On Computable Isomorphisms in Efficient Asymmetric Pairing Based Systems *

N.P. Smart[1] and F. Vercauteren[2]**

[1] Department of Computer Science,
University of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB,
United Kingdom
nigel@cs.bris.ac.uk
[2] Department of Electrical Engineering
University of Leuven
Kasteelpark Arenberg 10,
B-3001 Leuven-Heverlee,
Belgium
frederik.vercauteren@esat.kuleuven.be

Abstract. In this paper we examine the hard problems underlying asymmetric pairings, their precise relationships and how they affect a number of existing protocols. Furthermore, we present a new model for the elliptic curve groups used in asymmetric pairings, which allows both an efficient pairing and an efficiently computable isomorphism.

Keywords: Pairing-based cryptography, Tate pairing, elliptic curve

1 Introduction

In recent years we have seen the advent of various protocols based on pairings. However, much of the literature uses a confusing array of underlying hard problems and differing notation. This is because the original protocols were defined in the context of pairings on supersingular curves, in which the pairing is between a group and itself, i.e.

$$G \times G \longrightarrow G_T.$$

In later papers the emphasis has been on so-called asymmetric pairings, where the pairing is from two (possibly) different groups into a third, i.e.

$$G_1 \times G_2 \longrightarrow G_T.$$

* The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. The information in this document reflects only the authors' views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

** Postdoctoral Fellow of the Research Foundation - Flanders (FWO)

As a result, it is hard to reconcile one paper against another and in addition, the area is further confused by various authors insisting on a computable isomorphism from G_2 to G_1.

In this paper we elaborate on these differences in greater detail. We carefully point out what are the underlying hard problems and how they are related. In addition we point out the reliance on the computability of the isomorphism from G_2 to G_1. In many protocols the isomorphism is only needed to exist so as to define the protocol, and an isomorphism between two finite cyclic groups of the same order clearly exists a priori. The issue with whether the isomorphism is computable often only becomes important in the security proof. However, in such cases one could model the security proof with respect to a relativised result whereby the adversary has access to an oracle that computes this isomorphism.

The reason these considerations matter is that given standard representations in the literature for G_2 and G_1 it is impossible to have both a simultaneously efficient pairing and a computable isomorphism from G_2 to G_1. The only notable exception being supersingular curves with distortion map; however, we will mainly focus on ordinary curves. In this paper we also present a new model for G_2 which enables an efficiently computable pairing, a computable isomorphism on ordinary curves plus gives a reduced bandwidth representation. The only draw back with our model for G_2 is that it appears difficult to construct efficient hash functions with codomain equal to G_2.

Our paper is organised as follows. In Section 2 we examine the underlying hard problems in asymmetric pairings and examine their relationships and how they affect a number of existing protocols. This section is presented completely in the abstract with no mention of elliptic curves. In Section 3 we present the main ideas we require from the theory of pairings on elliptic curves. We summarise a number of known implementation issues and discuss some problems. In Section 4 we present our model of the group G_2 which allows both an efficient pairing and an efficient isomorphism from G_2 to G_1. The big advantage of our solution is that it is more memory efficient and that the trace and pairing operations are simpler since they essentially "come for free". In Section 5 we compare our model with other models of G_2 and we end with some other possible representations of G_2. Finally, Section 7 concludes the paper.

We would like to thank Steven Galbraith for comments on an earlier version of this paper.

2 Abstract Pairing Based Protocols

In this section we examine a number of pairing based protocols and point out some interesting observations which motivate the work in this paper. To emphasise the points we wish to make more clear, we present our arguments and observations completely abstract, without reference to any underlying concrete groups. Then in later sections when we specialise to elliptic curve groups one can more easily see how the problems discussed in this section apply to elliptic curve groups.

We first need to define an overall problem instance on which to base our pairing based protocols. We define a *pairing problem instance* to be a tuple $\Gamma = (q, G_1, G_2, G_T, P_1, P_2, \hat{p})$ where G_1, G_2 and G_T denote groups of prime order q. For convenience we shall write G_1 and G_2 additively and G_T multiplicatively. We let P_1 and P_2 denote two fixed generators of G_1 and G_2 respectively. Our pairing problem instance also contains a computable bilinear pairing

$$\hat{p} : G_1 \times G_2 \longrightarrow G_T.$$

We denote the pairing by $\hat{p}$, to stress that in this section we are dealing with an abstract pairing. In future sections we shall use the notation $\hat{t}$ when dealing with the concrete Tate pairing; in this case the groups G_1 and G_2 become subgroups of elliptic curves and G_T is a multiplicative subgroup of a finite field.

