely, given a co-DHP* instance (Q, zP_1, zP_2), we compute $zP_1 + czP_2 = zP_2'$. A co-DHP solver is then used to find the solution zQ of the co-DHP instance (Q, zP_2'), thus also obtaining the solution to the original co-DHP* instance. This shows that co-DHP* $\le$ co-DHP. □

It is not known whether knowledge of the integer c makes co-DHP or co-DHP* any easier. Suppose instead that P_2 and P_2' were chosen independently at random from $\mathbb{G}_2$ and $E[n] \setminus (\mathbb{G}_1 \cup \mathbb{G}_2)$, respectively, and $P_1 = \frac{1}{12}\operatorname{Tr}(P_2')$. In this scenario, the integer c satisfying $\rho(P_2') = cP_2$ is not known, and we do not know efficient reductions from co-DHP to co-DHP* or from co-DHP* to co-DHP. The fact that efficient reductions $\text{DLP}_{\mathbb{G}_2'} \le \text{DLP}_{\mathbb{G}_1}$ and $\text{DLP}_{\mathbb{G}_2'} \le \text{DLP}_{\mathbb{G}_2}$ are known, while efficient reductions $\text{DLP}_{\mathbb{G}_1} \le \text{DLP}_{\mathbb{G}_2'}$ and $\text{DLP}_{\mathbb{G}_2} \le \text{DLP}_{\mathbb{G}_2'}$ are not known, might cause some people to have more confidence in the hardness of $\text{DLP}_{\mathbb{G}_1}$ and $\text{DLP}_{\mathbb{G}_2}$ than of $\text{DLP}_{\mathbb{G}_2'}$, and consequently more confidence in hardness of co-DHP* than of co-DHP. One should also note that the Decisional DHP is easy in $\mathbb{G}_2'$, but not known to be easy in $\mathbb{G}_1$ or in $\mathbb{G}_2$ [6]. Thus, the existing evidence does not indicate any weakness in Type 3 pairings relative to Type 2 pairings, but rather that Type 3 pairings are at least as secure as Type 2 pairings.

2.4. **Representation of $\mathbb{G}_2'$.** In the remainder of the paper, we shall assume that $\mathbb{G}_2' = \langle P_2' \rangle$ where the point P_2' was selected at random from $E[n] \setminus (\mathbb{G}_1 \cup \mathbb{G}_2)$. Furthermore, we choose $P_1 = \frac{1}{12}\operatorname{Tr}(P_2')$, whence

$$\psi : \mathbb{G}_2' \to \mathbb{G}_1, \qquad Q \mapsto \frac{1}{12}\operatorname{Tr}(Q) \tag{2}$$

is an efficiently-computable isomorphism such that $\psi(P_2') = P_1$. We either set $P_2 = c^{-1}\rho(P_2')$ for an arbitrary integer $c \in [1, n-1]$ (in which case co-DHP and co-DHP* are provably equivalent), or set P_2 to be a randomly selected trace-0 point (in which case it is not known how to prove the equivalence of co-DHP and co-DHP*).

Given $Q \in \mathbb{G}_2'$, one can efficiently determine the unique $Q_1 \in \mathbb{G}_1$ and $Q_2 \in \mathbb{G}_2$ such that $Q = Q_1 + Q_2$; namely, $Q_1 = \psi(Q)$ and $Q_2 = \rho(Q) = Q - Q_1$. Writing

$$D(Q) = (\psi(Q), \rho(Q)), \tag{3}$$

and letting $\mathbb{H}_2' \subseteq \mathbb{G}_1 \times \mathbb{G}_2$ denote the range of D, we have an efficiently-computable isomorphism $D : \mathbb{G}_2' \to \mathbb{H}_2'$ whose inverse is also efficiently computable. Note that addition in $\mathbb{H}_2'$ is component-wise.

We note that Smart and Vercauteren [40] had previously proposed defining a pairing where the group $\mathbb{G}_2'$ was taken to be an order-n subgroup of $\mathbb{G}_1 \times \mathbb{G}_2$. Our observation is that such a representation is without loss of generality, i.e., it can be used for *all* order-n subgroups $\mathbb{G}_2'$ of $E[n]$ different from $\mathbb{G}_1$ and $\mathbb{G}_2$.

3. A particular BN curve

For the remainder of this paper, we will work with the BN curve

$$E/\mathbb{F}_p : Y^2 = X^3 + 3$$

with BN parameter $z = 6000000000001\mathrm{F2D}$ (in hexadecimal) [14]. For this choice of BN parameter, p is a 256-bit prime of Hamming weight 87, $n = \#E(\mathbb{F}_p)$ is a 256-bit prime of Hamming weight 91, $t - 1 = 6z^2$ is a 128-bit integer of Hamming weight 28, and the R-ate parameter $a = 6z + 2$ is a 66-bit integer of Hamming weight 9. Note that $p \equiv 7 \pmod 8$ (whence -2 is a nonsquare modulo p) and $p \equiv 1 \pmod 6$.

3.1. Field representation.

The extension field $\mathbb{F}_{p^{12}}$ is represented using tower extensions

$$\begin{aligned} \mathbb{F}_{p^2} &= \mathbb{F}_p[u]/(u^2 + 2), \\ \mathbb{F}_{p^6} &= \mathbb{F}_{p^2}[v]/(v^3 - \xi) \text{ where } \xi = -u - 1, \text{ and} \\ \mathbb{F}_{p^{12}} &= \mathbb{F}_{p^6}[w]/(w^2 - v). \end{aligned}$$

We also have the representation

$$\mathbb{F}_{p^{12}} = \mathbb{F}_{p^2}[W]/(W^6 - \xi) \text{ where } W = w.$$

Hence an element $\alpha \in \mathbb{F}_{p^{12}}$ can be represented in any of the following three ways:

$$\begin{aligned} \alpha &= a_0 + a_1 w \quad \text{where } a_0, a_1 \in \mathbb{F}_{p^6} \\ &= (a_{0,0} + a_{0,1}v + a_{0,2}v^2) + (a_{1,0} + a_{1,1}v + a_{1,2}v^2)w \quad \text{where } a_{i,j} \in \mathbb{F}_{p^2} \\ &= a_{0,0} + a_{1,0}W + a_{0,1}W^2 + a_{1,1}W^3 + a_{0,2}W^4 + a_{1,2}W^5. \end{aligned}$$

We let (m, s, i), $(\tilde{m}, \tilde{s}, \tilde{\imath})$, (M, S, I) denote the cost of multiplication, squaring, inversion in $\mathbb{F}_p$, $\mathbb{F}_{p^2}$, $\mathbb{F}_{p^{12}}$, respectively. Experimentally, we have $s \approx 0.9m$ and $i \approx 41m$ on a Pentium 4 processor [22]. In our cost estimates that follow, we will make the simplifying assumption $s \approx m$.

3.1.1. *Arithmetic in* $\mathbb{F}_{p^2}$. We have $\tilde{m} \approx 3m$ using Karatsuba's method which reduces a multiplication in a quadratic extension to 3 (rather than 4) small field multiplications; $\tilde{s} \approx 2m$ using the complex method: $(a + bu)^2 = (a - b)(a + 2b) - ab + (2ab)u$; and $\tilde{\imath} \approx i + 2m + 2s$ since $(a + bu)^{-1} = (a - bu)/(a^2 + 2b^2)$. Note also that p-th powering is free in $\mathbb{F}_{p^2}$ because $(a + bu)^p = a - bu$.

3.1.2. *Arithmetic in* $\mathbb{F}_{p^6}$. Karatsuba's method reduces a multiplication in a cubic extension to 6 (rather than 9) multiplications in the smaller field. Hence a multiplication in $\mathbb{F}_{p^6}$ costs $18m$. Squaring in $\mathbb{F}_{p^6}$ costs $2\tilde{m} + 3\tilde{s} = 12m$ via the following formulae [13]: if $\beta = b_0 + b_1 v + b_2 v^2 \in \mathbb{F}_{p^6}$ where $b_i \in \mathbb{F}_{p^2}$, then $\beta^2 = (A + D\xi) + (B + E\xi)v + (B + C + D - A - E)v^2$ where $A = b_0^2$, $B = 2b_0 b_1$, $C = (b_0 - b_1 + b_2)^2$, $D = 2b_1 b_2$, and $E = b_2^2$. Finally, as shown in [39, Section 3.2], inversion in $\mathbb{F}_{p^6}$ can be reduced to 1 inversion, 9 multiplications, and 3 squarings in $\mathbb{F}_{p^2}$.

3.1.3. *Arithmetic in* $\mathbb{F}_{p^{12}}$. Since $\mathbb{F}_{p^{12}}$ is a tower of quadratic, cubic, and quadratic extensions, Karatsuba's method gives $M \approx 54m$. By using the complex method for squaring in $\mathbb{F}_{p^{12}}$ and Karatsuba for multiplication in $\mathbb{F}_{p^6}$ and $\mathbb{F}_{p^2}$, we have $S \approx 36m$. Since inversion in $\mathbb{F}_{p^{12}}$ can be reduced to 1 inversion, 2 multiplications, and 2 squarings in $\mathbb{F}_{p^6}$, it follows that $I \approx i + 97m$.

3.1.4. *Sextic twist.* The sextic twist $\tilde{E}$ of E over $\mathbb{F}_{p^2}$ for which $n \mid \#\tilde{E}(\mathbb{F}_{p^2})$ is

$$\tilde{E}/\mathbb{F}_{p^2} : Y^2 = X^3 + 3/\xi.$$

The monomorphism $\phi : \tilde{E}(\mathbb{F}_{p^2}) \to E(\mathbb{F}_{p^{12}})$ is given by $(x, y) \mapsto (xW^2, yW^3)$. Thus the group isomorphism $\phi : \tilde{\mathbb{G}}_2 \to \mathbb{G}_2$ as well as its inverse can be computed at no cost.

3.2. **Elliptic curve operations.** A point (X, Y, Z) in jacobian coordinates corresponds to the point (x, y) in affine coordinates with $x = X/Z^2$ and $y = Y/Z^3$. The formulas for doubling a point in $E(\mathbb{F}_{p^d})$ represented in jacobian coordinates require 3 multiplications and 4 squarings in $\mathbb{F}_{p^d}$, while the formulas for mixed jacobian-affine addition in $E(\mathbb{F}_{p^d})$ require 8 multiplications and 3 squarings in $\mathbb{F}_{p^d}$.

