ON CRYPTOGRAPHIC PROTOCOLS EMPLOYING ASYMMETRIC PAIRINGS – THE ROLE OF Ψ REVISITED

SANJIT CHATTERJEE AND ALFRED MENEZES

ABSTRACT. Asymmetric pairings $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ for which an efficiently-computable isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ is known are called Type 2 pairings; if such an isomorphism ψ is not known then e is called a Type 3 pairing. Many cryptographic protocols in the asymmetric setting rely on the existence of ψ for their security reduction while some use it in the protocol itself. For these reasons, it is believed that some of these protocols cannot be implemented with Type 3 pairings, while for some the security reductions either cannot be transformed to the Type 3 setting or else require a stronger complexity assumption. Contrary to these widely held beliefs, we argue that Type 2 pairings are merely inefficient implementations of Type 3 pairings, and appear to offer no benefit for protocols based on asymmetric pairings from the point of view of functionality, security, and performance.

1. INTRODUCTION

Pairing-based cryptography, though proposed only at the turn of the century, has witnessed a tremendous growth. The successful application of pairings in the design of novel cryptographic protocols [28, 33, 9] and their potential use as a principal building block for many others fuelled this growth. The main thrust of research efforts in this area has been, and still is, in the development of new protocols. Simultaneously, substantial research has been carried out to find suitable pairings [32, 4] and efficient pairing algorithms [27, 30].

For three groups $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ of the same (prime) order, a pairing is a function $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ that is bilinear, non-degenerate, and efficiently computable. If $\mathbb{G}_1 = \mathbb{G}_2$ then the pairing is *symmetric* and following [24] we call it a Type 1 pairing. The pairing is *asymmetric* when $\mathbb{G}_1 \neq \mathbb{G}_2$. In the asymmetric setting, if there is an efficiently-computable isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ then e is called a Type 2 pairing. If no efficiently-computable isomorphism is known from $\mathbb{G}_2$ to $\mathbb{G}_1$ (or from $\mathbb{G}_1$ to $\mathbb{G}_2$) then we call it Type 3. Known examples of such pairings are the Weil and Tate pairings over suitable elliptic curve groups $\mathbb{G}_1$ and $\mathbb{G}_2$, and their modifications such as the ate pairing [27] and the R-ate pairing [30].[1] Since Type 1 pairings are quite restricted in terms of the choice of curves and are significantly slower than their asymmetric counterparts at higher security levels [26], they will not be considered in the remainder of this paper.

Current research [24] suggests that Type 3 is a better choice than Type 2 in terms of the size of elements in $\mathbb{G}_2$, the cost of performing group operations in $\mathbb{G}_2$, the cost of membership testing in $\mathbb{G}_2$, the feasibility of hashing into $\mathbb{G}_2$, and the cost of the pairing operation. However, these performance advantages do not immediately make Type 3 pairings an obvious choice for protocols described in the asymmetric setting. Cryptographic protocols are

Date: September 27, 2009.

[1] Cryptographically suitable pairings can also be defined from hyperelliptic curves and, more generally, from abelian varieties [23]. Since elliptic curve pairings are believed to offer superior performance, this paper will only be concerned with pairings derived from elliptic curves.

designed to realize some concrete functionality in a secure way. The question of security is *intrinsic* to that of functionality for such protocols. The question of efficiency enters only after issues of functionality and security have been settled. In the context of pairing-based protocols in the asymmetric setting, Type 3 can be considered to be a better choice only if it is possible to argue that whatever is achievable in terms of functionality and security in Type 2 can also be achieved in Type 3, and moreover if the overall performance of the Type 3 version of the protocol is at least as good (if not better) than its Type 2 counterpart.

Performance apart, the key difference between Type 2 and Type 3 is the existence or otherwise of the map ψ. While there exist some protocols in the asymmetric setting which do not require ψ to be efficiently computable, there are many others which do – either in the security proof or in the protocol itself. For example, consider the very first protocol described in the asymmetric setting – the Boneh-Lynn-Shacham (BLS) short signature scheme [11]. In [11] the authors observed that the efficiently-computable isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ is essential for the security of the protocol and can be avoided only at the cost of making a stronger complexity assumption. The same argument was later echoed by Shacham in his PhD Thesis [34] which asserts that "the map $[\psi]$ isn't merely a proof artifact". These observations were likely instrumental in causing much of the subsequent work in pairing-based cryptography to consider the Type 2 setting as the *natural* choice either when proposing a new protocol or when modifying a protocol from the symmetric to the asymmetric setting. While most of these protocols need ψ in the security argument only, some do use it in the protocol itself. The use of ψ in a protocol might further be construed as evidence that the map ψ cannot possibly be avoided altogether even if one is prepared to make stronger complexity assumptions.

Galbraith, Paterson and Smart in their timely work "Pairings for cryptographers" [24] provided an excellent exposition of what is achievable and what isn't when a particular type of pairing is employed. They too comment that for many pairing-based primitives, the "security proof does not apply if the cryptosystem is implemented using pairings of the third type" (i.e., Type 3). Since Type 3 offers better performance and flexibility they conclude that it would be desirable if the protocol designers prove the security of their protocol in the Type 3 setting.

While Type 3 might allow better performance for some protocols, the map ψ appears to be necessary for some of the known security reductions to go through. One possible way to resolve this dichotomy is to use a relativized assumption as introduced by Smart and Vercauteren [36]. Using such a notion it is assumed that an underlying hard problem remains hard even when the adversary is given oracle access to ψ. Such an oracle access allows a security reduction to go through in the Type 3 setting even when one cannot efficiently compute ψ. Smart and Vercauteren discuss the security of the Boneh-Franklin identity-based encryption (BF-IBE) scheme and the BLS signature scheme under this kind of relativized assumption. Their approach was further pursued by Chen, Cheng and Smart [18] to prove the security of some identity-based key agreement protocols in the Type 3 setting.

In [16] Chatterjee et al. took a closer look at the security and efficiency aspects of two signature schemes and two aggregate signature schemes when implemented with Barreto-Naehrig (BN) elliptic curves [4]. One of the contributions of that work is to establish that there is no security (or performance) benefit to be gained by using a Type 2 pairing instead of Type 3 for the particular schemes under consideration. For example, compared to Type 2, the BLS signature scheme in Type 3 does *not* depend on a stronger complexity assumption as was stated in [11], nor is it necessary to use a relativized assumption as suggested in [36]. This motivated us to further investigate the exact role played by ψ in pairing-based

protocols and in their security arguments. The current work takes the findings of [16] as its starting point and can be seen as a sequel of it in a broader canvas.

Specifically, we find it relevant to raise the following questions. Can any cryptographic protocol originally described in the Type 2 setting be transformed to the Type 3 setting? What about the security of the transformed protocol – is it equivalent to the original one? Is there any performance benefit to be accrued by working in the Type 3 setting after such a transformation?

Our study indicates that given any protocol, Protocol-2, described using a Type-2 pairing, and a security proof for Protocol-2 with respect to some problem $\mathcal{P}$-2, there is a natural transformation of Protocol-2 to a Protocol-3 that uses a Type-3 pairing, a natural transformation of $\mathcal{P}$-2 to $\mathcal{P}$-3, and a natural transformation of the security proof to one for Protocol-3 with respect to $\mathcal{P}$-3. Moreover, Protocol-3 is at least as efficient as Protocol-2, and $\mathcal{P}$-3 is equally as hard as $\mathcal{P}$-2 (for appropriately chosen parameters). In other words, ψ does not play any cryptographically significant role and hence there is no reason to use Protocol-2 instead of Protocol-3.

The remainder of the paper is organized as follows. In §2, we compare the performance of Type 2 and Type 3 pairings derived from elliptic curves having even embedding degree. The complexity assumptions in the asymmetric setting are reviewed in §3 and we demonstrate that for each complexity assumption in Type 2, there is a natural counterpart in Type 3 such that the two problems are equivalent when parameters are chosen in an appropriate way. §4 is devoted to existing protocols in the asymmetric setting. We show how some known protocols in the Type 2 setting can be transformed into Type 3 without affecting the functionality or security, and moreover it is sometimes possible to obtain better performance. These observations are extended in §5 where we provide general guidelines on how to transform a given protocol as well as its security argument from Type 2 to Type 3. Finally, we conclude in §6 with some open problems which we think will shed further light on the role of ψ in the context of cryptographic protocols in the asymmetric setting.

2. Asymmetric Pairings

Let $\mathbb{F}_q$ be a finite field of characteristic $p \geq 5$, and let E be an ordinary elliptic curve defined over $\mathbb{F}_q$. Let n be a prime divisor of $\#E(\mathbb{F}_q)$ satisfying $\gcd(n, q) = 1$, and let k (the embedding degree) be the smallest positive integer such that $n \mid q^k - 1$. We will assume that $k > 1$, whence $E[n] \subseteq E(\mathbb{F}_{q^k})$ where $E[n]$ denotes the n-torsion group of E. We will further assume that $n^3 \nmid \#E(\mathbb{F}_{q^k})$. Let $\mathbb{G}_T$ be the order-n subgroup of $\mathbb{F}_{q^k}^*$. The (full) Tate pairing is a non-degenerate bilinear function $\hat{e} : E[n] \times E[n] \to \mathbb{G}_T$ and can be defined as follows:

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

where $R \in E(\mathbb{F}_{q^k})$ with $R \notin \{\infty, P, -Q, P - Q\}$, and where the *Miller function* $f_{n,P}$ is a function whose only zeros and poles in E are a zero of order n at P and a pole of order n at ∞. For cryptographic applications, one generally restricts the domain of $\hat{e}$ to a product of two order-n subgroups $\mathbb{G}_1$ and $\mathbb{G}_2$ of $E[n]$. The first group $\mathbb{G}_1$ is taken to be $E(\mathbb{F}_q)[n]$, and any other order-n subgroup can be selected for the second group $\mathbb{G}_2$. Then the definition of $\hat{e}$ simplifies to $\hat{e}(P, Q) = (f_{n,P}(Q))^{(q^k-1)/n}$ for all $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$ [3]. Moreover, one assumes k to be even because then the 'denominator elimination' [3] speedup is applicable for the Type 3 pairings defined next.