Since G_1 and G_2 have the same prime group order there clearly is a group isomorphism φ between G_2 and G_1 such that $\varphi(P_2) = P_1$. In this section we make no assumption as to whether φ, or φ^{-1}, is efficiently computable.

We shall call one of our groups G_i, for either $i = 1$ or $i = 2$, *randomly samplable* if it is possible to simply write down a random element of the group, without necessarily computing rP_i for some random value of r. This notion will become important in later sections when analysing our pairing based protocols. In particular if a group is *not* randomly samplable then the only way one can produce new elements within a protocol is by computing rP_i for some value of r.

Given a pairing problem instance one can define a number of variations of the Bilinear Diffie–Hellman problem, as follows

Definition 1 (The $\mathtt{BDH}_{i,j,k}$ Problem).
Given a pairing problem instance $\Gamma = (q, G_1, G_2, G_T, P_1, P_2, \hat{p})$ and values $i, j, k \in \{1, 2\}$ we define the $\mathtt{BDH}_{i,j,k}$ problem to be the following: Given aP_i, bP_j and cP_k, with $a, b, c \in \mathbb{F}_q$, we are asked to compute $\hat{p}(P_1, P_2)^{abc}$.

The advantage of any adversary A against this problem is defined to be

$$\Pr[\alpha = \hat{p}(P_1, P_2)^{abc} : \alpha \leftarrow A(aP_i, bP_j, cP_k, \Gamma)].$$

Definition 2 (The $\mathtt{coBDH}_{j,k}$ Problem).
Given a pairing problem instance $\Gamma = (q, G_1, G_2, G_T, P_1, P_2, \hat{p})$ and values $j, k \in \{1, 2\}$ we define the $\mathtt{coBDH}_{j,k}$ problem to be the following: Given aP_1, aP_2, bP_j and cP_k, with $a, b, c \in \mathbb{F}_q$, we are asked to compute $\hat{p}(P_1, P_2)^{abc}$.

The advantage of any adversary A against this problem is defined to be

$$\Pr[\alpha = \hat{p}(P_1, P_2)^{abc} : \alpha \leftarrow A(aP_1, aP_2, bP_j, cP_k, \Gamma)].$$

Definition 3 (The $\mathtt{BDH}^{\varphi}_{i,j,k}$ Problem). *The $\mathtt{BDH}^{\varphi}_{i,j,k}$ problem is the same as the $\mathtt{BDH}_{i,j,k}$ problem, with the only difference that the adversary now has access to an oracle which computes values under the isomorphism φ mentioned above, but does not necessarily have access to an oracle which computes the inverse isomorphism φ^{-1}. Again the advantage is defined to be*

$$\Pr[\alpha = \hat{p}(P_1, P_2)^{abc} : \alpha \leftarrow A^{\varphi}(aP_i, bP_j, cP_k, \Gamma)].$$

There are clearly relationships between these problems:

- The problems $\mathtt{BDH}_{i,j,k}$ and $\mathtt{BDH}_{i',j',k'}$ are polynomial time equivalent if $i + j + k = i' + j' + k'$, the same holds for the problems $\mathtt{BDH}^{\varphi}_{i,j,k}$ and $\mathtt{BDH}^{\varphi}_{i',j',k'}$.
- In the case when $i + j + k \leq i' + j' + k'$ then an efficient algorithm to solve $\mathtt{BDH}^{\varphi}_{i,j,k}$ implies an efficient algorithm to solve $\mathtt{BDH}^{\varphi}_{i',j',k'}$ Hence, the $\mathtt{BDH}^{\varphi}_{1,1,1}$ problem is possibly harder than the $\mathtt{BDH}^{\varphi}_{2,2,2}$ problem.
- On the other hand the problems $\mathtt{BDH}_{i,j,k}$ and $\mathtt{BDH}_{i',j',k'}$ do not appear to be related when $i + j + k \neq i' + j' + k'$.
- An efficient algorithm to solve $\mathtt{BDH}_{i,j,k}$ implies an efficient algorithm to solve $\mathtt{BDH}^{\varphi}_{i,j,k}$, but the converse is only true when the map φ exists and is not just provided as an oracle.
- An efficient algorithm to solve $\mathtt{BDH}_{i,j,k}$ implies an efficient algorithm to solve $\mathtt{coBDH}_{j,k}$ and an efficient algorithm to solve $\mathtt{coBDH}_{j,k}$ implies an efficient algorithm to solve $\mathtt{BDH}^{\varphi}_{2,j,k}$.
- Finally, we also remark that an efficient algorithm to solve the discrete logarithm problem in either group G_1, G_2, G_T, implies an efficient algorithm to solve all the above problems.