3.3. **Type 2 versus Type 3 pairings.** Table 2 lists the bitlengths of elements in $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}'_2$ and $\mathbb{G}_T$, and the estimated costs of performing essential operations in these groups. Our estimates are consistent with the high-level comparisons in Table 3 of [17] and Table 5 of [12], with the notable exception that the representation we use for $\mathbb{G}'_2$ results in elements of significantly smaller bitlengths and significantly faster exponentiation.[3]

	Type 2	Type 3
Bitlength of elements in $\mathbb{G}_1$	257	257
Bitlength of elements in $\mathbb{G}'_2/\mathbb{G}_2$	770	513
Bitlength of elements in $\mathbb{G}_T$	1,024	1,024
Compressing elements in $\mathbb{G}_1$	free	free
Compressing elements in $\mathbb{G}'_2/\mathbb{G}_2$	free	free
Decompressing elements in $\mathbb{G}_1$	$315m$	$315m$
Decompressing elements in $\mathbb{G}'_2/\mathbb{G}_2$	$989m$	$674m$
Addition in $\mathbb{G}_1$	$11m$	$11m$
Doubling in $\mathbb{G}_1$	$7m$	$7m$
Addition in $\mathbb{G}'_2/\mathbb{G}_2$	$41m$	$30m$
Doubling in $\mathbb{G}'_2/\mathbb{G}_2$	$24m$	$17m$
Exponentiation in $\mathbb{G}_1$	1,533m	1,533m
Exponentiation in $\mathbb{G}'_2/\mathbb{G}_2$	4,585m	3,052m
Fixed-base exponentiation in $\mathbb{G}_1$	$718m$	$718m$
Fixed-base exponentiation in $\mathbb{G}'_2/\mathbb{G}_2$	2,624m	1,906m
Hashing into $\mathbb{G}_1$	$315m$	$315m$
Hashing into $\mathbb{G}'_2/\mathbb{G}_2$	—	3,726m
e_n/R_n Pairing	15,175m	15,175m
Testing membership in $\mathbb{G}_1$	free	free
Testing membership in $\mathbb{G}'_2/\mathbb{G}_2$	23,775m	3,052m

TABLE 2. Bitlengths of elements in $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}'_2$ and $\mathbb{G}_T$, and estimated costs (in terms of $\mathbb{F}_p$ multiplications) of basic operations.

[3] For example, Table 5 of [12] estimates the cost exponentiation in $\mathbb{G}'_2$ as 45 times the cost of exponentiation in $\mathbb{G}_1$; our ratio is 3.

3.3.1. *Representing elements in $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_2'$ and $\mathbb{G}_T$.* A point $Q = (x, y) \in \mathbb{G}_1$ can be represented in compressed form by $x \in \mathbb{F}_p$ plus a sign bit of $y \in \mathbb{F}_p$. The full y-coordinate can be recovered by solving $y^2 = x^3 + 3$ over $\mathbb{F}_p$ via $y = \pm\sqrt{x^3+3} = (x^3+3)^{(p+1)/4}$. The exponentiation can be performed using sliding windows of width 5, at a cost of $315m$.

A point $Q \in \mathbb{G}_2$ has preimage $(x, y) \in \tilde{\mathbb{G}}_2$ under ϕ^{-1} and thus can be represented in compressed form by $x \in \mathbb{F}_{p^2}$ plus a sign bit of $y \in \mathbb{F}_{p^2}$. The full y-coordinate $y = \pm\sqrt{x^3 + 3/\xi}$ can be recovered at a cost of 2 square roots in $\mathbb{F}_p$ plus $i + m + 2s$ using Scott's method for computing square roots in $\mathbb{F}_{p^2}$ [39]. The overall cost is $674m$.

As noted in §2.4, a point $Q = (x, y) \in \mathbb{G}_2'$ can be represented by the pair of points $D(Q) = (Q_1, Q_2) \in \mathbb{H}_2'$, where $Q_1 = \frac{1}{12}\operatorname{Tr}(Q)$ and $Q_2 = Q - Q_1$. Now, $\pi^i(Q)$ for $1 \le i \le 5$ can be computed by successive application of π, each of which costs $30m$ since p-th powering of an element in $\mathbb{F}_{p^{12}}$ costs $15m$ [22]. And, since p^6-th powering in $\mathbb{F}_{p^{12}}$ is free, $\pi^i(Q)$ for $6 \le i \le 11$ can thereafter be obtained for free. Then computing $\operatorname{Tr}(Q) = \sum_{i=0}^{11} \pi^i(Q)$ takes 11 additions in $E(\mathbb{F}_{p^{12}})$, after which computing $\frac{1}{12}\operatorname{Tr}(Q)$ requires one exponentiation in $\mathbb{G}_1$. Finally, determining $Q - Q_1$ requires one addition in $E(\mathbb{F}_{p^{12}})$. The overall cost of deriving the representation (Q_1, Q_2) from Q is $8{,}163m$. Recovering Q from (Q_1, Q_2) requires an addition in $E(\mathbb{F}_{p^{12}})$ at a cost of (at most) $540m$. We will henceforth identify $\mathbb{G}_2'$ and $\mathbb{H}_2'$ and assume that points in $\mathbb{G}_2'$ are represented as (Q_1, Q_2). Such points can be compressed by compressing Q_1 and Q_2, while the decompression cost is $989m$. Point addition and doubling can be done component-wise, at a cost of $41m$ and $24m$, respectively.

Lastly, elements in $\mathbb{G}_T$ can be compressed from 3072 bits to 1024 bits using the techniques described in [20]. We will ignore the $\mathbb{G}_T$ compression and decompression costs in our performance analysis of signature schemes as they are a negligible portion of overall costs.

3.3.2. *Exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_2'$.* Computing kQ (also known as point multiplication), where k is an integer and Q is an elliptic curve point, can be performed using the w-NAF method. The expected cost of this method with ℓ-bit exponents k is

$$1D + (2^{w-2} - 1)A + \frac{\ell}{w+1}A + \ell D,$$

where D is the cost of doubling an elliptic curve point and A is the cost of adding two elliptic curve points (see [23, Algorithm 3.36]). However, faster exponentiation can be achieved using the Gallant-Lambert-Vanstone (GLV) strategy [18].

Let $\beta \in \mathbb{F}_p$ be an element of order 3 in $\mathbb{F}_p$. Then $\lambda : (x, y) \mapsto (\beta x, y)$ is an efficiently-computable endomorphism of E defined over $\mathbb{F}_p$. The GLV strategy computes kQ as $k_1Q + k_2\lambda(Q)$ where k_1 and k_2 are half-length exponents. If interleaving is used to compute k_1Q and $k_2\lambda(Q)$, and if k_1 and k_2 are each represented in width-w NAF, then the expected cost of exponentiation in $\mathbb{G}_1$ is approximately

$$2\left(1D + (2^{w-2} - 1)A\right) + \frac{\ell}{w+1}A + \frac{\ell}{2}D.$$

Taking $w = 5$ yields a cost of $1{,}533m$.

Galbraith and Scott [16] presented a 4-dimensional GLV strategy for exponentiation in $\mathbb{G}_2$. If interleaving is used to compute the four exponentiations, and the quarter-length exponents are represented in width-w NAF, then the expected cost of exponentiation in $\mathbb{G}_2$ is approximately

$$4\left(1D + (2^{w-2} - 1)A\right) + \frac{\ell}{w+1}A + \frac{\ell}{4}D.$$

Taking $w = 4$ yields a cost of $3{,}052m$.

Lastly, exponentiation of a point (Q_1, Q_2) in $\mathbb{G}_2'$ can be performed as (kQ_1, kQ_2); the cost is $4{,}585m$.

If the point Q is fixed or known in advance, then the operation kQ can be significantly accelerated by precomputing some multiples of Q. For example, the fixed-base comb method with two tables (see [23, Algorithm 3.45]) with windows of width w has expected cost approximately

$$\left(\frac{2^w-1}{2^w}2e-1\right)A+(e-1)D,$$

where $d=\lceil \ell/w\rceil$ and $e=\lceil d/2\rceil$. Taking $w=5$ yields the expected costs $718m$ and $1,906m$ for fixed-base exponentiation in $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. As before, fixed-base exponentiation in $\mathbb{G}_2'$ can be performed as (kQ_1,kQ_2); the cost is 2,624m.

3.3.3. *Hashing into* $\mathbb{G}_1$*,* $\mathbb{G}_2$ *and* $\mathbb{G}_2'$*.* Hashing into $\mathbb{G}_1$ can be defined by first using a standard hash function (such as SHA-2) to hash to an x-coordinate of $E(\mathbb{F}_p)$. A y-coordinate can then be computed as $\sqrt{x^3+3}$. The dominant cost is for computing the square root in $\mathbb{F}_p$, yielding our cost estimate of $315m$ for hashing into $\mathbb{G}_1$.

Hashing into $\mathbb{G}_2$ can be defined by first using a standard hash function to hash to an x-coordinate of $\tilde{E}(\mathbb{F}_{p^2})$, then computing a y-coordinate as $\sqrt{x^3+b/\xi}\in\mathbb{F}_{p^2}$, and finally multiplying the resulting point by $\#\tilde{E}(\mathbb{F}_{p^2})/n$ to obtain a point of order n. Since square roots in $\mathbb{F}_{p^2}$ cost $674m$ and point multiplication in $\tilde{E}(\mathbb{F}_{p^2})$ costs $3,052m$, hashing into $\mathbb{G}_2$ can be performed at a cost of $3,726m$. The alternate hashing technique of Galbraith and Scott [16] has approximately the same cost for our particular BN curve, but uses less memory.

As noted in [17], no efficient method is known for hashing into $\mathbb{G}_2'$. More precisely, what is meant is that no efficient hash function is known for which computing the discrete logarithms of hash values (to the base of the fixed generator P_2') is infeasible. This condition is imposed in order to eliminate from consideration the hash functions that first map a message to an integer z in the interval $[0,n-1]$, and thereafter define the hash value to be zP_2'. All signature schemes considered in this paper would be insecure if such hash functions were employed.

3.3.4. *Pairing.* As Lemma 1 explains, a Type 2 pairing $e_n(P,Q)$ can be reduced to a Type 3 pairing $R_n(P,\hat{Q})$, where $\hat{Q}=Q-\pi^6(Q)$. Given the representation (Q_1,Q_2) of Q, we have $\pi^6(Q)=\pi^6(Q_1+Q_2)=\pi^6(Q_1)+\pi^6(Q_2)=Q_1-Q_2$. Hence $\hat{Q}=(\infty,2Q_2)$ and so computing $\hat{Q}$ requires only one doubling in $\mathbb{G}_2$. For simplicity, we will use the same cost estimate of 15,175m for computing e_n and R_n.