2.1. **Type 3 pairings.** Following [24], we denote by D the CM discriminant of E and set

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

and $d = k/e$. For example, BN curves [4] have $k = 12$, $e = 6$ and $d = 2$, whereas MNT curves [32] have $k = 6$, $e = 2$ and $d = 3$. Now, E has a unique degree-e twist $\tilde{E}$ defined over $\mathbb{F}_{q^d}$ such that $n \mid \#\tilde{E}(\mathbb{F}_{q^d})$ [27]. Let $\tilde{P}_2 \in \tilde{E}(\mathbb{F}_{q^d})$ be a point of order n, and let $\tilde{\mathbb{G}}_2 = \langle \tilde{P}_2 \rangle$. Then there is a monomorphism $\phi : \tilde{\mathbb{G}}_2 \to E(\mathbb{F}_{q^k})$ such that $P_2 = \phi(\tilde{P}_2) \notin \mathbb{G}_1$. The group $\mathbb{G}_2 = \langle P_2 \rangle$ is the Trace-0 subgroup of $E[n]$, so named because it consists of all points $P \in E[n]$ for which $\mathrm{Tr}(P) = \sum_{i=0}^{k-1} \pi^i(P) = \infty$, where π denotes the q-th power Frobenius. The monomorphism ϕ can be defined so that $\phi : \tilde{\mathbb{G}}_2 \to \mathbb{G}_2$ can be efficiently computed in both directions; therefore we can identify $\tilde{\mathbb{G}}_2$ and $\mathbb{G}_2$, and consequently $\mathbb{G}_2$ can be viewed as having coordinates in $\mathbb{F}_{q^d}$ (instead of in the larger field $\mathbb{F}_{q^k}$). The restriction of $\hat{e}$ to $\mathbb{G}_1 \times \mathbb{G}_2$ gives a pairing $t_n : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that is of the Type 3 variety because no efficiently-computable isomorphism is known from $\mathbb{G}_2$ to $\mathbb{G}_1$. Several Type 3 pairings that are faster to evaluate have been discovered. Among these are the ate pairing $a_n : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ [27] and the R-ate pairing $R_n : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ [30]. In particular, the R-ate pairing with BN curves is the fastest pairing presently known for the 128-bit security level. Like the ate pairing, it has the property that there is a fixed integer N (with $n \nmid N$) such that $R_n(P,Q) = \hat{e}(Q,P)^N$ for all $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$.

In the remainder of the paper we will consider the R-ate pairing and denote it by e_3.

2.2. **Type 2 pairings.** If $P_2' \in E[n]$ with $P_2' \notin \mathbb{G}_1$ and $P_2' \notin \mathbb{G}_2$, then $\mathbb{G}_2' = \langle P_2' \rangle$ is an order-n subgroup of $E(\mathbb{F}_{q^k})$ with $\mathbb{G}_2' \neq \mathbb{G}_1$ and $\mathbb{G}_2' \neq \mathbb{G}_2$. Bilinear pairings $e : \mathbb{G}_1 \times \mathbb{G}_2' \to \mathbb{G}_T$ are of the Type 2 variety because the map Tr is an efficiently-computable isomorphism from $\mathbb{G}_2'$ to $\mathbb{G}_1$. These pairings have the property that hashing onto $\mathbb{G}_2'$ is not feasible (other than by multiplying P_2' by a randomly selected integer).

Consider now the Type 2 pairing $e_2 : \mathbb{G}_1 \times \mathbb{G}_2' \to \mathbb{G}_T$ defined by $e_2(P,Q) = \hat{e}(Q,P)^{2N}$. Notice that this choice of pairing is without much loss of generality since in any cryptographic application of the pairing it makes no difference if the pairing is replaced by its $(2N)$-th power. As first shown in [29], the computation of e_2 is easily reduced to the task of computing the R-ate pairing.

Lemma 1 ([29])**.** *Let $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2'$. Then $e_2(P,Q) = e_3(P,\hat{Q})$, where $\hat{Q} = Q - \pi^f(Q)$ and $f = k/2$.*

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

$$\begin{aligned} e_2(P,Q) &= \hat{e}(Q,P)^{2N} \\ &= \hat{e}(2Q,P)^N \\ &= \hat{e}(Q + \hat{Q} + \pi^f(Q), P)^N \\ &= \hat{e}(\hat{Q},P)^N \cdot \hat{e}(Q + \pi^f(Q), P)^N \\ &= e_3(P,\hat{Q}), \end{aligned}$$

since $Q + \pi^f(Q) \in E(\mathbb{F}_{q^f})$ whence $\hat{e}(Q + \pi^f(Q), P) = 1$ [22, Lemma IX.8]. □

2.3. **Comparing the performance of Type 2 and Type 3 pairings.** Since points in $\mathbb{G}_2'$ have coordinates in $\mathbb{F}_{q^k}$ whereas points in $\mathbb{G}_2$ have coordinates in $\mathbb{F}_{q^d}$, it would appear that the ratio of the bitlengths of points in $\mathbb{G}_2'$ and $\mathbb{G}_2$ is k/d. Similarly, the ratio of the costs of addition in $\mathbb{G}_2'$ and $\mathbb{G}_2$ can be expected to be k^2/d^2 bit operations (using naive methods for extension field arithmetic). These ratios are given in Table 3 of [24]; see also Table 5 of [18]. However, as observed in [16], points in $\mathbb{G}_2'$ have a shorter representation which we describe next. We emphasize that this representation can be used for *all* order-n subgroups $\mathbb{G}_2'$ of $E[n]$ different from $\mathbb{G}_1$ and $\mathbb{G}_2$.

Let P_2' be an arbitrary point from $E[n] \setminus (\mathbb{G}_1 \cup \mathbb{G}_2)$, and set $\mathbb{G}_2' = \langle P_2' \rangle$. Define $P_1 = \frac{1}{k}\mathrm{Tr}(P_2')$ so that the map

$$\psi : \mathbb{G}_2' \to \mathbb{G}_1, \qquad Q \mapsto \frac{1}{k}\mathrm{Tr}(Q) \tag{3}$$

is an efficiently-computable isomorphism with $\psi(P_2') = P_1$. Finally, set $P_2 = c^{-1}(P_2' - P_1)$ for an arbitrary integer $c \in \mathbb{Z}_n^*$. Then $P_2 \in \mathbb{G}_2$ and the map

$$\rho : \mathbb{G}_2' \to \mathbb{G}_2, \qquad Q \mapsto Q - \psi(Q) \tag{4}$$

is an efficiently-computable isomorphism with $\rho(P_2') = cP_2$.

Now, given a point $Q \in \mathbb{G}_2'$, one can efficiently determine the unique points $Q_1 \in \mathbb{G}_1$ 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 $D(Q) = (\psi(Q), \rho(Q))$, 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. Hence, without loss of generality, points $Q \in \mathbb{G}_2'$ can be represented by a pair of points (Q_1, Q_2) with $Q_1 \in \mathbb{G}_1$ and $Q_2 \in \mathbb{G}_2$. Note that arithmetic in $\mathbb{G}_2'$ with this representation is component-wise. Thus the ratio of the bitlengths of points in $\mathbb{G}_2'$ and $\mathbb{G}_2$ is in fact $(d+1)/d$, while the ratio of the costs of addition in $\mathbb{G}_2'$ and $\mathbb{G}_2$ is $(d^2+1)/d^2$.

Table 2 of [16] lists the costs of performing basic operations in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_2'$ for a particular BN curve. The table confirms the expectation that basic operations in $\mathbb{G}_2'$ are only marginally more expensive than the operations in $\mathbb{G}_2$. One exception is that testing membership in $\mathbb{G}_2'$ is several times more expensive than membership testing in $\mathbb{G}_2$ since the former requires two pairing operations.

In summary, we have shown that while the basic operations in the Type 2 group $\mathbb{G}_2'$ are indeed more expensive than in the Type 3 group $\mathbb{G}_2$, the differences are not as high as previously reported. The same is true for the bitlengths of points in $\mathbb{G}_2'$ versus $\mathbb{G}_2$. The remainder of the paper will compare the security and efficiency of protocols that use the pairings e_2 or e_3. For consistency with the literature on pairing-based protocols, we will use multiplicative notation for elements of $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_2'$. In particular, the generators of these groups will be denoted by g_1, g_2 and g_2', and the identity element will be denoted by 1.