We can also define variations of the Computational Diffie–Hellman problem.

Definition 4 (The $\mathtt{CDH}_{i,j,k}$ Problem). *Given a pairing problem instance $\Gamma = (q, G_1, G_2, G_T, P_1, P_2, \hat{p})$ and values $i, j, k \in \{1, 2\}$ we define the $\mathtt{CDH}_{i,j,k}$ problem to be the following: Given aP_i and bP_j, with $a, b \in \mathbb{F}_q$, we are asked to compute abP_k. The advantage of any adversary A against this problem is defined to be*

$$\Pr[\alpha = abP_k : \alpha \leftarrow A(aP_i, bP_j, \Gamma)].$$

A variant $\mathtt{CDH}^{\varphi}_{i,j,k}$ of this problem can be defined similarly to above, as can similar relations between the problems. In addition we see that an efficient algorithm to solve $\mathtt{CDH}_{i,j,k}$ can be used to solve the $\mathtt{BDH}_{i,j,k'}$, problem when $k \neq k'$.

Decisional variants of the above problems can also be defined, as can then the resulting gap problems (i.e. the above computational problems relative to an oracle which solves the equivalent decisional problem). Note, however that certain decisional problems are easy, for example the decision variant of $\mathtt{CDH}_{i,j,k}$ with $i \neq j$ is simple since one can use the pairing to test for a valid Diffie–Hellman tuple.

We shall now turn to discussing what these concepts imply for a number of pairing based cryptosystems.

2.1 The Boneh–Franklin Encryption Scheme

Here we discuss the Boneh–Franklin ID-based encryption scheme [3] in the context of our abstract pairings above. We assume the reader is familiar with the scheme and so only recap on the salient points for which we need do define notation.

Given a pairing problem instance $\Gamma = (q, G_1, G_2, G_T, P_1, P_2, \hat{p})$ the Boneh–Franklin scheme requires the following four elements of the groups G_1 and G_2 to be defined.

1. The public key of the trust authority is defined to be $R = xP_i$, for $i \in \{1, 2\}$, and some secret value $x \in \mathbb{F}_q$.
2. The public key of the user is defined to be $Q_{\mathtt{ID}} = H(\mathtt{ID}) \in G_j$, where $j \in \{1, 2\}$ and H is a cryptographic hash function from $\{0, 1\}^*$ to G_j.
3. The ephemeral key in the ciphertext is given by $U = rP_k$, for $k \in \{1, 2\}$, and some ephemeral secret value $r \in \mathbb{F}_q$.
4. The secret key of the user $S_{\mathtt{ID}} \in G_l$, for $l \in \{1, 2\}$, must satisfy either $S_{\mathtt{ID}} = xQ_{\mathtt{ID}}$, or $S_{\mathtt{ID}} = \varphi(xQ_{\mathtt{ID}})$.

From this set up we can instantly notice a number of points:

– Due to point 2 the group G_j must be randomly samplable, otherwise one would never be able to implement such a hash function.
– Due to point 4 we must have either $l = j$, or if there is an oracle to compute φ we may also have $(j, l) = (2, 1)$.
– The encryptor needs to compute the pairing of either $Q_{\mathtt{ID}}$ and R, or $Q_{\mathtt{ID}}$ and $\varphi(R)$, or $\varphi(Q_{\mathtt{ID}})$ and R. This implies that either $i \neq j$, or if there is an oracle to compute φ we may also have $i = j = 2$.
– The decryptor needs to compute the pairing of either $S_{\mathtt{ID}}$ and U, or $S_{\mathtt{ID}}$ and $\varphi(U)$, or $\varphi(S_{\mathtt{ID}})$ and U. This implies that either $k \neq l$, or if there is an oracle to compute φ we may also have $k = l = 2$.