3.3.5. *Testing membership in* $\mathbb{G}_1$*,* $\mathbb{G}_2$*,* $\mathbb{G}_2'$*.* Our descriptions of the signature schemes implicitly assume that all signature components are *valid*, i.e., belong to the appropriate group. Since the reductionist security proofs for these schemes require that signature components be valid, it seems prudent that the signature schemes include the validity checks. (However, we should mention that no attacks of the signature schemes have been reported in the literature if the validity checks are omitted.) For example, in the BLS-2 signature scheme (see §4.1.1) a verifier should first confirm that the signature σ is indeed an element of $\mathbb{G}_1$ before proceeding with the verification, while in Waters-2b (see §4.2.3) the verifier should first confirm that $\alpha\in\mathbb{G}_1$ and $\beta\in\mathbb{G}_2'$. Note that testing membership of public key elements needs to be performed only once by a certification authority who issues certificates for public keys.

For the BN curve under consideration, testing membership of a point Q in $\mathbb{G}_1$ is very efficient, and simply amounts to verifying that Q belongs to $E(\mathbb{F}_p)$. This costs a small number of $\mathbb{F}_p$ multiplications, which we will henceforth take as 'free'. Testing membership of a point Q in $\mathbb{G}_2$ involves a fast check that $\phi^{-1}(Q)$ is in $\tilde{E}(\mathbb{F}_{p^2})$, followed by an exponentiation in $\mathbb{G}_2$ to verify that $nQ=\infty$. Testing membership of (Q_1,Q_2) in $\mathbb{G}_2'$ is more costly — it can be done by first verifying that $Q_1\in\mathbb{G}_1$ and $Q_2\in\mathbb{G}_2$, and finally testing that $R_n(Q_1,T_2)=R_n(T_1,Q_2)$, where $D(P_2')=(T_1,T_2)$ can be precomputed. The total cost is 23,775m.

4. BLS versus Waters

4.1. **BLS signature scheme.** The Boneh-Lynn-Shacham (BLS) signature scheme [9] is notable because signatures can be considerably shorter than ElGamal-type signatures. The scheme was described in [9] in the setting of Type 1 and Type 2 pairings.

4.1.1. *BLS-2 signature scheme.* Let $e : \mathbb{G}_1 \times \mathbb{G}_2' \to \mathbb{G}_T$ be a Type 2 pairing, let $\psi : \mathbb{G}_2' \to \mathbb{G}_1$ be an efficiently-computable isomorphism with $\psi(g_2') = g_1$, and let $H : \{0,1\}^* \to \mathbb{G}_1$ be a hash function.

Alice's private key is an integer $x \in_R [1, n-1]$, while her public key is $X = (g_2')^x$. To sign a message M, Alice computes $h = H(M)$ and $\sigma = h^x$. Her signature on M is σ. To verify the signed message (M, σ), Bob computes $h = H(M)$ and accepts if and only if $e(\sigma, g_2') = e(h, X)$.

Correctness of the verification algorithm follows because

$$e(\sigma, g_2') = e(h^x, g_2') = e(h, (g_2')^x) = e(h, X).$$

The following security result was proven in [9]. We present an informal outline of a proof that contains the essential ideas behind the conventional reductionist security argument. Recall that a signature scheme is said to be *secure* if it is existentially unforgeable under an adaptive chosen-message attack by a computationally bounded adversary.

Theorem 1. *If co-DHP in* $(\mathbb{G}_1, \mathbb{G}_2')$ *is hard, and* H *is a random function, then the BLS-2 signature scheme is secure.*

Argument. The adversary $\mathcal{A}$ is given a public key $X \in \mathbb{G}_2'$ and a signing oracle. Because of the assumption regarding randomness of the hash function, the choice of messages that $\mathcal{A}$ submits to the signing oracle is irrelevant. Hence the signing oracle provides $\sigma \in \mathbb{G}_1$ such that $e(\sigma, g_2') = e(h, X)$ for random $h \in \mathbb{G}_1$. However, since $\mathcal{A}$ can generate such (h, σ) pairs itself by selecting random integers y and computing $h = g_1^y$ and $\sigma = \psi(X)^y$, the signing oracle is effectively useless to $\mathcal{A}$.

Thus $\mathcal{A}$'s task is reduced to computing $\sigma = h^x$ given $h \in_R \mathbb{G}_1$ and $X \in \mathbb{G}_2'$. This is precisely an instance of co-DHP in $(\mathbb{G}_1, \mathbb{G}_2')$. □

4.1.2. *BLS-3 signature scheme.* Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a Type 3 pairing, and let $H : \{0,1\}^* \to \mathbb{G}_1$ be a hash function. The BLS-3 signature scheme is identical to the BLS-2 scheme, except that the public key is $X = g_2^x \in \mathbb{G}_2$ instead of $X = (g_2')^x \in \mathbb{G}_2'$. Theorem 2 asserts that security of BLS-3 in the random oracle model is implied by the co-DHP* assumption; we omit the standard reductionist security proof.

Theorem 2. *If co-DHP* in* $(\mathbb{G}_1, \mathbb{G}_2)$ *is hard, and* H *is a random function, then the BLS-3 signature scheme is secure.*

It is straightforward to show that BLS-2 is insecure if co-DHP is easy, i.e., the security of BLS-2 is *equivalent* to co-DHP. Interestingly, the security of BLS-3 is not known to be equivalent to co-DHP*; that is, it is not known whether BLS-3 is secure in the event that an efficient algorithm is discovered for solving co-DHP*.

Let us denote by BLS-3b the variant of BLS-3 where Alice's public key is $(W = g_1^x, X = g_2^x)$. A certification authority who issues a certificate to Alice is responsible for checking that (W, X) is a valid public key; this entails verifying that $W \in \mathbb{G}_1$, $X \in \mathbb{G}_2$, $W \neq 1$, $X \neq 1$, and $e(W, g_2) = e(g_1, X)$. Signature generation and verification for BLS-3b are identical to that of BLS-3. It is easy to see that an efficient algorithm for co-DHP* can be used to break BLS-3b. Conversely, the argument for Theorem 1 can be modified to show that hardness of co-DHP* implies the security of BLS-3b.

Theorem 3. *If co-DHP* in* $(\mathbb{G}_1, \mathbb{G}_2)$ *is hard, and* H *is a random function, then the BLS-3b signature scheme is secure.*

Argument. The adversary $\mathcal{A}$ is given a public key (W, X) and a signing oracle. Because of the assumption regarding randomness of the hash function, the choice of messages that $\mathcal{A}$ submits to the signing oracle is irrelevant. Hence the signing oracle provides $\sigma \in \mathbb{G}_1$ such that $e(\sigma, g_2) = e(h, X)$ for random $h \in \mathbb{G}_1$. However, since $\mathcal{A}$ can generate such (h, σ) pairs itself by selecting random integers y and computing $h = g_1^y$ and $\sigma = W^y$, the signing oracle is effectively useless to $\mathcal{A}$.

Thus $\mathcal{A}$'s task is reduced to computing $\sigma = h^x$ given $h \in_R \mathbb{G}_1$ and $(W, X) \in \mathbb{G}_1 \times \mathbb{G}_2$. This is precisely an instance of co-DHP* in $(\mathbb{G}_1, \mathbb{G}_2)$. □

4.2. **Waters signature scheme.** The Waters signature scheme [41] is notable because it has a security proof that does not make the random oracle assumption. The scheme was originally described in the setting of a Type 1 pairing.

4.2.1. *Waters-1 signature scheme.* Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a Type 1 pairing, let k denote the security parameter, and let $\bar{H} : \{0,1\}^* \to \{0,1\}^k$ be a collision-resistant hash function. Let $u_0, u_1, \ldots, u_k$ be randomly selected elements of $\mathbb{G}$, and denote $U = (u_0, u_1, \ldots, u_k)$. The Waters hash function $H : \{0,1\}^* \to \mathbb{G}$ is defined as $H(M) = u_0 \prod_{i=1}^k u_i^{m_i}$, where $\bar{H}(M) = (m_1, m_2, \ldots, m_k)$ and each $m_i \in \{0,1\}$.

Alice's private key is a randomly chosen group element $Z = g^z \in \mathbb{G}$, while her public key is $\zeta = e(g,g)^z$. To sign a message M, Alice selects $r \in_R [1, n-1]$ and computes $h = H(M)$, $\alpha = Zh^r$, and $\beta = g^r$. Her signature on M is $\sigma = (\alpha, \beta)$. To verify the signed message (M, σ), Bob computes $h = H(M)$ and accepts if and only if $e(\alpha, g) = \zeta \cdot e(\beta, h)$. Correctness of the verification algorithm follows because

$$e(\alpha, g) = e(Zh^r, g) = e(Z, g) \cdot e(h^r, g) = e(g,g)^z \cdot e(\beta, h) = \zeta \cdot e(\beta, h).$$

Observe that if a party knows the logarithms $x_i = \log_g u_i$ for each $0 \leq i \leq k$, then that party can recover Alice's private key Z from a single signed message (M, σ). This is because if $\bar{H}(M) = (m_1, m_2, \ldots, m_k)$ and $c = x_0 + \sum_{i=1}^k m_i x_i \bmod n$, then

$$h = H(M) = u_0 \prod_{i=1}^k u_i^{m_i} = g^{x_0} \prod_{i=1}^k g^{x_i m_i} = g^{x_0 + \sum_{i=1}^k m_i x_i} = g^c,$$

and consequently the party can compute c and $Z = \alpha/\beta^c$. Thus it is imperative that no party know the discrete logarithms of the hash function parameters u_i. This property can be ensured by requiring that the u_i's be generated verifiably at random by a third party, i.e., a third party selects the u_i as outputs of a one-way function and makes the inputs publicly available.

Observe also that an attacker who learns the per-message secret r corresponding to a single signed message (M, σ) can recover Alice's private key $Z = \alpha/h^r$. Thus, per-message secrets in the Waters-1 signature scheme (and also in the variants of Waters and LOSSW considered in this paper) have to be securely generated, used, and destroyed, just as in the ElGamal signature scheme and its many variants. The BLS and BGLS schemes, which are deterministic and do not have any per-message secrets, do not have this drawback.