3. Hardness Assumptions in the Asymmetric Setting

Security of a pairing-based protocol is based on some hard problem in the respective pairing groups. The standard practice is to argue the security of the protocol in terms of a reduction from the hard problem to breaking the protocol in an appropriate security model. Suppose that we have a protocol, Protocol-2, described in the Type 2 setting whose security is based on some hard problem $\mathcal{P}$-2 in that setting. Also suppose that we have some means of obtaining a version of Protocol-2, say Protocol-3, in the Type 3 setting which achieves the same functionality as Protocol-2. Simultaneously, we would like the assurance that Protocol-3 is at least as secure as Protocol-2. One way to achieve this is to define a version of problem $\mathcal{P}$-2, say $\mathcal{P}$-3 in the Type 3 setting, and then argue the security of Protocol-3

based on $\mathcal{P}$-3. Now, if $\mathcal{P}$-3 can be shown to be at least as hard as $\mathcal{P}$-2, then the assurance provided by the reduction for Protocol-3 is at least as high as the assurance provided by the reduction for Protocol-2. If $\mathcal{P}$-2 and $\mathcal{P}$-3 are computationally equivalent, then the assurances provided by the reductions are the same. Hence the security of Protocol-2 and Protocol-3 can be compared if one can find a way to define $\mathcal{P}$-3 so that it is at least as hard as $\mathcal{P}$-2.

For example, as shown in [16], security of BLS in the Type 2 setting is based on the co-DHP problem (compute h^z given $h \in \mathbb{G}_1$ and ${g'_2}^z \in \mathbb{G}'_2$), whereas that of BLS in the Type 3 setting is based on the co-DHP* problem (compute h^z given $h, g_1^z \in \mathbb{G}_1$ and $g_2^z \in \mathbb{G}_2$). Furthermore, these two problems are equivalent if the generators g_1, g_2, g'_2 are appropriately chosen. We next show equivalence of the Type 2 and Type 3 variants of the bilinear Diffie-Hellman (BDH) problem.

3.1. **The bilinear Diffie-Hellman assumption.** The bilinear Diffie-Hellman problem was originally defined in the symmetric setting [9] and later extended to the asymmetric setting [25, 14]. It is possible to define several versions of the problem in the asymmetric setting – see e.g., [36, 13] and the note on variants of BDH below. We consider the following version of the problem in Type 2 used by Galindo in [25] and which is the same as the problem discussed in [14] when specialized to Type 2.

Definition 1 (Bilinear Diffie-Hellman Problem in Type 2 (BDH-2))**.** Given $g_1^\alpha \in \mathbb{G}_1$ and ${g'_2}^\beta, {g'_2}^\gamma \in \mathbb{G}'_2$ for $\alpha, \beta, \gamma \in_R \mathbb{Z}_n$, the BDH-2 problem is to compute the Type 2 pairing value $e_2(g_1, g'_2)^{\alpha\beta\gamma}$. The BDH-2 assumption asserts that the BDH-2 problem is hard. The decisional version DBDH-2 of the problem is to decide, given $(g_1^\alpha, {g'_2}^\beta, {g'_2}^\gamma, Z) \in \mathbb{G}_1 \times \mathbb{G}'_2 \times \mathbb{G}'_2 \times \mathbb{G}_T$, whether or not $Z = e_2(g_1, g'_2)^{\alpha\beta\gamma}$. The DBDH-2 assumption is that the DBDH-2 problem is hard.

We define a version of the BDH-2 problem in Type 3 as follows.

Definition 2 (Bilinear Diffie-Hellman Problem in Type 3 (BDH-3))**.** Given $g_1^\alpha, g_1^\beta, g_1^\gamma \in \mathbb{G}_1$ and $g_2^\beta, g_2^\gamma \in \mathbb{G}_2$ for $\alpha, \beta, \gamma \in_R \mathbb{Z}_n$, the BDH-3 problem is to compute the Type 3 pairing value $e_3(g_1, g_2)^{\alpha\beta\gamma}$. The decisional version DBDH-3 of the problem is defined analogously to DBDH-2. The BDH-3 (resp. DBDH-3) assumption 3 asserts that the BDH-3 (resp. DBDH-3) problem is hard.

It might appear at first sight that the BDH-3 assumption is stronger than BDH-2 as two extra elements, namely $g_1^\beta, g_1^\gamma \in \mathbb{G}_1$, are provided as input to the BDH-3 problem. However, note that one can easily compute these values in BDH-2 by virtue of ψ. Hence we consider these two problems as natural counterparts and in fact they are equivalent as we show in Lemma 3. Essentially the same argument also applies for the decisional versions DBDH-2 and DBDH-3.

Lemma 2. *Let g_1, g'_2, g_2 be generators of $\mathbb{G}_1, \mathbb{G}'_2, \mathbb{G}_2$ with $g_1 = \psi(g'_2)$ and $g_2 = (\rho(g'_2))^{1/c}$ for some $c \in \mathbb{Z}_n^*$. Then $e_2(g_1, g'_2) = e_3(g_1, g_2)^{2c}$.*

Proof. First note that $g'_2 = g_1 g_2^c$. Hence, by Lemma 1, we have

$$e_2(g_1, g'_2) = e_3\left(g_1, \frac{g'_2}{\pi^f(g'_2)}\right) = e_3\left(g_1, \frac{g_1 g_2^c}{g_1(\pi^f(g_2))^c}\right).$$

The result then follows if we can establish that $\pi^f(g_2) = g_2^{-1}$.

Note that $\mathrm{Tr}(\pi(g_2)) = \pi(\mathrm{Tr}(g_2)) = \pi(1) = 1$, whence $\pi(g_2) \in \mathbb{G}_2$. Hence we can write $\pi(g_2) = g_2^r$ for some $r \in [1, n-1]$. Since $\pi^k(g_2) = g_2$, we have ${g_2^r}^k = g_2$ and hence

$$g_2^{(r^f-1)(r^f+1)} = 1. \tag{5}$$

Now, $n \nmid r^f - 1$ since $\pi^f(g_2) \neq g_2$. Hence (5) implies that $g_2^{r^f+1} = 1$, so $\pi^f(g_2) = g_2^{-1}$. □

Lemma 3. *Let g_1, g_2', g_2 be generators of $\mathbb{G}_1, \mathbb{G}_2', \mathbb{G}_2$ with $g_1 = \psi(g_2')$ and $g_2 = (\rho(g_2'))^{1/c}$ for some $c \in \mathbb{Z}_n^*$. Then BDH-2 is equivalent to BDH-3.*

Proof. Given a BDH-2 problem instance $(g_1^\alpha, {g_2'}^\beta, {g_2'}^\gamma)$, we apply the function $\rho : \mathbb{G}_2' \to \mathbb{G}_2$ to obtain $g_2^\beta = (\rho({g_2'}^\beta))^{1/c}$ and $g_2^\gamma = (\rho({g_2'}^\gamma))^{1/c}$ and apply $\psi : \mathbb{G}_2' \to \mathbb{G}_1$ to obtain $g_1^\beta = \psi({g_2'}^\beta)$ and $g_1^\gamma = \psi({g_2'}^\gamma)$. The resulting BDH-3 problem instance $(g_1^\alpha, g_1^\beta, g_1^\gamma, g_2^\beta, g_2^\gamma)$ is given to the BDH-3 solver which returns $e_3(g_1, g_2)^{\alpha\beta\gamma}$ from which the solution $e_2(g_1, g_2')^{\alpha\beta\gamma}$ of the original BDH-2 problem is easily obtained by Lemma 2. This establishes that BDH-2 $\leq$ BDH-3.

Conversely, given a BDH-3 problem instance $(g_1^\alpha, g_1^\beta, g_1^\gamma, g_2^\beta, g_2^\gamma)$, we compute ${g_2'}^\beta = g_1^\beta (g_2^\beta)^c$ and ${g_2'}^\gamma = g_1^\gamma (g_2^\gamma)^c$. The resulting BDH-2 problem instance $(g_1^\alpha, {g_2'}^\beta, {g_2'}^\gamma)$ is given to the BDH-2 solver which returns $e_2(g_1, g_2')^{\alpha\beta\gamma}$. Thereafter, the solution $e_3(g_1, g_2)^{\alpha\beta\gamma}$ of the original BDH-3 problem is easily obtained showing that BDH-3 $\leq$ BDH-2. □

The formal equivalence between (D)BDH-2 and (D)BDH-3 is established under the condition that the parameter c is known. No such equivalence is known if c is unknown, nor is there any indication that one problem is weaker than the other. Note that BDH-2 can be solved either by solving the Diffie-Hellman problem (DHP) in $\mathbb{G}_1$ or $\mathbb{G}_2'$ or by solving co-DHP. Similarly, BDH-3 can be solved either by solving DHP in $\mathbb{G}_1$ or $\mathbb{G}_2$ or by solving co-DHP*. Currently there is no evidence to suggest that DHP is any easier in $\mathbb{G}_2$ than in $\mathbb{G}_2'$, or, for that matter, co-DHP* is any easier than co-DHP (see §2.3 of [16] for a discussion on the relationship between co-DHP and co-DHP*).