Since to implement any scheme requiring an oracle for φ, we need not just an oracle, but an actual efficient and explicit algorithm, this implies the following: If φ is not efficiently computable then to implement the Boneh–Franklin scheme one requires either $(i, j, k, l) = (1, 2, 1, 2)$ or $(2, 1, 2, 1)$. However, if φ is efficiently computable then one can have any one of the following tuples for (i, j, k, l)

$$(2, 2, 2, 2), (1, 2, 2, 2), (2, 2, 1, 2), (1, 2, 1, 2)(2, 2, 2, 1), (1, 2, 2, 1), (2, 1, 2, 1).$$

Subject to the constraint that the group G_j is randomly samplable.

If we now turn to the proof of the Boneh–Franklin encryption scheme, in particular the proofs of Lemma 4.2 and 4.6 of [3], and extending them to the more general situation of $G_1 \neq G_2$ as considered in our work, we see that the hash function H is modelled as a random oracle via the following simulation: For any identity $\mathtt{ID}_t$ the random oracle H is computed via $Q_{\mathtt{ID}_t} = b_t P_j$, for some random b_t, with the corresponding values of $S_{\mathtt{ID}_t}$ computed via $b_t R$, or $\varphi(b_t R)$. This implies that either $i = j$, or φ is available via oracle access to the adversary and $i = 2$ and $j = 1$.

Hence, the existing security proof of the Boneh–Franklin encryption scheme implies that:

– If φ is not efficiently computable then one needs to select $(i, j, k, l) = (2, 1, 2, 1)$ and the security proof of the scheme is relative to the hardness of the $\mathtt{BDH}^{\varphi}_{2,1,2}$ problem. In other words, although the scheme in this instance may not require an efficiently computable φ the security proof is relative to an adversary which has oracle access to φ. Hence, if one is unwilling to accept such a relativised security result then one has to revert to basing ones security on the

$\mathtt{coBDH}_{1,2}$ problem, as pointed out by [3]. However, the $\mathtt{coBDH}_{1,2}$ problem is also somewhat unnatural.

– If φ is effectively computable then one needs to select (i, j, k, l) from one of the following

$$(2, 2, 2, 2), (2, 2, 1, 2), (2, 2, 2, 1), (2, 1, 2, 1).$$

The security proof is then relative to the hardness of $\mathtt{BDH}^{\varphi}_{i,j,k}$, except that now φ is not only given as oracle access to the adversary, but is actually computable. Hence, the hardness is relative to the standard $\mathtt{BDH}_{i,j,k}$ problem. Given our comments on the relative hardness of these problems earlier, one should probably choose $(i, j, k, l) = (2, 2, 1, 2)$ or $(2, 1, 2, 1)$.

Hence, if one wishes to have a scheme which is both provably secure and for which the security is related to a well studied problem, i.e. the $\mathtt{BDH}_{i,j,k}$ problem, then one needs to use a pairing for which there is an efficiently computable isomorphism.

2.2 The Boneh–Lynn–Shacham Signature Scheme

This section contains a similar analysis of the BLS short signature scheme [4] in the context of our abstract pairings above.

Given a pairing problem instance $\Gamma = (q, G_1, G_2, G_T, P_1, P_2, \hat{p})$ the BLS requires the following three elements of the groups G_1 and G_2 to be defined.

1. The public key of the user is defined to be $R = xP_i$, for $i \in \{1, 2\}$, and some secret value $x \in \mathbb{F}_q$.
2. The hash of a message is defined to be $Q_{\mathtt{M}} = H(\mathtt{M}) \in G_j$, where $j \in \{1, 2\}$ and H is a cryptographic hash function from $\{0, 1\}^*$ to G_j.
3. The signature is given $S \in G_k$, where either $S = xQ_{\mathtt{M}}$ or $S = \varphi(xQ_{\mathtt{M}})$.

From this set up we can instantly notice a number of points:

– Due to point 2 the group G_j must be randomly samplable, otherwise one would never be able to implement such a hash function.
– Due to point 3 we must have either $j = k$, or if there is an oracle to compute φ we may also have $(j, k) = (2, 1)$.
– To verify a signature we need to compute two pairings:
  - We need to compute the pairing of either either $Q_{\mathtt{M}}$ and R, or $Q_{\mathtt{M}}$ and $\varphi(R)$, or $\varphi(Q_{\mathtt{M}})$ and R. This implies that either $i \neq j$, or if there is an oracle to compute φ we may also have $i = j = 2$.
  - We also need to compute the pairing of either S and P_i, or S and $\varphi(P_i)$, or $\varphi(S)$ and P_i. This implies that either $k \neq i$, or if there is an oracle to compute φ we may also have $i = k = 2$.