The following result was proven in [41]. We present an informal outline of the proof.

Theorem 4. *If DHP in $\mathbb{G}$ is hard, and $\bar{H}$ is collision-resistant, then the Waters-1 signature scheme is secure.*

Argument. Suppose we are given an instance (g^x, g^y) of the DHP in $\mathbb{G}$. We show how an adversary $\mathcal{A}$ of the Waters-1 scheme can be used to compute g^{xy}.

Set $\zeta = e(g^x, g^y) = e(g,g)^{xy}$; the corresponding (unknown) private key is $Z = g^{xy}$. Let q be an upper bound on the number of signing queries made by $\mathcal{A}$, and select $t \in_R [0, k]$. Select $a_0, a_1, \ldots, a_k \in_R [0, q-1]$ and $b_0, b_1, \ldots, b_k \in_R [0, n-1]$, and compute $u_0 = (g^x)^{a_0 - tq} g^{b_0}$ and $u_i = (g^x)^{a_i} g^{b_i}$ for $1 \leq i \leq k$. Note that $H(M) = u_0 \prod_{i=1}^k u_i^{m_i} = g^{xF+J}$, where $F = F(M) = -tq + a_0 + \sum_{i=1}^k a_i m_i \bmod n$ and $J = J(M) = b_0 + \sum_{i=1}^k b_i m_i \bmod n$. Next, run $\mathcal{A}$ with public key ζ and hash function parameters $U = (u_0, u_1, \ldots, u_k)$.

A request by $\mathcal{A}$ for a signature on a message M' is handled as follows. Compute $F = F(M')$ and $J = J(M')$. If $F = 0$, then the experiment is aborted. Otherwise, select $\hat{r} \in_R [1, n-1]$ and return the signature

$\sigma = (\alpha, \beta)$ where $\alpha = (g^y)^{-J/F} h^{\hat{r}}$ and $\beta = g^{\hat{r}}(g^y)^{-1/F}$. To see that (α, β) is a valid signature on M', set $r = \hat{r} - y/F$ and observe that $\beta = g^{r+y/F} g^{-y/F} = g^r$ and $\alpha = g^{-yJ/F} g^{(xF+J)(r+y/F)} = g^{xy} g^{(xF+J)r} = Zh^r$.

Suppose now that $\mathcal{A}$ eventually outputs a valid signature $\sigma = (\alpha, \beta)$ on a (new) message M. If $F(M) \neq 0$, then the experiment is aborted. Otherwise, we must have $\beta = g^r$ and $\alpha = Zh^r = Z(g^r)^J$, and hence $Z = \alpha/\beta^J$ can be computed. It can be verified that the probability[4] of not aborting is at least

$$\left(1 - \frac{1}{q}\right)^q \frac{1}{(k+1)q} \approx \frac{1}{(k+1)q}.$$

□

4.2.2. *Waters-2a signature scheme.* Let $e : \mathbb{G}_1 \times \mathbb{G}_2' \to \mathbb{G}_T$ be a Type 2 pairing, and let $\psi : \mathbb{G}_2' \to \mathbb{G}_1$ be an efficiently-computable isomorphism with $\psi(g_2') = g_1$. There are many ways in which the Waters-1 signature scheme can be implemented in the setting of a Type 2 pairing. We consider two variants — one having very short signatures, and the other having short, shared hash function parameters.

Suppose first that we insist on short signatures, i.e., $\alpha, \beta \in \mathbb{G}_1$. Then the verification equation must be $e(\alpha, g_2') = \zeta \cdot e(\beta, h)$, and hence $h = H(M) \in \mathbb{G}_2'$. However, as mentioned in §3.3.3, no efficient method is known for hashing into $\mathbb{G}_2'$ and hence the hash function parameters U cannot be generated verifiably at random by a third party. Thus, it would appear that each user Alice must generate her own hash function parameters $U = (u_0, u_1, \ldots, u_k)$ by selecting $x_i \in_r [0, n-1]$ and computing $u_i = (g_2')^{x_i}$ for $0 \leq i \leq k$; Alice's public key then consists of U in addition to $\zeta = e(g_1, g_2')^z$. In order to accelerate signature generation, Alice stores $X = (x_0, x_1, \ldots, x_k)$ as a private key in addition to $Z = g_1^z$.

To sign a message M, Alice selects $r \in_R [1, n-1]$ and computes $h' = g_1^{x_0 + \sum m_i x_i}$; note that $h' = \psi(H(M))$. Then Alice's signature on M is $\sigma = (\alpha, \beta)$, where $\alpha = Z(h')^r$ and $\beta = g_1^r$. To verify the signed message (M, σ), Bob computes $h = H(M)$ and accepts if and only if $e(\alpha, g_2') = \zeta \cdot e(\beta, h)$. The proof of Theorem 4 can be readily modified to obtain the following.

Theorem 5. *If co-DHP in $(\mathbb{G}_1, \mathbb{G}_2')$ is hard, and $\bar{H}$ is collision-resistant, then the Waters-2a signature scheme is secure.*

4.2.3. *Waters-2b signature scheme.* One disadvantage of the Waters-2a scheme is that public keys are very large. If one insists on short and shared hash function parameters U, then the following variant of the Waters scheme can be deployed.

A third party generates the hash function parameters $U \in \mathbb{G}_1^{k+1}$ verifiably at random. Alice's private key is $Z = g_1^z$, and her public key is $\zeta = e(g_1, g_2')^z$. To sign a message M, Alice selects $r \in_R [1, n-1]$ and computes $h = H(M)$, $\alpha = Zh^r$ and $\beta = (g_2')^r$. Alice's signature on M is $\sigma = (\alpha, \beta)$. To verify the signed message (M, σ), Bob computes $h = H(M)$ and accepts if and only if $e(\alpha, g_2') = \zeta \cdot e(h, \beta)$. As before, the proof of Theorem 4 can be readily modified to obtain the following.

Theorem 6. *If co-DHP in $(\mathbb{G}_1, \mathbb{G}_2')$ is hard, and $\bar{H}$ is collision-resistant, then the Waters-2b signature scheme is secure.*

4.2.4. *Waters-3a signature scheme.* Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a Type 3 pairing, and let $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ be the isomorphism satisfying $\psi(g_2) = g_1$. There are many ways in which the Waters-1 signature scheme can be implemented in the setting of a Type 3 pairing. As in the case of Waters-2, we consider two variants — one having very short signatures, and the other having short, shared hash function parameters.

Suppose first that we insist on short signatures, i.e., $\alpha, \beta \in \mathbb{G}_1$. Then the verification equation must be $e(\alpha, g_2) = \zeta \cdot e(\beta, h)$, and hence $h = H(M) \in \mathbb{G}_2$. However, the signer computes $\alpha = Zh^r$, and so we also require $h \in \mathbb{G}_1$. To get around the fact that an efficient algorithm for computing ψ is not known, we require

[4] The derivation of this probability assumes that the $\bar{H}(M')$ are pairwise distinct for the messages M' whose signatures were requested by $\mathcal{A}$. This assumption is valid because of the collision resistance of $\bar{H}$.

that each user Alice select $x_i \in_R [0, n-1]$ and compute $u_i = g_2^{x_i}$ for $0 \le i \le k$. Then Alice's private key is $Z = g_1^z$ and $X = (x_0, x_1, \ldots, x_k)$, while her public key is $\zeta = e(g_1, g_2)^z$ and $U = (u_0, u_1, \ldots, u_k)$.

To sign a message M, Alice selects $r \in_R [1, n-1]$ and computes $h' = g_1^{x_0 + \sum m_i x_i}$; note that $h' = \psi(H(M))$. Then Alice's signature on M is $\sigma = (\alpha, \beta)$, where $\alpha = Z(h')^r$ and $\beta = g_1^r$. To verify the signed message (M, σ), Bob computes $h = H(M)$ and accepts if and only if $e(\alpha, g_2) = \zeta \cdot e(\beta, h)$. The proof of Theorem 4 can be readily modified to obtain the following.

Theorem 7. _If co-DHP* in $(\mathbb{G}_1, \mathbb{G}_2)$ is hard, and $\bar{H}$ is collision-resistant, then the Waters-3a signature scheme is secure._

4.2.5. _Waters-3b signature scheme._ One disadvantage of the Waters-3a scheme is that public keys are very large. If one insists on short and shared hash function parameters U, then the following variant of the Waters scheme can be deployed.

A third party generates the hash function parameters $U \in \mathbb{G}_1^{k+1}$ verifiably at random. Alice's private key is $Z = g_1^z$, and her public key is $\zeta = e(g_1, g_2)^z$. To sign a message M, Alice selects $r \in_R [1, n-1]$ and computes $h = H(M)$, $\alpha = Zh^r$, $\beta = g_2^r$, and $\gamma = g_1^r$. Alice's signature on M is $\sigma = (\alpha, \beta, \gamma)$. To verify the signed message (M, σ), Bob computes $h = H(M)$ and accepts if and only if $e(\alpha, g_2) = \zeta \cdot e(h, \beta)$ and $e(\gamma, g_2) = e(g_1, \beta)$.

The extra signature component γ appears to be necessary in order for the security reduction to go through; that is, when the adversary $\mathcal{A}$ (see the proof of Theorem 4) returns a forgery (M, σ) where $\sigma = (\alpha, \beta, \gamma)$, we have $\alpha = Zh^r \in \mathbb{G}_1$, $\beta = g_2^r \in \mathbb{G}_2$, and $\gamma = g_1^r \in \mathbb{G}_1$. Hence the co-DHP* solver can compute $Z = \alpha / \gamma^J$. With this change in mind, the proof of Theorem 7 can be modified to establish the following.

Theorem 8. _If co-DHP* in $(\mathbb{G}_1, \mathbb{G}_2)$ is hard, and $\bar{H}$ is collision-resistant, then the Waters-3b signature scheme is secure._

4.3. **Comparisons.** Table 3 compares the BLS and Waters signature schemes when implemented using Type 2 and Type 3 pairings derived from the BN curve described in §3. The sizes and operation costs were derived from the estimates given in Table 2. Note that the operation costs include the cost of testing group membership (cf. §3.3.5).