Variants of BDH. As already noted, it is possible to formulate different versions of BDH in the asymmetric setting. Some of these variants have been used to argue the security of some existing protocols. For example, Smart and Vercauteren [36] discuss several such variants (including the relativized versions) and their relationships and show that the security of different versions of BF-IBE relies on different versions of the BDH problem. Boyen provides a general statement of the problem (called BDH') in [13] for all known pairing types, the earliest mention of which can be traced back to the work of Boyen, Mei and Waters [14].[2] In Type 2 the problem is to compute $e_2(g_1, g_2')^{\alpha\beta\gamma}$ given $g_1^\alpha, g_1^\beta \in \mathbb{G}_1$ and ${g_2'}^\beta, {g_2'}^\gamma \in \mathbb{G}_2'$; we call this problem BDH-2b. Clearly BDH-2b and BDH-2 are equivalent. The analogous problem in Type 3, which we call BDH-3b, is obtained by replacing elements of $\mathbb{G}_2'$ by elements of $\mathbb{G}_2$ and the task is to compute $e_3(g_1, g_2)^{\alpha\beta\gamma}$. However, it is not known whether BDH-3b is equivalent to BDH-3 (or for that matter to BDH-2 or BDH-2b). Still another variant in Type 3, which we call BDH-3c, is to compute $e_3(g_1, g_2)^{\alpha\beta\gamma}$ given $g_1^\alpha \in \mathbb{G}_1$ and $g_2^\beta, g_2^\gamma \in \mathbb{G}_2$. BDH-3c is attributed to Galbraith in [13]. It is easy to see that BDH-3 $\leq$ BDH-3b $\leq$ BDH-3c, but currently we do not know anything in the reverse direction.

3.2. **Other assumptions.** A large array of complexity assumptions have been proposed so far in pairing groups; see [13] for a listing of such assumptions. Most of these assumptions come in two flavors – computational and decisional as in the case of BDH. Many of these assumptions were initially introduced in the symmetric setting to be generalized later in the asymmetric setting. As we have noted, some authors [36, 18, 17] also used the notion

[2]The statement allows uniform description of the problem across different settings. However, that does not imply that the problem remains equivalent in different settings. In other words, though the "statement complexity" remains the same the computational complexity may be quite different!

of relativized assumption in the Type 3 setting – assuming that the problem remains hard even when the adversary is given oracle access to ψ.

Their apparent diversity notwithstanding, the complexity assumptions in pairing groups can be broadly classified into two categories.

(i) Assumptions where the problem does not explicitly involve any element from $\mathbb{G}_T$ – examples are the co-DHP, Linear, Strong DH, Hidden SDH, and Poly-SDH assumptions.
(ii) Assumptions where the problem involves a pairing computation and hence element of $\mathbb{G}_T$ – examples are BDH, Bilinear DH Inversion (BDHI), and Bilinear DH Exponent (BDHE) assumptions.

Given a hardness assumption in the Type 2 setting, our primary concern here is to formulate its natural counterpart in Type 3. Note that if one is given some element $g_2'^{z} \in \mathbb{G}_2'$, then one can easily obtain $g_1^z = \psi(g_2'^{z})$ without knowing z. So if the input to problem $\mathcal{P}$-2 includes $g_2'^{z}$ then that implicitly implies that $g_1^z \in \mathbb{G}_1$ is also part of the input. However, that is not the case for Type 3 as ψ is not known in that setting. Hence, as in the case of BDH and co-DHP, we insist that both g_1^z and g_2^z be included in the input to $\mathcal{P}$-3 in order to make it the natural counterpart of $\mathcal{P}$-2 in the Type 3 setting. For an assumption in Category (i) above, we can then use the argument of Lemma 2 of [16] (which shows the equivalence of co-DHP and co-DHP*) to show that $\mathcal{P}$-3 is equivalent to $\mathcal{P}$-2, while the argument put forth in Lemma 3 above for the case of BDH can be readily adapted to show equivalence between $\mathcal{P}$-2 and $\mathcal{P}$-3 for an assumption in Category (ii).

Remark 1. For some cryptographic protocols in Type 3, it might be possible to drop one or more elements of $\mathbb{G}_1$ from the problem statement of $\mathcal{P}$-3 without affecting the reduction. For example, BF-IBE-3 can be proven secure under BDH-3b (see §4.1). However, we do not know any protocol that can be proven secure under the seemingly weaker assumption BDH-3c. Note that the input to BDH-3c includes only $g_1^\alpha \in \mathbb{G}_1$ along with $g_2^\beta, g_2^\gamma \in \mathbb{G}_2$, whereas all known security reductions based on BDH in the asymmetric setting require at least one of g_1^β or g_1^γ as part of the problem input. We note that the situation is similar for the "weaker statement" of the Linear assumption in Type 3 as stated in [13] (also attributed to Galbraith). In fact an interesting open question is to what extent one can prune $\mathcal{P}$-3 and still use it in the security reduction of a natural cryptographic problem.

4. Protocols in the Asymmetric Setting

We revisit some existing pairing-based protocols in the asymmetric setting. Some of these protocols employ the isomorphism ψ in the protocol itself and some others only in the security reduction. The purpose of this investigation is twofold – to determine the exact role played by ψ in the functionality and security of these protocols and then to investigate whether it is possible to avoid the use of ψ altogether. We begin in §4.1 with the most famous protocol in the pairing-based setting – the identity-based encryption scheme of Boneh and Franklin [9]. This protocol was originally described in the symmetric setting but can also be implemented in the asymmetric setting [25, 36]. In contrast to previous findings, our study indicates that Type 3 is indeed a better choice than Type 2 for BF-IBE taking into account functionality, security and efficiency. We then show in §4.2 and §4.3 that this observation extends to some other known protocols where ψ is used either in the protocol and/or in the security reduction.

4.1. **Boneh-Franklin IBE.**

For simplicity we focus on the basic version of the protocol (called BasicIdent in [9]). The same arguments apply to the full version. While it is known

that the protocol can be implemented in both Type 2 and Type 3 (and it is possible to better optimize the protocol in Type 3), an earlier work [36] gives the impression that the security in the Type 3 setting might depend on a stronger complexity assumption.

BF-IBE-2. The master secret of the key generation center (KGC) is $x \in_R \mathbb{Z}_n$ and the corresponding public key is $g_{\mathsf{pub}} = {g'_2}^x \in \mathbb{G}'_2$. Given a user identity $\mathsf{id} \in \{0,1\}^*$, the public key of the user is $h_{\mathsf{id}} = H_1(\mathsf{id}) \in \mathbb{G}_1$, where $H_1 : \{0,1\}^* \to \mathbb{G}_1$ is a publicly computable hash function. The corresponding private key is $d_{\mathsf{id}} = h_{\mathsf{id}}^x$. To encrypt a message $M \in \{0,1\}^n$ a sender chooses $r \in_R \mathbb{Z}_n$ and sends $\langle {g'_2}^r, M \oplus H_2(e_2(h_{\mathsf{id}}, g_{\mathsf{pub}})^r)\rangle$ where $H_2 : \mathbb{G}_T \to \{0,1\}^n$ is another publicly computable hash function. The receiver computes $H_2(e_2(d_{\mathsf{id}}, {g'_2}^r))$ and then xors it with the second component of the ciphertext to obtain M. The decryption process succeeds since $e_2(d_{\mathsf{id}}, {g'_2}^r) = e_2(h_{\mathsf{id}}^x, g'_2)^r = e_2(h_{\mathsf{id}}, g_{\mathsf{pub}})^r$. The security of BF-IBE-2 is argued by a reduction from BDH-2 to the breaking of BF-IBE-2.

BF-IBE-3. The above scheme can be directly implemented in Type 3 – the KGC's public key will be $g_{\mathsf{pub}} = g_2^x \in \mathbb{G}_2$ and similarly the ephemeral key in the ciphertext will be $g_2^r \in \mathbb{G}_2$. The security of BF-IBE-3 is argued by a reduction from BDH-3b to the breaking of BF-IBE-3.

A variant of BF-IBE in Type 3 can be obtained by hashing the identities into $\mathbb{G}_2$. We call this variant BF-IBE-3b which gives a smaller ciphertext overhead. Such an optimization is not possible for BF-IBE-2 as we do not know how to hash into $\mathbb{G}'_2$.[3]

BF-IBE-3b. The master public key of the key generation center is now $g_{\mathsf{pub}} = g_1^x \in \mathbb{G}_1$ while $H_1 : \{0,1\}^* \to \mathbb{G}_2$ and so both $h_{\mathsf{id}} = H_1(\mathsf{id})$ and d_{id} are in $\mathbb{G}_2$. To encrypt a message $M \in \{0,1\}^n$ a sender chooses $r_R \in \mathbb{Z}_n$ and sends $\langle g_1^r, M \oplus H_2(e(g_{\mathsf{pub}}, h_{\mathsf{id}})^r)\rangle$. The receiver computes $H_2(e(g_1^r, d_{\mathsf{id}}))$ and then xors it with the second component of the ciphertext to obtain M. The security of BF-IBE-3b is argued by a reduction from BDH-3b.

Note that the ciphertext overhead in BF-IBE-3b is one element of $\mathbb{G}_1$ (namely g_1^r), while that in BF-IBE-2 is one element of $\mathbb{G}'_2$. Furthermore, exponentiation in $\mathbb{G}_1$ is faster than exponentiation in $\mathbb{G}'_2$, and hence BF-IBE-3b is a better choice as far as performance is concerned.

Smart and Vercauteren [36] observed that the security of BF-IBE-3 can be reduced to either BDH-3c with oracle access to ψ or to BDH-3b without such oracle access. In the first case one does not know how to simulate the oracle and in the second case they consider the problem (which they call coBDH$_{1,2}$) to be "somewhat unnatural". Based on these observations they conclude that one should use a pairing with an efficiently-computable isomorphism, i.e., Type 2 for BF-IBE. However, as we have already noted, BDH-3b is at least as hard as BDH-2 and so Type 3 is overall a better choice for BF-IBE.