This implies the following: If φ is not efficiently computable then to implement the BLS scheme one requires either $(i, j, k) = (1, 2, 2)$ or $(2, 1, 1)$. However, if φ

is efficiently computable then one can have any one of the following tuples for (i, j, k)
$$(1, 2, 2), (2, 1, 1), (2, 2, 1), (2, 2, 2).$$
Subject to the constraint that the group G_j is randomly samplable.

We now turn to the proof of the BLS signature scheme, in particular the simulation in $\mathtt{Game}_1$ of the security proof. The simulation of the hash function H works in roughly the same way as the simulation in the proof of the Boneh–Franklin encryption scheme mentioned earlier. The hash function $H(\mathtt{M}_t)$ is simulated via b_tP_j, which means the signature queries can be answered via b_tR, if $i = k$, or $\varphi(b_tR)$ if $i \neq k$ and $i = 2$.

Hence, the existing security proof of the Boneh–Franklin encryption scheme implies that:

- If φ is not efficiently computable then one needs to select $(i, j, k) = (2, 1, 1)$ and the security proof of the scheme is relative to the hardness of the $\mathtt{CDH}^{\varphi}_{2,1,1}$ problem. In other words, although the scheme in this instance may not require an efficiently computable φ the security proof is relative to an adversary which has oracle access to φ.
- If φ is effectively computable then one needs to select (i, j, k) from one of the following
$$(2, 1, 1), (2, 2, 1), (2, 2, 2).$$
The security proof is then relative to the hardness of $\mathtt{CDH}^{\varphi}_{i,j,k}$, except that now φ is not only given as oracle access to the adversary it is actually computable, hence the hardness is relative to the standard $\mathtt{CDH}_{i,j,k}$ problem.

2.3 Other Protocols

We now discuss a number of other protocols. Our list is not exhaustive and we point out just a number of problems.

In Boneh, Boyen and Shacham's paper [2] on short group signatures the authors present a group signature scheme in a pairing problem instance Γ. The main protocol is based on very different assumptions to those presented here, but the security proof relies on an adversary with access to the isomorphism φ. However, as an aside the authors of [2] present a simplified scheme which is based on a decisional variant of the $\mathtt{CDH}_{1,1,1}$ problem. However, for this problem to be hard the map φ^{-1} must be hard to compute. In general it does appear that φ^{-1} is hard to compute for all examples where one knows how to compute φ efficiently, except again for supersingular curves with distortion map.

In Boneh and Shacham's paper [5] on group signatures with verifier-local revocation one needs an isomorphism from G_2 to G_1 so as to be able to implement the scheme. Of all the papers which present their protocols in the asymmetric pairing setting, this is the only one we could find which actually required a computable isomorphism to be able to implement the protocol. This is not to say that one of the protocols described in the symmetric pairing setting in the literature when generalised to the asymmetric setting will not require a computable isomorphism. However, there is a problem with implementing the scheme of [5] in our context below. The scheme also requires a hash function with domain equal to G_2, which appears hard to construct for our representation of G_2 given in Section 4.

3 Pairings on Elliptic Curves

We now turn to actual concrete pairings based on (ordinary) elliptic curves. We assume that our elliptic curves are defined over a field $\mathbb{F}_p$ of characteristic greater than three. To aid computational efficiency we choose curves which have equations of the form

$$E : y^2 = x^3 - 3x + b$$

where $b \in \mathbb{F}_p$. We assume that $\#E(\mathbb{F}_p)$ is divisible by a large prime q and that there is a small integer k such that q divides $p^k - 1$, but q does not divide $p^l - 1$ for any integer $l < k$. Such curves are often called MNT curves as an algorithm for their construction was first given in [6].

Given such a curve the points of order q, i.e. the group $E[q]$, are defined over $\mathbb{F}_{p^k}$ and they form a two dimensional vector space over $\mathbb{F}_q$. As a basis of this vector space we can take P_1 and P_2 where P_1 is a point of order q in $E(\mathbb{F}_p)$ and P_2 is a point of order q which is defined over $\mathbb{F}_{p^k}$ but not $\mathbb{F}_p$.