	BLS-2	BLS-3	Waters-2a	Waters-2b	Waters-3a	Waters-3b
Security	ROM	ROM	collision resistance		collision resistance	
assumptions	co-DHP	co-DHP*	co-DHP		co-DHP*	
Public key size	770	513	48.3KB	1,024	32.1KB	1,024
Signature size	257	257	514	1027	514	1027
Sig. generation	1,848	1,848	1,447	5,576	1,447	5,576
Sig. verification	22,342	22,027	26,601	47,210	25,193	47,210

Table 3. Comparisons of the BLS and Waters signature schemes implemented with Type 2 and Type 3 pairings derived from the BN curve described in §3. Public key and signature sizes are given in bits or kilobytes (KB), while signature generation and verification costs are in terms of $\mathbb{F}_p$ multiplications.

For example, in the Waters-2a scheme (cf. §4.2.2), a public key consists of an element $\zeta \in \mathbb{G}_T$ and 257 hash function parameters $u_i \in \mathbb{G}_2'$. Recall that in our representation of $\mathbb{G}_2'$, we have $u_i = (Q_{i,1}, Q_{i,2})$ where $Q_{i,1} \in \mathbb{G}_1$ and $Q_{i,2} \in \mathbb{G}_2$. These elements are not compressed due to the high decompression cost, so the public key size is 48.3 kilobytes. A signature has two components, $\alpha, \beta \in \mathbb{G}_1$, each of which can be compressed to 257 bits, giving a signature size of 514 bits. Signature generation requires two fixed-based exponentiations in $\mathbb{G}_1$ — one to compute $(h')^r = g_1^{(x_0 + \sum m_i x_i) r}$ and the other to compute $\beta = g_1^r$ — and one multiplication in $\mathbb{G}_1$ to compute α, yielding a total cost of $1,447m$. Signature verification requires 128 $\mathbb{G}_2'$ multiplications on average to compute $h = u_0 \prod_{i=1}^{256} u_i^{m_i}$, two $\mathbb{G}_1$-decompressions (of α and β), and a product of pairings $e(\alpha, g_2') \cdot e(\beta^{-1}, h)$, for a total cost of $26,601m$.

Examining Table 3, we see that BLS-2 and BLS-3 have similar performance metrics, except that public key sizes in the former are slightly larger and signature verification slightly slower. Waters-3a outperforms Waters-2a in terms of verification speed and public key size while, somewhat surprisingly, Waters-2b and Waters-3b have identical performance metrics.

However, the differences between BLS-2 and BLS-3 are quite superficial. The BLS-2 public key $X \in \mathbb{G}_2'$ is larger than the BLS-3 public key only because the former includes an extra $\mathbb{G}_1$-component. This component is not actually used in signature generation or verification (recall from §3.3.4 that computing the Type 2 pairing $e_n(h, X)$ is equivalent to computing a Type 3 pairing $R_n(h, X_2)$ where X_2 is derived solely from the $\mathbb{G}_2$-component of X). BLS-2 signature verification is slightly slower than BLS-3 because of the cost of decompressing the unnecessary $\mathbb{G}_1$-component of X. Thus, if the generators P_1, P_2, P_2' are chosen so that co-DHP and co-DHP* are provably equivalent (see §2.4), then there is no reason to include the extra $\mathbb{G}_1$ component in the BLS-2 public key — in that case BLS-2 is identical to BLS-3 in every respect. If, on the other hand, P_1, P_2, P_2' are chosen so that co-DHP and co-DHP* cannot be proven equivalent, then the only significant difference between BLS-2 and BLS-3 is that the former's reductionist proof is with respect to co-DHP instead of co-DHP*.

Similarly, the differences between Waters-2a and Waters-3a arise because hashing in the former produces both a $\mathbb{G}_1$-component and a $\mathbb{G}_2$-component. If P_1, P_2, P_2' are chosen so that co-DHP and co-DHP* are provably equivalent, then the extra $\mathbb{G}_1$-component in Waters-2a can be dropped in which case Waters-2a and Waters-3a become identical.

Overall, BLS-3 is the superior scheme, outperforming both Waters-3a and Waters-3b (except that signature generation is slightly slower than for Waters-3a), with its only drawback being that the security proof makes the random oracle assumption.

5. BGLS versus LOSSW

Roughly speaking, an aggregate signature scheme is a signature scheme which has the additional property that, given signatures $\sigma_1, \sigma_2, \ldots, \sigma_\ell$ on messages $M_1, M_2, \ldots, M_\ell$ generated by ℓ entities $A_1, A_2, \ldots, A_\ell$, anyone can compute a *single* signature σ which can be used by a verifier to confirm the authenticity of $M_1, M_2, \ldots, M_\ell$. Aggregate signature schemes have found applications in secure routing protocols [42, 43], storing ballots on voting machines [4], and micropayment systems [10].

5.1. **BGLS aggregate signature scheme.**

BGLS, which is based on the BLS signature scheme, was originally described in the setting of a Type 2 pairing [7]. We present BGLS-2 in §5.1.1, and then describe BGLS-3, the BGLS variant that is based on the BLS-3b signature scheme.

5.1.1. *BGLS-2 aggregate signature scheme.* Let $e : \mathbb{G}_1 \times \mathbb{G}_2' \to \mathbb{G}_T$ be a Type 2 pairing, let $\psi : \mathbb{G}_2' \to \mathbb{G}_1$ be an efficiently-computable isomorphism with $\psi(g_2') = g_1$, and let $H : \{0,1\}^* \to \mathbb{G}_1$ be a hash function.

In the BGLS-2 scheme, each user A_i has a BLS-2 private key x_i and public key $X_i = (g_2')^{x_i}$. In the signature generation stage, each user A_i generates its BLS-2 signature $\sigma_i = h_i^{x_i}$ on message M_i, where $h_i = H(M_i)$. The aggregate signature is $\sigma = \prod \sigma_i$, and can be computed by any party. To verify an aggregate signature σ, a verifier first checks that the M_i are pairwise distinct[5], then computes $h_i = H(M_i)$, and finally accepts if and only if $e(\sigma, g_2') = \prod e(h_i, X_i)$. Correctness of the verification algorithm follows because

$$e(\sigma, g_2') = e(\prod \sigma_i, g_2') = \prod e(\sigma_i, g_2') = \prod e(h_i, X_i).$$

[5] The requirement that the messages M_i be pairwise distinct can be removed by hashing the public key X_i together with M_i; see [3].

An aggregate signature scheme is said to be *secure* [7] if no computationally bounded adversary is successful in the following task. The adversary $\mathcal{A}$ is given a victim's target public key X_1 and a signing oracle with respect to X_1. $\mathcal{A}$'s task is to produce $\ell - 1$ public keys $X_2, \ldots, X_\ell$ (for any $\ell > 1$ of $\mathcal{A}$'s choosing), ℓ messages $M_1, \ldots, M_\ell$, and a valid aggregate signature σ (on $M_1, \ldots, M_\ell$ with respect to $X_1, \ldots, X_\ell$). Of course, $\mathcal{A}$ cannot have queried the signing oracle with M_1.

The following security result was proven in [7]. We present an informal outline of a proof that contains the essential ideas behind the conventional reductionist security argument.

Theorem 9. *If co-DHP in $(\mathbb{G}_1, \mathbb{G}'_2)$ is hard, and H is a random function, then the BGLS-2 aggregate signature scheme is secure.*

Argument. For simplicity, we assume that $\ell = 2$. The adversary $\mathcal{A}$ is given a public key $X_1 \in \mathbb{G}'_2$ and a signing oracle with respect to X_1. As was argued in the proof of Theorem 1, the assumption regarding randomness of H implies that the signing oracle is effectively useless to $\mathcal{A}$.

Since H is a random function, $\mathcal{A}$ may as well select M_1 and M_2 first (with $M_1 \neq M_2$, and consequently $H(M_1) = H(M_2)$ with negligible probability). Its task then is to find $X_2 \in \mathbb{G}'_2$ and $\sigma \in \mathbb{G}_1$ such that

$$e(\sigma, g'_2) = e(h_1, X_1) \cdot e(h_2, X_2), \tag{4}$$

where $h_1 = H(M_1)$ and $h_2 = H(M_2)$. It can now be seen that an algorithm $\mathcal{P}$ for solving this problem yields an algorithm for solving co-DHP. Namely, given a co-DHP instance (h_1, X_1), select random y_2 and compute $h_2 = g_1^{y_2}$. Then use $\mathcal{P}$ to find $X_2 \in \mathbb{G}'_2$ and $\sigma \in \mathbb{G}_1$ satisfying (4). Then $\sigma = h_1^{x_1} h_2^{x_2} = h_1^{x_1} \psi(X_2)^{y_2}$, where $x_1 = \log_{g'_2} X_1$ and $x_2 = \log_{g'_2} X_2$. Thus the co-DHP solution can be obtained by computing $\sigma / \psi(X_2)^{y_2} = h_1^{x_1}$. □

5.1.2. *BGLS-3 aggregate signature scheme.* Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a Type 3 pairing, and let $H : \{0,1\}^* \to \mathbb{G}_1$ be a hash function. Let $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ be the isomorphism satisfying $\psi(g_2) = g_1$.

In the BGLS-3 scheme, each user A_i has a BLS-3b private key x_i and public key $(W_i = g_1^{x_i}, X_i = g_2^{x_i})$. A certification authority who issues a certificate for A_i's public key is responsible for checking that (W_i, X_i) is a valid public key. Signature generation, signature aggregation, and signature verification are identical to BGLS-2. Since W_i is not used during signature generation, aggregation or verification, it need not be included in the public key; however its inclusion in the certification process appears to be necessary for a reductionist security proof to go through.

Theorem 10. *If co-DHP* in $(\mathbb{G}_1, \mathbb{G}_2)$ is hard, and H is a random function, then the BGLS-3 aggregate signature scheme is secure.*

Argument. For simplicity, we assume that $\ell = 2$. The adversary $\mathcal{A}$ is given a public key (W_1, X_1) and a signing oracle with respect to (W_1, X_1). As was argued in the proof of Theorem 3, the assumption regarding randomness of H implies that the signing oracle is effectively useless to $\mathcal{A}$.