4.2. **Protocols employing ψ.** Some protocols in the asymmetric setting employ ψ in the protocol itself. For example consider the verifiably encrypted signature scheme and ring signature scheme of Boneh, Gentry, Lynn and Shacham (BGLS) [10] and the group signature scheme with verifier-local revocation of Boneh and Shacham [12]. Here we describe the ring signature scheme and its security argument in the original Type 2 setting and then show how one can easily modify both to allow working in the Type 3 setting. A similar argument applies to the verifiably encrypted signature scheme. As observed in [36], the

[3]Galindo in [25] assumed the existence of such a hash function. However, it is easy to either modify the protocol in [25] which then corresponds to BF-IBE-2 above or to change the security assumption to BDH-3b.

Boneh-Shacham group signature scheme cannot be implemented in either Type 2 or Type 3 (cf. footnote 4).

A ring signature on a message is constructed using the public keys of a set of users U and the private key of a single user $u \in U$. The verification process gives the assurance that the signature was produced using one of the private keys of U but the verifier should not be able to determine which particular $u \in U$ signed the message. This is the so-called *signer-ambiguity* property of a ring signature.

BGLS-2 ring signature scheme. Let $e_2 : \mathbb{G}_1 \times \mathbb{G}_2' \rightarrow \mathbb{G}_T$ be a Type 2 pairing and $\psi : \mathbb{G}_2' \rightarrow \mathbb{G}_1$ be an efficiently-computable isomorphism with $\psi(g_2') = g_1$. The signature scheme employs a hash function $H : \{0,\ 1\}^* \rightarrow \mathbb{G}_1$.

An individual signer's private key is an integer $x \in_R \mathbb{Z}_n$ and her public key is $X = (g_2')^x$. Given the public keys $X_1, \ldots, X_\ell$ of a set U of ℓ users, a message $M \in \{0,1\}^*$, and a private key x_s corresponding to one of the users in U whose public key is X_s, the signer chooses $a_i \in_R \mathbb{Z}_n$ for each $i \neq s$ and computes $h = H(M)$ and $\sigma_s = (h/\psi(\prod_{i \neq s} X_i^{a_i}))^{1/x_s}$. She also computes $\sigma_i = g_1^{a_i}$ for each $i \neq s$. The ring signature on M is $\sigma = \langle \sigma_1, \ldots, \sigma_\ell \rangle$. Given (M, σ), the verifier computes $h = H(M)$ and accepts if and only if $e_2(h, g_2') = \prod_{i=1}^{\ell} e_2(\sigma_i, X_i)$.

The correctness of the verification algorithm can be easily checked using the bilinear property. We show this for $\ell = 2$ and $s = 1$. For $h = H(M)$, we have $\sigma_1 = (h/\psi(X_2^{a_2}))^{1/x_1} = (h/g_1^{a_2 x_2})^{1/x_1}$ and $\sigma_2 = g_1^{a_2}$. Thus

$$e_2(\sigma_1, X_1)e_2(\sigma_2, X_2) = e_2((hg_1^{-a_2 x_2})^{1/x_1},\ {g_2'}^{x_1})e_2(g_1^{a_2}, {g_2'}^{x_2}) = e_2(h, g_2').$$

There are two aspects of security for a ring signature scheme – signer ambiguity and unforgeability. Using a probabilistic argument it has been shown in [10] that the signer's identity is unconditionally protected. Unforgeability is based on the following complexity assumption: given g_1^{ab} and ${g_2'}^b$ it is hard to compute g_1^a. We call it the co-divisible computational Diffie-Hellman (co-DCDH) assumption – see [2] for the statement of DCDH in the Diffie-Hellman setting. Here we briefly reproduce the original security argument.

Security argument. For simplicity, we assume $\ell = 2$; the argument can be easily extended to any $\ell > 2$. Given g_1^{ab} and ${g_2'}^b$, the challenger $\mathcal{S}$ sets $X_1 = {g_2'}^b$ and $X_2 = ({g_2'}^b)^{x_2}$ for some $x_2 \in_R \mathbb{Z}_n$, and gives X_1, X_2 to the adversary $\mathcal{A}$. On receiving a hash query, $\mathcal{S}$ flips a coin which shows 0 with some probability p and 1 otherwise; $\mathcal{S}$ selects $r \in_R \mathbb{Z}_n$ and returns $(g_1^{ab})^r$ if the outcome of the coin flip is 0, and returns $\psi({g_2'}^b)^r$ in the other case. On receiving a signing query for a message M, $\mathcal{S}$ aborts if the outcome of the coin flip on the hash query for M was 0. Otherwise, $H(M) = \psi({g_2'}^b)^r$ where r was chosen by $\mathcal{S}$, and $\mathcal{S}$ returns the signature $\sigma = \langle g_1^{r - a_2 x_2},\ g_1^{a_2} \rangle$ for some $a_2 \in_R \mathbb{Z}_n$. When $\mathcal{A}$ outputs a valid forgery on some message M, $\mathcal{S}$ first checks that the output of the coin flip for the corresponding hash query on M was 0; otherwise it aborts. Then, $H(M) = g_1^{abr}$ for some r chosen by $\mathcal{S}$ and $\mathcal{S}$ returns $(\sigma_1 \sigma_2^{x_2})^{1/r}$ as the solution to its own challenge. This completes the argument.

The map ψ is used in the protocol itself when the signer computes $\psi(X_i^{a_i})$, and also in the security argument to compute $\psi({g_2'}^b)$ when simulating the random oracle H. Both these uses, however, can be easily avoided as we detail below.

BGLS-3 ring signature scheme. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a Type 3 pairing and $H : \{0,\ 1\}^* \rightarrow \mathbb{G}_1$ a hash function. The private key of an individual signer is $x \in_R \mathbb{Z}_n$ and her public key is $(W = g_1^x,\ X = g_2^x)$. A certification authority entrusted with certifying the public key should verify that $W \in \mathbb{G}_1$, $X \in \mathbb{G}_2$, $W \neq 1$, $X \neq 1$, and $e(g_1, X) = e(W, g_2)$. Note that the public key effectively gives $\psi(X) = W$ though we do not have any means

to compute ψ when supplied with a random element of $\mathbb{G}_2$. Except for this modification BGLS-3 is identical to BGLS-2. In the signing algorithm, the signer now uses W_i in place of $\psi(X_i)$. The unforgeability of BGLS-3 can be established under the co-DCDH* assumption that given g_1^{ab}, g_1^b, g_2^b it is hard to compute g_1^a; co-DCDH and co-DCDH* are equivalent under the assumption that the parameter c (cf. Lemma 2) is known. The security argument for BGLS-2 can be modified in the obvious way by setting $g_1^b = \psi(g_2^b)$. Breaking BGLS-3 is thus seen to be equivalent to solving co-DCDH*.

Note that in BGLS-3 the signer only needs the $\mathbb{G}_1$ component of the public keys of the other users in U while the verifier only needs the $\mathbb{G}_2$ component of the public keys of users in U. They can respectively ignore the other component from $\mathbb{G}_2$ or $\mathbb{G}_1$, thus leading to decreased communication bandwidth and computational cost than in BGLS-2.

4.3. **Protocols where ψ is used in the security argument only.** Several authors have used ψ to argue the security of pairing-based protocols in the asymmetric settings – these protocols include signature schemes [11, 6, 8], encryption schemes [17], and key agreement [18].

The case of the BLS signature scheme has already been discussed in detail in [16]. Several other signature schemes also use ψ in the security reduction. Some examples are the short signature scheme of Boneh and Boyen [6], the short group signature scheme of Boneh, Boyen and Shacham [8], and the anonymous short group signature of Delerablée and Pointcheval [21]. In a later version [7] Boneh and Boyen modified the security argument of their short signature scheme in [6] to remove the necessity of ψ and the same argument can also be used for the short group signature scheme of Boneh, Boyen and Shacham.

While it is known that for some protocols (e.g., BF-IBE or the Boneh-Boyen short signature scheme) it is possible to give a security reduction without taking recourse to ψ, we would like to argue in general that it is possible to avoid ψ in the security argument. As an illustrative example we consider the case of an identity-based key agreement protocol. However, the argument put forth here is quite general and is applicable to other protocols.

In [18], Chen, Cheng and Smart discuss the security and efficiency issues of several identity-based key agreement protocols in the context of different types of pairings in a modification of the Bellare-Rogaway key exchange model [5]. They show that the identity-based key agreement protocol originally proposed by Smart [35] and later modified by Chen and Kudla [19] (called SCK-1 and SCK-2 in [18]) and an enhanced version of the protocol due to McCullagh and Barreto [31, 20] (called e-MB-2 in [18]) can be proven secure in the asymmetric setting. Both the reductions make use of the map ψ and hence in Type 3 depend on a relativized assumption, i.e., oracle access to ψ. It should be noted that the protocols themselves do not require the map ψ.

The security reductions are quite involved (and runs into several pages). However, it is possible to avoid the map ψ in the security argument by including some extra elements of $\mathbb{G}_1$ in the problem instance of the respective complexity assumption for Type 3 in the manner described in §3. Here we provide a high-level description of how this can be achieved for the security argument of SCK-1. A similar approach is applicable for SCK-2 and e-MB-2.