Since H is a random function, $\mathcal{A}$ may as well select M_1 and M_2 first (with $M_1 \neq M_2$, and consequently $H(M_1) = H(M_2)$ with negligible probability). Its task then is to find $X_2 \in \mathbb{G}_2$, $W_2 = \psi(X_2)$, and $\sigma \in \mathbb{G}_1$ satisfying (4). It can now be seen that an algorithm $\mathcal{P}$ for solving this problem yields an algorithm for solving co-DHP*. Namely, given a co-DHP* instance (h_1, W_1, X_1) (where $h_1 \in \mathbb{G}_1$, $X_1 \in \mathbb{G}_2$ and $W_1 = \psi(X_1)$), select random y_2 and compute $h_2 = g_1^{y_2}$. Then use $\mathcal{P}$ to find $X_2 \in \mathbb{G}_2$, $W_2 = \psi(X_2)$, and $\sigma \in \mathbb{G}_1$ satisfying (4). Then $\sigma = h_1^{x_1} h_2^{x_2} = h_1^{x_1} W_2^{y_2}$, where $x_1 = \log_{g_2} X_1$ and $x_2 = \log_{g_2} X_2$. Thus the co-DHP* solution can be obtained by computing $\sigma / W_2^{y_2} = h_1^{x_1}$. □

5.2. **LOSSW aggregate signature scheme.** LOSSW, which is based on the Waters signature scheme, was originally described in the setting of a Type 1 pairing [29]. We present LOSSW-1 in §5.2.1. In §5.2.2 and §5.2.3, we present the LOSSW variants that are based on the Waters-3a and Waters-3b signature schemes. We will not consider LOSSW in the setting of a Type 2 pairing because, as seen in §4.3, Type 2 pairings do not offer any advantages over Type 3 pairings for the Waters signature scheme.

5.2.1. *LOSSW-1 aggregate signature scheme.* Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a Type 1 pairing. In the LOSSW-1 scheme, signature aggregation is *sequential.* The first user A_1 generates a Waters-1 signature (α_1, β_1) on a message M_1 and forwards the signed message to A_2. The second user verifies the signature of the received message, sets $\beta_2 = \beta_1$ and generates her signature (α_2', β_2) for message M_2; in order to compute α_2', A_2 needs to know the discrete logarithm of the hash function parameters, and therefore should have her own hash function parameters. A_2 then sets $\alpha_2 = \alpha_1 \cdot \alpha_2'$, randomizes β_2 and α_2, and forwards the messages M_1, M_2, and the aggregated-thus-far signature (α_2, β_2) to A_3. Similarly, A_3 proceeds to sign her message M_3, and aggregates her signature to obtain (α_3, β_3). LOSSW-1 signature verification is not sequential. We next describe LOSSW-1 in further detail.

Each user A_j has a private key (X_j, Z_j), where $X_j = (x_{j,0}, x_{j,1}, \ldots, x_{j,k})$ with $x_{j,i} \in_R [0, n-1]$, and $Z_j = g^{z_j}$ with $z_j \in_R [0, n-1]$. The corresponding public key is (U_j, ζ_j), where $U_j = (u_{j,0}, u_{j,1}, \ldots, u_{j,k})$ with $u_{j,i} = g^{x_{j,i}}$, and $\zeta_j = e(g,g)^{z_j}$. The parameters U_j define A_j's own Waters hash function $H_j : \{0,1\}^* \to \mathbb{G}$.

User A_1's Waters-1 signature on M_1 is (α_1, β_1), where $\alpha_1 = Z_1 h_1^{r_1}$ with $h_1 = H_1(M_1)$, and $\beta_1 = g^{r_1}$. Upon receiving (M_1, α_1, β_1) from A_1, user A_2 first verifies the signature. She then computes $\alpha_2' = Z_2 h_2^{r_1}$ where $h_2 = H_2(M_2)$. Since A_2 does not know r_1, she obtains $h_2^{r_1}$ by computing β_1^c where $c = \log_g H_2(M_2)$ (using her private key X_2). She then computes $\alpha_2 = \alpha_1 \alpha_2'$, selects $r_2 \in_R [0, n-1]$, and computes $\beta_2 = \beta_1 g^{r_2}$ and $\alpha_2 \leftarrow \alpha_2 h_1^{r_2} h_2^{r_2}$. Note that $\beta_2 = g^{r_1+r_2}$ and $\alpha_2 = Z_1 Z_2 h_1^{r_1+r_2} h_2^{r_1+r_2}$. A_2 forwards $(M_1, M_2, \alpha_2, \beta_2)$ to A_3 who first verifies the aggregated-thus-far signature (using the verification algorithm described next), signs M_3, and aggregates in a similar fashion.

Given a list of ℓ messages $M_1, M_2, \ldots, M_\ell$ purportedly signed by (pairwise distinct) users $A_1, A_2, \ldots, A_\ell$, and aggregate signature $(\alpha_\ell, \beta_\ell)$, a verifier computes $h_j = H_j(M_j)$ for $1 \le j \le \ell$ and accepts if and only if $e(\alpha_\ell, g) = (\prod_{j=1}^{\ell} \zeta_j) \cdot e(\beta_\ell, \prod_{j=1}^{\ell} h_j)$.

For $\ell = 2$, correctness of the verification algorithm follows because

$$e(\alpha_2, g) = e(Z_1 Z_2 h_1^{r_1+r_2} h_2^{r_1+r_2}, g) = e(Z_1, g) \cdot e(Z_2, g) \cdot e((h_1 h_2)^{r_1+r_2}, g) = \zeta_1 \cdot \zeta_2 \cdot e(\beta_2, h_1 h_2).$$

Correctness for $\ell > 2$ can be similarly checked.

A sequential aggregate signature scheme (for which signature verification is not sequential) is said to be *secure* [29] if no computationally bounded adversary is successful in the following task. The adversary $\mathcal{A}$ is given a victim's target public key X_1. $\mathcal{A}$ can request certification for any public key of its choosing, provided that it furnishes the corresponding private key. $\mathcal{A}$'s task is to produce $\ell - 1$ certified public keys $X_2, \ldots, X_\ell$ (for any $\ell > 1$ of $\mathcal{A}$'s choosing), ℓ messages $M_1, \ldots, M_\ell$, and a valid aggregate signature σ (on $M_1, \ldots, M_\ell$ with respect to $X_1, \ldots, X_\ell$). During its attack, $\mathcal{A}$ can, upon providing a valid aggregated-thus-far signature on a sequence of messages corresponding to a sequence of certified public keys (not including X_1), request the aggregation of the victim's signature on any message M'. Of course, M_1 cannot be one of the messages M' queried to its sequential aggregate signing oracle.

The security proof for LOSSW-1 shows how a successful LOSSW-1 forger can be used to break the Waters-1 signature scheme.

Theorem 11 ([29]). *If DHP in $\mathbb{G}$ is hard, and $\bar{H}$ is collision-resistant, then the LOSSW-1 aggregate signature scheme is secure.*

Argument. The LOSSW-1 adversary $\mathcal{A}$ is given the victim's Waters-1 public key (U_1, ζ_1). For simplicity, we assume that $\mathcal{A}$ selects exactly one key pair (U_2, ζ_2), (X_2, Z_2). Also for simplicity, we assume that all aggregate signatures are derived from two signatures, the first contributed by $\mathcal{A}$ and the second by the victim.

Now, a request by $\mathcal{A}$ for the aggregate signature on (M_2', M_1'), where (α_2, β_2) is $\mathcal{A}$'s (valid) signature on M_2', can be handled by asking the Waters-1 signing oracle for a signature (α_1, β_1) on M_1'. It can easily be verified that (α, β) is a valid aggregate signature where $\beta = \beta_1$ and $\alpha = \alpha_1 Z_2 \beta_1^c$, and where $c = x_{2,0} + \sum_{i=1}^{k} m_{2,i} x_{2,i}$

and $\bar{H}(M_2') = (m_{2,1}, \ldots, m_{2,k})$. Thus $\mathcal{A}$'s requests can be properly answered by using the Waters-1 signing oracle.

Eventually $\mathcal{A}$ produces a valid aggregate signature (α, β) on (M_1, M_2), where M_1 was not previously provided to its aggregate signing oracle (and hence M_1 was not queried to the Waters-1 signing oracle). Since $\alpha = Z_1 Z_2 H_1(M_1)^r H_2(M_2)^r$ where $\beta = g^r$, $\mathcal{A}$'s private key (X_2, Z_2) can then be used to produce the victim's signature $(Z_1 H_1(M_1)^r, \beta)$ on M_1, i.e., produce a Waters-1 forgery. The proof is now completed by appealing to Theorem 4. □

The security proof for LOSSW-1 makes essential use of the requirement that a user present its private key (X_j, Z_j) to the certification authority in order to get its public key (U_j, ζ_j) certified. A security model with this requirement is called the *certified-key model*. Requiring the certified-key model for LOSSW-1 and its variants is, of course, a significant drawback compared to the BGLS scheme where the certification authority is not entrusted with user private keys.

5.2.2. *LOSSW-3a aggregate signature scheme.* Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a Type 3 pairing, and let $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ be the isomorphism satisfying $\psi(g_2) = g_1$.

Each user A_j has a private key (X_j, Z_j), where $X_j = (x_{j,0}, x_{j,1}, \ldots, x_{j,k})$ with $x_{j,i} \in_R [0, n-1]$, and $Z_j = g_1^{z_j}$ with $z_j \in_R [0, n-1]$. The corresponding public key is (U_j, V_j, ζ_j), where $U_j = (u_{j,0}, u_{j,1}, \ldots, u_{j,k})$ with $u_{j,i} = g_2^{x_{j,i}}$, $V_j = (v_{j,0}, v_{j,1}, \ldots, v_{j,k})$ with $v_{j,i} = g_1^{x_{j,i}}$, and $\zeta_j = e(g_1, g_2)^{z_j}$. The parameters U_j define A_j's own Waters hash function $H_j : \{0,1\}^* \to \mathbb{G}_2$, while V_j defines $\psi(H_j)$, i.e., $\psi(H_j)(M) = \psi(H_j(M))$.

User A_1's Waters-3a signature on M_1 is (α_1, β_1), where $\alpha_1 = Z_1 (h_1')^{r_1}$ with $h_1' = \psi(H_1)(M_1)$, and $\beta_1 = g_1^{r_1}$. Upon receiving (M_1, α_1, β_1) from A_1, user A_2 verifies the signature, and uses her private key X_2 to compute $\alpha_2' = Z_2 (h_2')^{r_1}$ where $h_2' = \psi(H_2)(M_2)$. She then computes $\alpha_2 = \alpha_1 \alpha_2'$, selects $r_2 \in_R [0, n-1]$, and computes $\beta_2 = \beta_1 g_1^{r_2}$ and $\alpha_2 \leftarrow \alpha_2 (h_1')^{r_2} (h_2')^{r_2}$. A_2 forwards $(M_1, M_2, \alpha_2, \beta_2)$ to A_3 who verifies the aggregated-thus-far signature (using the verification algorithm described next), signs M_3, and aggregates in a similar fashion.

Given a list of ℓ messages $M_1, M_2, \ldots, M_\ell$ purportedly signed by (pairwise distinct) users $A_1, A_2, \ldots, A_\ell$, and aggregate signature $(\alpha_\ell, \beta_\ell)$, a verifier computes $h_j = H_j(M_j)$ for $1 \le j \le \ell$ and accepts if and only if $e(\alpha_\ell, g_2) = (\prod_{j=1}^{\ell} \zeta_j) \cdot e(\beta_\ell, \prod_{j=1}^{\ell} h_j)$.

Theorem 12 can be established by reducing an LOSSW-3a forger to a Waters-3a forger, where the Waters-3a signature scheme described in §4.2.4 is slightly modified to include $V = (v_0, v_1, \ldots, v_k)$ with $v_i = g_1^{x_i}$ in the public key.

Theorem 12. *If co-DHP* in $(\mathbb{G}_1, \mathbb{G}_2)$ is hard, and $\bar{H}$ is collision-resistant, then the LOSSW-3a aggregate signature scheme is secure.*

5.2.3. *LOSSW-3b aggregate signature scheme.* Each user A_j has a private key (X_j, Z_j), where $X_j = (x_{j,0}, x_{j,1}, \ldots, x_{j,k})$ with $x_{j,i} \in_R [0, n-1]$, and $Z_j = g_1^{z_j}$ with $z_j \in_R [0, n-1]$. The corresponding public key is (U_j, ζ_j), where $U_j = (u_{j,0}, u_{j,1}, \ldots, u_{j,k})$ with $u_{j,i} = g_1^{x_{j,i}}$, and $\zeta_j = e(g_1, g_2)^{z_j}$. The parameters U_j define A_j's own Waters hash function $H_j : \{0,1\}^* \to \mathbb{G}_1$.

User A_1's Waters-3b signature on M_1 is $(\alpha_1, \beta_1, \gamma_1)$, where $\alpha_1 = Z_1 h_1^{r_1}$ with $h_1 = H_1(M_1)$, $\beta_1 = g_2^{r_1}$, and $\gamma_1 = g_1^{r_1}$. Upon receiving $(M_1, \alpha_1, \beta_1, \gamma_1)$ from A_1, user A_2 verifies the signature, and uses her private key X_2 to compute $\alpha_2' = Z_2 h_2^{r_1}$ where $h_2 = H_2(M_2)$. She then computes $\alpha_2 = \alpha_1 \alpha_2'$, selects $r_2 \in_R [0, n-1]$, and computes $\beta_2 = \beta_1 g_2^{r_2}$, $\gamma_2 = \gamma_1 g_1^{r_2}$, and $\alpha_2 \leftarrow \alpha_2 h_1^{r_2} h_2^{r_2}$. A_2 forwards $(M_1, M_2, \alpha_2, \beta_2, \gamma_2)$ to A_3 who verifies the aggregated-thus-far signature (using the verification algorithm described next), signs M_3, and aggregates in a similar fashion.

Given a list of ℓ messages $M_1, M_2, \ldots, M_\ell$ purportedly signed by (pairwise distinct) users $A_1, A_2, \ldots, A_\ell$, and aggregate signature $(\alpha_\ell, \beta_\ell, \gamma_\ell)$, a verifier computes $h_j = H_j(M_j)$ for $1 \le j \le \ell$ and accepts if and only $e(\alpha_\ell, g_2) = (\prod_{j=1}^{\ell} \zeta_j) \cdot e(\prod_{j=1}^{\ell} h_j, \beta_\ell)$ and $e(\gamma_\ell, g_2) = e(g_1, \beta_\ell)$.

Theorem 13 can be established by reducing an LOSSW-3b forger to a Waters-3b forger.

Theorem 13. *If co-DHP* in $(\mathbb{G}_1, \mathbb{G}_2)$ is hard, and $\bar{H}$ is collision-resistant, then the LOSSW-3b aggregate signature scheme is secure.*

5.3. **Comparisons.** Table 4 compares the BGLS and LOSSW aggregate signature schemes when implemented using Type 2 and Type 3 pairings derived from the BN curve described in §3. The sizes and operation costs were derived from the estimates given in Table 2. Note that the operation costs include the cost of testing group membership (cf. §3.3.5).

	BGLS-2	BGLS-3	LOSSW-3a	LOSSW-3b
Security assumptions	ROM co-DHP	ROM co-DHP*	collision resistance co-DHP* certified key model	
Public key size	770	513	48.3KB	16.2KB
Signature size	257	257	514	1027
Sig. generation	1,848	1,848	26,505+5,343d	52,920+1,473d
Sig. verification	15,490+6,852ℓ	15,490+6,537ℓ	21,269+3,924ℓ	45,737+1,473ℓ

Table 4. Comparisons of the BGLS and LOSSW aggregate signature schemes implemented with Type 2 and Type 3 pairings derived from the BN curve described in §3. Public key and signature sizes are given in bits or kilobytes (KB), while signature generation and verification costs are in terms of $\mathbb{F}_p$ multiplications. The number of signatures aggregated thus far is denoted by d. The total number of signatures aggregated is denoted by ℓ.

For example, in the LOSSW-3a scheme (cf. §5.2.2), a public key consists of an element $\zeta_j \in \mathbb{G}_T$, 257 (uncompressed) hash function parameters $v_{j,i} \in \mathbb{G}_1$, and 257 (uncompressed) hash function parameters $u_{j,i} \in \mathbb{G}_2$, so the public key size is 48.3 kilobytes. An aggregate signature has two components $\alpha_d, \beta_d \in \mathbb{G}_1$, each of which can be compressed to 257 bits, giving a signature size of 514 bits. Signature verification requires two $\mathbb{G}_1$-decompressions (of α_ℓ and β_ℓ), ℓ evaluations of the hash functions H_j and $(\ell - 1)$ multiplications in $\mathbb{G}_2$ to compute $\prod h_j$, a product of pairings $e(\alpha_\ell, g_2) \cdot e(\beta_\ell^{-1}, \prod h_j)$, and $(\ell - 1)$ multiplications in $\mathbb{G}_T$ to compute $\prod \zeta_j$, for a total cost of $21,269 + 3,924\ell$ $\mathbb{F}_p$-multiplications. The $(d + 1)$-th signer first verifies the aggregated-thus-far signature at a cost of $21,269 + 3,924d$ $\mathbb{F}_p$-multiplications. Generating her signature component α'_{d+1} requires one exponentiation and one multiplication in $\mathbb{G}_1$, while producing β_{d+1} requires one fixed-based exponentiation and one multiplication in $\mathbb{G}_1$. Computing $(\prod h'_j)^{r_{d+1}}$ requires one evaluation each of $\psi(H_1), \ldots, \psi(H_{d+1})$, d multiplications in $\mathbb{G}_1$, and one exponentiation in $\mathbb{G}_1$. Finally, α_{d+1} can be computed with 2 further multiplications in $\mathbb{G}_1$. The total cost of signature generation by the $(d + 1)$-th signer is thus $26,505 + 5,343d$ $\mathbb{F}_p$-multiplications.

Examining Table 4, we see that BGLS-2 and BGLS-3 have similar performance metrics, except that public key sizes in the former are slightly larger and signature verification slightly slower. Signature generation for LOSSW-3a is faster than for LOSSW-3b for $d \le 6$, while LOSSW-3a signature verification is faster than LOSSW-3b verification for $\ell \le 9$. BGLS-3 has far smaller public keys, shorter signatures, and faster signature generation than LOSSW-3a and LOSSW-3b. It should be noted, however, that for applications of aggregate signatures where signing is inherently sequential and where a signer must verify the aggregated-thus-far signature before adding its own signature, the cost of BGLS-2 and BGLS-3 signature generation is effectively 17,338+6,852d and 17,338+6,537d, respectively. Signature verification for BGLS-3 is slower than LOSSW-3a if $\ell \ge 3$ and slower than LOSSW-3b if $\ell \ge 6$. However, BGLS-3 signature verification is not as slow as the preliminary analysis in [29] would suggest, even though BGLS-3 verification requires $\ell + 1$ pairings, while LOSSW-3a and LOSSW-3b only require 2 and 4 pairings, respectively. For example, if $\ell = 10$ signatures have been aggregated, then BGLS-3 signature verification costs 80,860m, while LOSSW-3a and LOSSW-3b verification cost 60,509m and 60,467m, respectively.

BGLS-3 signature verification times can be improved to 14,145+4,518ℓ $\mathbb{F}_p$-multiplications at the expense of increasing the public key size to 9.25 kilobytes. This speedup can be achieved by not compressing the public key X_i, and by precomputing and storing the 73 lines needed in Algorithm 2; these lines depend on X_i when the pairing $e(h_i, X_i)$ is evaluated, and on the fixed parameter g_2 when the pairing $e(\sigma, g_2)$ is evaluated. With these modifications, BGLS-3 signature verification is faster than LOSSW-3a for $\ell \leq 12$, and faster than LOSSW-3b for $\ell \leq 10$. In particular, if $\ell = 10$ then BGLS-3 verification costs 59,325m.

It would appear that the evaluation of a product of eta pairings [1] (for supersingular elliptic curves with embedding degrees $k = 4$ and $k = 6$) cannot be accelerated in the same manner as was done for the product of R-ate pairings in §2.1. Thus, LOSSW signature verification can be expected to be faster than BGLS when implemented with a Type 1 pairing.

6. Concluding remarks

We have shown that there is no performance or security benefit to be gained by using a Type 2 pairing instead of a Type 3 pairing when implementing the BLS, Waters, BGLS and LOSSW signature schemes with BN pairings. Our observations extend to Type 2 and Type 3 pairings derived from any elliptic curve with even embedding degree. Furthermore, they suggest more broadly that Type 2 pairings are merely inefficient implementations of Type 3 pairings, and offer no benefit for protocols based on asymmetric pairings from the point of view of functionality, security, and performance.

We also demonstrated that the BGLS aggregate signature scheme outperforms the LOSSW scheme in every respect except that signature verification in the latter is faster when a large number of signatures have been aggregated. Furthermore, any criticism of the use of the random oracle model in the reductionist security proof for BGLS would appear to be outweighed by the use of the certified-key model for LOSSW. Thus the removal of the random oracle assumption that was the main motivation for the design of the LOSSW scheme comes at a considerable price.