Protocol SCK-1 in Type 3. The master secret key of the Key Generation Center (KGC) is $s \in_R \mathbb{Z}_n$ and the corresponding master public key is $R = g_2^s$. Given an identity string $\mathsf{ID} \in \{0,1\}^*$, the KGC obtains the public key as $h_{\mathsf{ID}} = H_1(\mathsf{ID})$, where $H_1 : \{0,1\}^* \to \mathbb{G}_1$ is a publicly computable hash function. The corresponding private key is $d_{\mathsf{ID}} = h_{\mathsf{ID}}^s \in \mathbb{G}_1$. Two parties, A and B with key pairs (h_A, d_A) and (h_B, d_B) respectively, run the protocol

as follows.

$$A \rightarrow B \quad : \quad E_A = g_2^x, \text{ where } x \in_R \mathbb{Z}_n$$
$$B \rightarrow A \quad : \quad E_B = g_2^y, \text{ where } y \in_R \mathbb{Z}_n.$$

A (resp. B) checks whether E_B (resp. E_A) belongs to $\mathbb{G}_2$. If the check is successful A computes $K = e_3(h_B^x, R) \cdot e_3(d_A, E_B)$ and $(E_B)^x = g_2^{xy}$ while B computes $K = e_3(h_A^y, R) \cdot e_3(d_B, E_A)$ and $(E_A)^y = g_2^{xy}$. The session key is computed as $SK = H_2(A, B, E_A, E_B, g_2^{xy}, K)$, where H_2 is another publicly computable hash function.

The authors consider the random oracle model. Session key security of SCK-1 in the Type 3 setting is established assuming the hardness of BDH-3c with oracle access to ψ, and key forward secrecy under the computational Diffie-Hellman (CDH) assumption in $\mathbb{G}_2$ with oracle access to ψ. We do not elaborate on the security arguments – interested readers can consult Theorems 1 and 2 in [18]. Instead, we only indicate how the map ψ can be avoided in the security arguments.

In Theorem 1, the simulator invokes the oracle ψ to compute either $\psi(g_2^\beta)$ or $\psi(g_2^{\beta r})$ where g_2^β is part of the BDH-3c problem instance and $r \in_R \mathbb{Z}_n$ is chosen by the simulator. The simulator can easily compute the corresponding values without the oracle if it is provided with $g_1^\beta \in \mathbb{G}_1$ as part of the problem instance, which means the security now depends on BDH-3b. This is perfectly acceptable, because the security of SCK-1 in Type 2 depends upon BDH-2 and we know that BDH-3b is at least as hard as BDH-2.

In Theorem 2, given g_2^a and g_2^b and oracle access to ψ the simulator interacts with the adversary to compute g_2^{ab}. The oracle is used in a pairing computation which is either of the form $e_3(\psi(X), Y)$ or $e_3(\psi(Y), X)$ where (X, Y) are the protocol messages exchanged in a key agreement session. At least one of these messages, say $X \in \mathbb{G}_2$, is chosen by the simulator and set to either g_2^{ar} or g_2^{br}, where $r \in_R \mathbb{Z}_n$ is chosen by the simulator. So the simulator can compute $\psi(X)$ without oracle access to ψ if she is also provided with g_1^a and g_1^b as part of the problem instance. This together with the fact that $e_3(\psi(Y), X) = e_3(\psi(X), Y)$ now allows the simulation to go through. Note that the alternative formulation of the CDH problem in $\mathbb{G}_2$ used here is equivalent to the CDH problem in $\mathbb{G}_2'$ if the generators are appropriately chosen.

5. Transforming a Protocol from Type 2 to Type 3

As case studies we have discussed in the last section how some known protocols in the Type 2 setting can be transformed to the Type 3 setting without affecting the functionality or security. We would like to generalize these observations now to propose some guidelines on how to transform an arbitrary Type-2 protocol, Protocol-2, to a Type-3 protocol, Protocol-3, where Protocol-3 is at least as secure as Protocol-2 and it is possible to achieve a better performance.

Before delving into the guidelines let's consider a (hypothetical) situation where such a transformation is *not* possible. Suppose that a protocol in the Type 2 setting involves a hash function $H : \{0,1\}^* \rightarrow \mathbb{G}_2'$ and the following steps – given a public string $\mathsf{str} \in \{0,1\}^*$, first obtain $X = H(\mathsf{str}) \in \mathbb{G}_2'$ and then compute $\psi(X) \in \mathbb{G}_1$. This protocol cannot be transformed into Type 3 because given a random $X \in \mathbb{G}_2$ one does not know how to compute $\psi(X) \in \mathbb{G}_1$. However, the protocol cannot be implemented in Type 2 either, because we do not know how to hash into $\mathbb{G}_2'$ in the first place.[4] Hence in the following we assume that

[4] For example, the Boneh-Shacham group signature scheme with verifier-local revocation [12] requires precisely these two steps and hence cannot be implemented in either Type 2 or Type 3.

Protocol-2 does not require hashing into $\mathbb{G}_2'$ and argue that it is always possible to *naturally* transform it into Protocol-3.

Suppose we have a protocol, Protocol-2, and its security reduction with respect to some hard problem $\mathcal{P}$-2. Let $\mathcal{P}$-3 be the natural counterpart of $\mathcal{P}$-2 in Type 3 as described in §3. Our aim is to obtain the corresponding Protocol-3 with an analogous security reduction with respect to $\mathcal{P}$-3. Protocol-2 will typically include elements from $\mathbb{G}_1, \mathbb{G}_2'$ and $\mathbb{G}_T$ and so also the corresponding security reduction. (In the security reduction, the challenger essentially simulates the protocol environment for an attacker based on the given instance of the hard problem.) Note that the elements of $\mathbb{G}_1$ and $\mathbb{G}_T$ will not be affected in any way when we convert Protocol-2 and its security reduction to Type 3. So our primary concern will be with the elements of $\mathbb{G}_2'$.

Let g_1, g_2', g_2 be the generators of $\mathbb{G}_1$, $\mathbb{G}_2'$ and $\mathbb{G}_2$ respectively, where $g_1 = \psi(g_2')$ and $g_2 = \rho(g_2')$ (for simplicity we assume $c = 1$). Recall from §2.2 that for any elliptic curve with even embedding degree, the task of computing the Type 2 pairing $e_2 : \mathbb{G}_1 \times \mathbb{G}_2' \to \mathbb{G}_T$ can be easily reduced to the task of computing the Type 3 pairing $e_3 : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Recall also from §2.3 that every element $X \in \mathbb{G}_2'$ can be represented by the pair $(X_1, X_2) = (\psi(X), \rho(X)) \in \mathbb{G}_1 \times \mathbb{G}_2$. Given Protocol-2 and its reductionist security argument with respect to $\mathcal{P}$-2, let's first consider the following 'preliminary' transformation. Represent each $X \in \mathbb{G}_2'$ appearing in Protocol-2 by $(X_1, X_2) \in \mathbb{G}_1 \times \mathbb{G}_2$. Also make similar changes in the problem instance of $\mathcal{P}$-2 and the security reduction.

It is easy to see that *any* protocol in the Type 2 setting as well as its security argument can be rewritten in this way where elements of $\mathbb{G}_2'$ are represented by elements of $\mathbb{G}_1 \times \mathbb{G}_2$. By Lemma 2, computing a Type 2 pairing $e_2(\cdot, X)$ is equivalent to computing the Type 3 pairing value $e_3(\cdot, X_2)$, and hence it is possible to employ e_3 directly instead of e_2 in the protocol. It is now possible to argue the security of this "modified" protocol with respect to $\mathcal{P}$-3. This new representation allows a better performance than the conventional representation of a protocol in the Type 2 setting without affecting the functionality and security of the protocol.

However, just because the protocol is now described in terms of elements of $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ does not necessarily mean that we are no longer in the Type 2 setting. What really changed now is the representation of elements of $\mathbb{G}_2'$, i.e., each $X \in \mathbb{G}_2'$ involved in Protocol-2 will now be written as $(\psi(X), \rho(X)) \in \mathbb{G}_1 \times \mathbb{G}_2$. But for some protocols (and their security argument) it might be sufficient to work only with the $\mathbb{G}_2$ component of X and the $\mathbb{G}_1$ component is actually redundant. In that case one can drop those redundant elements of $\mathbb{G}_1$ from the protocol description without affecting functionality and security. The *modified* protocol will now be in the Type 3 setting and (in most cases) allows a still better performance.

We illustrate this process of transformation with an example – the SCK-1 protocol of §4.3. We start with the protocol in Type 2 with new representation of $\mathbb{G}_2'$. Then for each $X = (X_1, X_2) \in \mathbb{G}_2'$ in the protocol description, check whether $X_1 = \psi(X) \in \mathbb{G}_1$ is *necessary* for the protocol or the security argument; if not then X_1 is dropped. At the end we get the corresponding protocol description in Type 3.

Protocol SCK-1 in Type 2. The KGC has a master secret key $s \in_R \mathbb{Z}_n$ and a corresponding master public key $R = {g_2'}^s = (g_1^s, g_2^s) = (R_1, R_2) \in \mathbb{G}_1 \times \mathbb{G}_2$. Given an identity string $\mathsf{ID} \in \{0,1\}^*$, the corresponding public and private keys are obtained as in §4.3, i.e., $h_{\mathsf{ID}} = H_1(\mathsf{ID}) \in \mathbb{G}_1$ and $d_{\mathsf{ID}} = h_{\mathsf{ID}}^s \in \mathbb{G}_1$. Two parties, A and B with key pairs (h_A, d_A) and

(h_B, d_B) respectively, run the protocol as follows.

$$\begin{aligned} A \to B &: X = {g_2'}^x = (g_1^x, g_2^x) = (X_1, X_2), \text{ where } x \in_R \mathbb{Z}_n \\ B \to A &: Y = {g_2'}^y = (g_1^y, g_2^y) = (Y_1, Y_2), \text{ where } y \in_R \mathbb{Z}_n. \end{aligned}$$

On receiving Y, A first checks whether Y actually belongs to $\mathbb{G}_2'$. If the check is successful she computes $K = e_2(h_B^x, R) \cdot e_2(d_A, Y)$ and then the session key $SK = H_2(A, B, X, Y, Y^x, K)$. B follows an analogous procedure to compute the same session key.