Our analysis of the various signature schemes is deficient in four respects.

(1) It neglects to account for the lack of tightness in the reductionist security proofs. This is not a concern for the BLS and BGLS schemes, as tight reductions can be achieved for slight variants obtained by applying the Katz-Wang trick [26] (see also [3]). However, the Katz-Wang trick does not appear to be applicable to the Waters and LOSSW schemes, whose reductionist security proofs are not tight.

(2) We did not utilize the Chatterjee-Sarkar/Naccache strategy [11, 32] to reduce the size of the hash function parameters in the Waters-2a, Waters-3a, LOSSW-3a and LOSSW-3b signature schemes. However, the possible reduction in the size of hash function parameters results in a very significant loss in the tightness of the security reduction (and no significant improvement in the signature generation and verification speeds), so the overall effectiveness of deploying the strategy is debatable in the present context. For example, the public key size for LOSSW-3b can be reduced from 16.2 kilobytes to 2,312 bits by grouping the bits of $\bar{H}(M)$ into 64-bit segments, but then the reductionist security proof becomes less tight by a factor of 2^{64}.

(3) The concrete estimates in Tables 3 and 4 are for a specific BN curve. Other well-chosen BN curves, such as the one with BN parameter $z = 4080000000000001$ (in hexadecimal) [33] may have different performance characteristics. Nonetheless, we expect that our conclusions about the relative performance of the various signature schemes will not drastically change for well-chosen BN curves.

(4) Even though our cost estimates are quite detailed, they ignore additions, subtractions and other overhead, and furthermore do not account for memory usage. Thus, the relative performance of the various signature schemes may change when implemented on a particular platform.

Acknowledgements

Thanks to Koray Karabina for showing us Lemma 2.

References

[1] 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.

[2] P. Barreto and M. Naehrig, "Pairing-friendly elliptic curves of prime order", *Selected Areas in Cryptography – SAC 2005*, Lecture Notes in Computer Science, 3897 (2006), 319–331.

[3] M. Bellare, C. Namprempre and G. Neven, "Unrestricted aggregate signatures", *Automata, Languages and Programming – ICALP 2007*, Lecture Notes in Computer Science, 4596 (2007), 411–422.

[4] J. Bethencourt, D. Boneh and B. Waters, "Cryptographic methods for storing ballots on a voting machine", *The 14th Annual Network & Distributed System Security Symposium – NDSS 2007*.

[5] B. den Boer, "Diffie-Hellman is as strong as discrete log for certain primes", *Advances in Cryptology – CRYPTO '88*, Lecture Notes in Computer Science, 403 (1996), 530–539.

[6] D. Boneh, X. Boyen and H. Shacham, "Short group signatures", *Advances in Cryptology – CRYPTO 2004*, Lecture Notes in Computer Science, 3152 (2004), 41–55.

[7] D. Boneh, C. Gentry, B. Lynn and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps", *Advances in Cryptology – EUROCRYPT 2003*, Lecture Notes in Computer Science, 2656 (2003), 416–432.

[8] D. Boneh and R. Lipton, "Algorithms for black-box fields and their application to cryptography", *Advances in Cryptology – CRYPTO '96*, Lecture Notes in Computer Science, 1109 (1996), 283–297.

[9] D. Boneh, B. Lynn and H. Shacham, "Short signatures from the Weil pairing", *Advances in Cryptology – ASIACRYPT 2001*, Lecture Notes in Computer Science, 2248 (2001), 514–532. Full version: *Journal of Cryptology*, 17 (2004), 297–319.

[10] D. Catalano, G. Ruffo and R. Schifanella, "A P2P market place based on aggregate signatures", *Parallel and Distributed Processing and Applications – ISPA 2005 Workshops*, Lecture Notes in Computer Science, 3759 (2005), 54–63.

[11] S. Chatterjee and P. Sarkar, "Trading time for space: Towards an efficient IBE scheme with short(er) public parameters in the standard model", *Information Security and Cryptology – ICISC 2005*, Lecture Notes in Computer Science, 3935 (2006), 424–440.

[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] J. Chung and A. Hasan, "Asymmetric squaring formulae", *18th IEEE Symposium on Computer Arithmetic (ARITH '07)*, 113–122.

[14] A. Devegili, M. Scott and R. Dahab, "Implementing cryptographic pairings over Barreto-Naehrig curves", *Pairing-Based Cryptography – Pairing 2007*, Lecture Notes in Computer Science 4575 (2007), 197–207.

[15] S. Galbraith, "Pairings", Chapter IX of I. Blake, G. Seroussi, and N. Smart, eds., *Advances in Elliptic Curve Cryptography*, Vol. 2, Cambridge University Press, 2005.

[16] S. Galbraith and M. Scott, "Exponentiation in pairing-friendly groups using homomorphisms", *Pairing-Based Cryptography – Pairing 2008*, Lecture Notes in Computer Science 5209 (2008), 211–224.

[17] S. Galbraith, K. Paterson and N. Smart, "Pairings for cryptographers", *Discrete Applied Mathematics*, 156 (2008), 3113–3121.

[18] R. Gallant, R. Lambert and S. Vanstone, "Faster point multiplication on elliptic curves with efficient endomorphisms", *Advances in Cryptology – CRYPTO 2001*, Lecture Notes in Computer Science, 2139 (2001), 190–200.

[19] D. Gordon, "Discrete logarithms in $GF(p)$ using the number field sieve", *SIAM Journal on Discrete Mathematics*, 6 (1993), 124–138.

[20] R. Granger, D. Page and M. Stam, "A comparison of CEILIDH and XTR", *Algorithmic Number Theory: 6th International Symposium, ANTS-VI*, Lecture Notes in Computer Science, 3076 (2004), 235–249.

[21] R. Granger and N. Smart, "On computing products of pairings", Cryptology ePrint Archive Report 2006/172, 2006. Available from http://eprint.iacr.org/2006/172.

[22] D. Hankerson, A. Menezes and M. Scott, "Software implementation of pairings", in *Identity-Based Cryptography*, M. Joye and G. Neven, eds., IOS Press, 2008.

[23] D. Hankerson, A. Menezes and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer, 2004.

[24] F. Hess, N. Smart and F. Vercauteren, "The eta pairing revisited", *IEEE Transactions on Information Theory*, 52 (2006), 4595–4602.

[25] B. Kang and J. Park, "On the relationship between squared pairings and plain pairings", Cryptology ePrint Archive Report 2005/112, 2005. Available from http://eprint.iacr.org/2005/112.

[26] J. Katz and N. Wang, "Efficiency improvements for signature schemes with tight security reductions", *Proceedings of the 10th ACM Conference on Computer and Communications Security*, 2003, 155–164.

[27] A. Joux, "A one round protocol for tripartite Diffie-Hellman", *Algorithmic Number Theory: 4th International Symposium, ANTS-IV*, Lecture Notes in Computer Science, 1838 (2000), 385–393.

[28] E. Lee, H.-S. Lee and C.-M. Park, "Efficient and generalized pairing computation on abelian varieties", *IEEE Transactions on Information Theory*, 55 (2009), 1793–1803.

[29] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham and B. Waters, "Sequential aggregate signatures and multisignatures without random oracles", *Advances in Cryptology – EUROCRYPT 2006*, Lecture Notes in Computer Science, 4004 (2006), 465–485.

[30] U. Maurer, "Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms", *Advances in Cryptology – CRYPTO '94*, Lecture Notes in Computer Science 839 (1994), 271–281.

[31] A. Muzereau, N. Smart and F. Vercauteren, "The equivalence between the DHP and DLP for elliptic curves used in practical applications", *LMS Journal of Computation and Mathematics*, 7 (2004), 50–72.

[32] D. Naccache, "Secure and *practical* identity-based encryption", *IET Information Security*, 1 (2007), 59–64.

[33] Y. Nogami, M. Akane, Y. Sakemi, H. Kato and Y. Morikawa, "Integer variable χ-based ate pairing", *Pairing-Based Cryptography – Pairing 2008*, Lecture Notes in Computer Science 5209 (2008), 178–191.

[34] J. Pollard, "Monte Carlo methods for index computation mod p", *Mathematics of Computation*, 32 (1978), 918–924.

[35] O. Schirokauer, "Discrete logarithms and local units", *Philosophical Transactions of the Royal Society: Physical and Engineering Sciences*, 345 (1993), 409–423.

[36] O. Schirokauer, "Using number fields to compute logarithms in finite fields", *Mathematics of Computation*, 69 (2000), 1267–1283.

[37] O. Schirokauer, "The number field sieve for integers of low hamming weight", Cryptology ePrint Archive Report 2006/107, 2006. Available from http://eprint.iacr.org/2006/107.

[38] M. Scott, "Computing the Tate pairing", *Topics in Cryptology – CT-RSA 2005*, Lecture Notes in Computer Science 3376 (2005), 293–304.

[39] M. Scott, "Implementing cryptographic pairings", *Pairing-Based Cryptography – Pairing 2007*, Lecture Notes in Computer Science 4575 (2007), 177–196.

[40] N. Smart and F. Vercauteren, "On computable isomorphisms in efficient asymmetric pairing-based systems", *Discrete Applied Mathematics*, 155 (2007), 538–547.

[41] B. Waters, "Efficient identity-based encryption without random oracles", *Advances in Cryptology – EUROCRYPT 2005*, Lecture Notes in Computer Science, 3494 (2005), 114–127.

[42] M. Zhao, S. Smith and D. Nicol, "Aggregated path authentication for efficient BGP security", *Proceedings of the 12th ACM Conference on Computer and Communications Security*, 2005, 128–138.

[43] M. Zhao, S. Smith and D. Nicol, "The performance impact of BGP security", *IEEE Security*, 19:6 (2005), 42–48.

Department of Combinatorics & Optimization, University of Waterloo, Waterloo, Ontario N2L 3G1 Canada
E-mail address: s2chatte@uwaterloo.ca

Department of Mathematics & Statistics, Auburn University, Auburn, Alabama 36849 USA
E-mail address: hankedr@auburn.edu

Department of Combinatorics & Optimization, University of Waterloo, Waterloo, Ontario N2L 3G1 Canada
E-mail address: edward.m.knapp@gmail.com

Department of Combinatorics & Optimization, University of Waterloo, Waterloo, Ontario N2L 3G1 Canada
E-mail address: ajmeneze@uwaterloo.ca