In our new representation of $\mathbb{G}_2'$, testing whether Y belongs to $\mathbb{G}_2'$ amounts to deciding whether Y_1 belongs to $\mathbb{G}_1$ and Y_2 belongs to $\mathbb{G}_2$ and then computing a product of pairings. Similarly, computing $Y^x \in \mathbb{G}_2'$ amounts to computing $Y_1^x \in \mathbb{G}_1$ and $Y_2^x \in \mathbb{G}_2$. Also note that $e_2(\cdot, R)$ can be easily computed from $e_3(\cdot, R_2)$ and similarly $e_2(\cdot, X)$ and $e_2(\cdot, Y)$. So the new representation of $\mathbb{G}_2'$ allows a much better performance than the conventional representation. However, we are still in the Type 2 setting.

Now we consider whether it is possible to drop any of the $\mathbb{G}_1$ components R_1, X_1, Y_1 of the $\mathbb{G}_2'$ elements R, X, Y from the protocol description without affecting the functionality and security of the protocol. This can be a two step process. First we check whether, for example, $X_1 = \psi(X)$ (and not just X) is explicitly used at any stage in the protocol or the security argument. If there is no such use we right away discard X_1 from the protocol description. Otherwise in the next step we check whether it is still possible to obtain X_1 without recourse to ψ. In that case also we drop X_1 from the protocol description.

Investigating this way we observe that R_1 is not used in the protocol but the simulator needs it in the security argument. However, in the simulation R_1 is provided as part of the original problem instance so we can drop it from the protocol description. Next consider the $\mathbb{G}_1$ component of the protocol messages, i.e., X_1 and Y_1. They too do not play any *independent* role in the protocol (apart from being a part of the representation of X and Y) while in the security argument one of them (but not both) is explicitly used by the simulator. So it may appear at first sight that we have to provide X_1 and Y_1 as part of the protocol message. However, in a typical simulation of a key-agreement session at least one of the messages (say X) is chosen by the simulator. As we already argued in §4.3, the simulator can directly compute X_1 based on what she received as part of the problem instance. So we can drop X_1 and Y_1 and modify the protocol so that the protocol messages are elements of $\mathbb{G}_2$ only (i.e., X_2, Y_2). Note that in the modified protocol A derives the session key as $SK = H_2(A, B, X_2, Y_2, Y_2^x, K)$ (and analogously for B).

This modified protocol achieves a better performance both in terms of communication bandwidth and computational cost because the users are no longer burdened with the unnecessary $\mathbb{G}_1$ components of the messages and the master public key. Note that once these $\mathbb{G}_1$ components are dropped there is no way to obtain the conventional representation of the corresponding elements of $\mathbb{G}_2'$ or to apply ψ to them and we are *actually* in the Type 3 setting.

The transformation discussed above gives us the essential cue for the actual transformation from Protocol-2 to Protocol-3. Namely, for each $X \in \mathbb{G}_2'$ involved in a particular step of Protocol-2 or its security argument, the corresponding step in Protocol-3 or its security argument requires the computation of $X_2 \in \mathbb{G}_2$, and also $X_1 \in \mathbb{G}_1$ if $\psi(X)$ is necessary for the protocol or its security argument.[5] Note that in Protocol-3 and its security argument, X_2 can be obtained in the same way as X in the case of Protocol-2. However, we must have

[5]We consider $\psi(X)$ to be necessary if either $\psi(X)$ or some $\psi(Y)$ is required in the protocol or the security argument where $Y \in \mathbb{G}_2'$ is derived from X.

some means to obtain X_1 in Type 3 when necessary. The transformation will fail only when we are unable to do so.

In Protocol-2, X can be part of the static data such as the private key, the public key or the public parameters, or the "run-time" data, e.g., the signature in a signature scheme, the ciphertext in an encryption scheme, or the ephemeral key in a key agreement protocol. Given Protocol-2 and its security reduction, we consider these situations separately for the conversion.

Private data contains an element $X \in \mathbb{G}'_2$. If either the protocol or the security reduction requires $\psi(X)$ then include both $X_1 \in \mathbb{G}_1$ and $X_2 \in \mathbb{G}_2$ in the private information in Protocol-3. If neither requires $\psi(X)$ then include only X_2.

Public data contains an element $X \in \mathbb{G}'_2$. X can be part of the public key of a user when we are in the traditional public-key setting or it can be part of the public parameters of the KGC when we are in the identity-based setting. Whatever be the case, the entity who generated X must also know its discrete log with respect to a known base in $\mathbb{G}'_2$ (since hashing into $\mathbb{G}'_2$ is not feasible). There are two possibilities depending upon whether the protocol uses $\psi(X)$ or not.

(1) Protocol-2 employs $\psi(X)$: Replace $X = {g'_2}^x$ in the public data of Protocol-2 with $X_1 = g_1^x \in \mathbb{G}_1$ and $X_2 = g_2^x \in \mathbb{G}_2$ in Protocol-3. This is always doable because the entity who generated X must know x. In the traditional PKI setting a certification authority entrusted with certifying the public key can easily verify whether they are properly generated or not, and in the identity-based setting they are assumed to be properly generated by the trusted KGC. If in Protocol-2 a user Alice requires only $\psi(X)$ (resp. X) then she will be concerned with only X_1 (resp. X_2) in Protocol-3. This is the case for the BGLS-3 ring signature scheme which, as we have already noted, performs better than the BGLS-2 ring signature because now the ring signer is concerned only with the $\mathbb{G}_1$ component of the public key of the other signers in the group while the verifier is concerned with the $\mathbb{G}_2$ components only. They can respectively ignore the $\mathbb{G}_2$ and $\mathbb{G}_1$ components of the public key leading to decreased communication bandwidth.

(2) $\psi(X)$ is necessary for the security argument of Protocol-2: Replace $X = {g'_2}^x$ in Protocol-2 with $X_2 = g_2^x$ in Protocol-3, as was done in the case of BF-IBE-3 in §4.1, and BLS-3 and Waters-3a in [16]. This allows smaller public keys and hence (sometimes) less computation in Protocol-3. In some protocols it might still be necessary to include $X_1 = g_1^x$ in the public key. For example, X_1 may be required for the reduction to go through as in the BGLS-3 aggregate signature scheme (see [16]). Note that even if X_1 were included in the public key or the public parameters, it would never be required in the actual protocol run. So the users will effectively ignore X_1 in their computation, which in turn will lead to better optimization in terms of communication bandwidth and perhaps even computational cost compared to Protocol-2.

Protocol message contains elements of $\mathbb{G}'_2$. Suppose that in Protocol-2 Bob receives a message from Alice containing some $Y \in \mathbb{G}'_2$. Alice may compute Y either solely based on her secret information or in combination with some public data, such as Bob's public key or the system-wide public parameters. If Bob requires $\psi(Y)$ then in Protocol-3 Alice should send both $Y_1 \in \mathbb{G}_1$ and $Y_2 \in \mathbb{G}_2$ to Bob. This is doable but, to the best of our knowledge, is not required by any existing protocol. If Bob's computation in Protocol-2 depends only on Y then it is sufficient in Protocol-3 to send only Y_2. However, for the known security argument

of some protocols to go through, it might be necessary to send Y_1 also. For example, in the security argument of BF-IBE-3 it is sufficient to send the $\mathbb{G}_2$ component only of the ephemeral key, while for the security argument of the Waters-3b signature scheme (see [16]) we need both the $\mathbb{G}_1$ and $\mathbb{G}_2$ components for one of the signature components. In the former case, Protocol-3 will be more optimized than Protocol-2 while it will be at least as efficient as Protocol-2 in the latter.

Now consider the security reduction for Protocol-3 where the challenger essentially simulates the protocol environment for an adversary based on the problem instance in the given security model. Note that Protocol-3 is derived based on both Protocol-2 and its security reduction. Once we have obtained Protocol-3 in this manner, it is relatively straightforward to modify the security argument of Protocol-2 to get an analogous security argument for Protocol-3.

Like the protocol, our only concern here is with those elements $X \in \mathbb{G}_2'$ for which the challenger needs to compute $\psi(X)$ in the simulation. Note that if X is part of the problem instance of $\mathcal{P}$-2 then by definition both X_1 and X_2 must be part of the problem instance of $\mathcal{P}$-3. So we only need to consider the following two situations in the simulation in Type 2 and show how one can obtain X_1 in each case in Type 3.

(1) X is part of the message sent by the adversary: This can only happen if there is a similar step in Protocol-2 and has been taken care of when we transformed it into Protocol-3. Namely the corresponding message in Protocol-3 includes both X_1 and X_2 and hence the adversary must send both to the challenger.
(2) The challenger generates X: The challenger cannot generate this by hashing into $\mathbb{G}_2'$. So the challenger can generate X either solely based on its own random coin tosses, or along with that randomness it may also depend on the adversary's input or the problem instance $\mathcal{P}$-2. Whatever be the case, the challenger will have enough information to generate both X_1 and X_2 in the security reduction of Protocol-3.

6. Concluding Remarks

Many pairing-based protocols in the asymmetric setting rely on the existence of an efficiently-computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$, i.e., the Type 2 setting. Some earlier works in pairing-based cryptography gave the impression that such an isomorphism is necessary for the functionality or the security of the protocols (or both). In contrast, we have demonstrated that relying on such an isomorphism is more of an artifact of initial research in this area rather than an actual necessity as far as the functionality and security of the protocols are concerned. Moreover, restricting a protocol to such a setting in most cases has a negative impact on performance.

In particular, we have provided evidence in support of the following assertions:

(1) For any hard problem $\mathcal{P}$-2 in the Type 2 setting, there is a natural counterpart $\mathcal{P}$-3 in the Type 3 setting which is equally hard.
(2) Any protocol Protocol-2 and its security argument based on $\mathcal{P}$-2 in the Type 2 setting can be naturally converted to Protocol-3 with an analogous security argument based on $\mathcal{P}$-3 in the Type 3 setting.
(3) Protocol-3 is at least as efficient as Protocol-2 and in most cases outperforms Protocol-2. In some situations one can further optimize Protocol-3, for example, by hashing into $\mathbb{G}_2$ instead of $\mathbb{G}_1$ as is the case with BF-IBE-3b. Such optimizations are not possible for Protocol-2 as it is not known how to hash into $\mathbb{G}_2'$.

Setting aside the question of performance, our study indicates that there is no major difference of cryptographic significance when a protocol originally described in the Type 2 setting is transformed into Type 3. This leads us to posit the question of whether there exists a cryptographic protocol which necessarily has to be restricted to Type 2 for implementation or security reasons. We conclude with the following remarks and question.

(1) Some authors have used the extended Diffie-Hellman assumption (XDH) to argue the security of their protocols [8, 15, 21]. The XDH assumption requires that the decision Diffie-Hellman problem (DDH) is hard in $\mathbb{G}_1$. If it is also required that DDH is hard in $\mathbb{G}_2$ then the corresponding assumption is called symmetric XDH (SXDH) [1]. Note that in the Type 2 setting, DDH is easy in $\mathbb{G}_2'$ because of ψ. Hence, the protocols described in [1, 37] where security is based on the SXDH assumption cannot be instantiated securely in Type 2 (but can be implemented in Type 3 since DDH is believed to be hard in $\mathbb{G}_2$). On the other hand, if some Protocol-2 or its security argument requires DDH to be hard in $\mathbb{G}_1$ but easy in $\mathbb{G}_2'$, then it may appear at first glance that Protocol-2 cannot be converted to the Type 3 setting. However, we expect that the conversion would be possible by adding the appropriate elements from $\mathbb{G}_1$ to the protocol or its security argument in the same manner as was used earlier to avoid ψ. Note that we are not aware of the existence of any such protocols.

(2) In Type 3 it is possible to make a minimalist complexity assumption such as BDH-3c as proposed by Galbraith [13]. Such minimalist assumptions do not have any counterpart in Type 2. The question is whether there exists a protocol whose security is based on such an assumption.

(3) Some protocols, such as the Boneh-Shacham group signature scheme with verifier-local revocation [12] and the SYL, RYY and BMP identity-based key agreement protocols (see [18]), that involve hashing into $\mathbb{G}_2$ followed by an application of ψ on the resulting hash values cannot be implemented in either Type 2 or Type 3. However, they can be implemented using Type 1 pairings.

Acknowledgements

Thanks to M. Prem Laxman Das for simplifying the proof of Lemma 2.

References

[1] G. Ateniese, J. Camenisch, S. Hohenberger and B. de Medeiros, "Practical group signatures without random oracles", Cryptology ePrint Archive Report 2005/385. Available from http://eprint.iacr.org/2005/385.

[2] F. Bao, R. Deng and H. Zhu, "Variations of Diffie-Hellman problem", *Information and Communication Security – ICICS 2003*, Lecture Notes in Computer Science, 2836 (2003), 301–312.

[3] P. Barreto, B. Lynn and M. Scott, "Efficient implementation of pairing-based cryptosystems", *Journal of Cryptology*, 17 (2004), 321–334.

[4] 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.

[5] M. Bellare and P. Rogaway, "Entity authentication and key distribution", *Advances in Cryptology – CRYPTO 1993*, Lecture Notes in Computer Science, 773 (1993), 232–249.

[6] D. Boneh and X. Boyen, "Short signatures without random oracles", *Advances in Cryptology – EUROCRYPT 2004*, Lecture Notes in Computer Science, 3027 (2004), 56–73.

[7] D. Boneh and X. Boyen, "Short signatures without random oracles and the SDH assumption in bilinear groups", *Journal of Cryptology*, 21 (2008), 149–177.

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

[9] D. Boneh and M. Franklin, "Identity-based encryption from Weil pairing", *SIAM Journal of Computing*, 32 (2003), 586–615.
[10] 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.
[11] D. Boneh, B. Lynn and H. Shacham, "Short signatures from the Weil pairing", *Journal of Cryptology*, 17 (2004), 297–319.
[12] D. Boneh and H. Shacham, "Group signatures with verifier-local revocation", *11th ACM Conference on Computer and Communications Security – CCS 2004*, ACM Press, 2004, 168–177.
[13] X. Boyen, "The uber-assumption family – A unified complexity framework for blinear groups", *Pairing-Based Cryptography – Pairing 2008*, Lecture Notes in Computer Science, 5209 (2008), 39–56.
[14] X. Boyen, Q. Mei and B. Waters, "Direct chosen ciphertext security from identity-based techniques", *12th ACM Conference on Computer and Communications Security – CCS 2005*, ACM Press, 2005, 320–329.
[15] J. Camenisch, S. Hohenberger and A. Lysyanskaya, "Compact E-cash", *Advances in Cryptology – EUROCRYPT 2005*, Lecture Notes in Computer Science, 3494 (2005), 302–321.
[16] S. Chatterjee, D. Hankerson, E. Knapp and A. Menezes, "Comparing two pairing-based aggregate signature schemes", *Designs, Codes and Cryptography*, to appear. Also available from http://eprint.iacr.org/2009/060.
[17] L. Chen and Z. Cheng, "Security proof of the Sakai-Kasahara's identity-based encryption scheme", *Cryptography and Coding*, Lecture Notes in Computer Science, 3706 (2005), 442–459.
[18] L. Chen, Z. Cheng and N. Smart, "Identity-based key agreement protocols from pairings", *International Journal of Information Security*, 6 (2007), 213–241.
[19] L. Chen and C. Kudla, "Identity-based authenticated key agreement from pairings", *IEEE Computer Security Foundations Workshop*, 219–233, 2003. The modified version is available at Cryptology ePrint Archive, Report 2002/184.
[20] Z. Cheng and L. Chen, "On security proof of McCullagh-Barreto's key agreement protocol and its variants", Cryptology ePrint Archive Report 2005/201. Available from http://eprint.iacr.org/2005/201.
[21] C. Delerablée and D. Pointcheval, "Dynamic fully anonymous short group signatures", *Progress in Cryptology – VIETCRYPT 2006*, Lecture Notes in Computer Science, 4341 (2006), 193–210.
[22] 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.
[23] S. Galbraith, F. Hess and F. Vercauteren, "Hyperelliptic pairings", *Pairing-Based Cryptography – Pairing 2007*, Lecture Notes in Computer Science, 4575 (2007), 108-131.
[24] S. Galbraith, K. Paterson and N. Smart, "Pairings for cryptographers", *Discrete Applied Mathematics*, 156 (2008), 3113–3121.
[25] D. Galindo, "Boneh-Franklin Identity-based encryption revisited", *Automata, Language and Programming – ICALP 2005*, Lecture Notes in Computer Science, 3580 (2005), 791–802.
[26] D. Hankerson, A. Menezes and M. Scott, "Software implementation of pairings", in *Identity-Based Cryptography*, M. Joye and G. Neven, eds., IOS Press, 2008.
[27] F. Hess, N. Smart and F. Vercauteren, "The eta pairing revisited", *IEEE Transactions on Information Theory*, 52 (2006), 4595–4602.
[28] A. Joux, "A one round protocol for tripartite Diffie-Hellman", *Journal of Cryptology*, 17 (2004), 263–276.
[29] 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.
[30] E. Lee, H. Lee and C. Park, "Efficient and generalized pairing computation on abelian varieties", *IEEE Transactions on Information Theory*, 55 (2009), 1793–1803.
[31] N. McCullagh and P. Barreto, "A new two-party identity-based authenticated key agreement", *Topics in Cryptology – CT-RSA 2005*, Lecture Notes in Computer Science, 3376 (2005), 262–274.
[32] A. Miyaji, M. Nakabayashi and S. Tanako, "New explicit condition of elliptic curve trace for FR-reduction", *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E84-A (2001), 1234-1243.
[33] R. Sakai, K. Ohgishi and M. Kasahara, "Cryptosystems based on pairing over elliptic curve" (in Japanese), *The 2000 Symposium on Cryptography and Information Security*, 2000.
[34] H. Shacham, "New paradigms in signature schemes", PhD thesis, Stanford University, 2005.
[35] N. Smart, "An identity based authenticated key agreement protocol based on the Weil pairing", *Electronics Letters*, 38 (2002), 630–632.

[36] N. Smart and F. Vercauteren, "On computable isomorphisms in efficient pairing-based systems", *Discrete Applied Mathematics*, 155 (2007), 538–547.
[37] B. Waters, "Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions", *Advances in Cryptology – CRYPTO 2009*, Lecture Notes in Computer Science, 5677 (2009), 595–618.

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

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