The modified Tate pairing $\hat{t}$ is a bilinear pairing operating on the groups

$$\hat{t} : E[q] \times \big(E(\mathbb{F}_{p^k})/q \cdot E(\mathbb{F}_{p^k})\big) \longrightarrow \mathbb{F}_{p^k}^*.$$

The pairing is such that $\hat{t}(P, P) = 1$ for all P in $E[q]$; indeed, the pairing is trivial for all pairs which are linearly dependent when projected to the vector space $E[q]$. It is therefore common to restrict the first component to the points of order q in $E(\mathbb{F}_p)$ and the second component to a subgroup of $E(\mathbb{F}_{p^k})$ which is linearly independent of the points of order q in $E(\mathbb{F}_p)$. The question arises as to which subgroup should be chosen for the second component.

As mentioned earlier a number of protocols require a pairing for which there is a computable isomorphism from G_2 to G_1. In this situation the natural choice for G_1 is to be equal to $E(\mathbb{F}_p)[q]$ and the natural isomorphism from a subgroup of $E(\mathbb{F}_{p^k})$ to $E(\mathbb{F}_p)$ is given by the trace map

$$\varphi(P) = \mathtt{Tr}(P) = \sum_{\sigma \in \mathtt{Gal}(\mathbb{F}_{p^k}/\mathbb{F}_p)} P^\sigma.$$

However, if G_2 is a subgroup of $E(\mathbb{F}_{p^k})$ of order divisible by q which is distinct from G_1, then it appears difficult to construct a hash function with codomain G_2. Clearly by generating a random point of $E(\mathbb{F}_{p^k})$ and then multiplying by the cofactor $\#E(\mathbb{F}_{p^k})/q$ we will obtain a random element of $E[q]$. But the probability of this random element in $E[q]$ lying in the chosen subgroup G_2 is $1/q$.

We first present an optimisation trick for computing the pairing already described in [1]. If k is even, i.e. $k = 2d$, then we let χ denote an element of $\mathbb{F}_{p^d}$

which is a quadratic non-residue, i.e. we have

$$\mathbb{F}_{p^k} = \mathbb{F}_{p^d}[\sqrt{\chi}].$$

If we let d be odd then one can select $\chi \in \mathbb{F}_p$, which means that multiplication and division by χ can be performed in linear time.

We can now consider the quadratic twist of E over the field $\mathbb{F}_{p^d}$ given by

$$E' : \chi y^2 = x^3 - 3x + b.$$

Notice, we use this form of the quadratic twist of the curve rather than the form used in [1] so as to enable standard tricks to efficiently compute the group law on $E'(\mathbb{F}_{p^d})$.

There is an injective group homomorphism

$$\Phi : \begin{cases} E'(\mathbb{F}_{p^d}) \longrightarrow E(\mathbb{F}_{p^k}) \\ (x, y) \longmapsto (x, \sqrt{\chi}y). \end{cases}$$

Using a standard implementation trick, see [1], the Tate pairing of an element $P \in E(\mathbb{F}_p)$ with an element $Q \in \mathtt{Im}(\Phi)$ can be computed more efficiently than the pairing on an arbitrary element of $E(\mathbb{F}_{p^k})$.

The group order of $E'(\mathbb{F}_{p^d})$ is given by $N'_d = p^d + 1 + t_d$, whilst the group order of $E(\mathbb{F}_{p^d})$ is given by $N_d = p^d + 1 - t_d$. We note that since k is the smallest integer such that $p^k - 1$ is divisible by q we see that $p^d + 1$ must be divisible by q. However, on noticing that

$$N_d + N'_d = 2(p^d + 1),$$

and that N_d is also divisible by q, we see that N'_d is also divisible by q. In particular we see that $E'(\mathbb{F}_{p^d})$ contains a subgroup of order q. We let this subgroup be generated by P'_2.

Hence, $\langle \Phi(P'_2) \rangle$ is a subgroup of order q of $E(\mathbb{F}_{p^k})$ which is linear independent of the subgroup generated by P_1. Since in this paper G_1 will always be given by $\langle P_1 \rangle$, we shall denote $\langle \Phi(P'_2) \rangle$ by $G_1^{\perp}$.

If we pair an element of G_1 with an element of $E(\mathbb{F}_{p^k})$ with x-coordinate in $\mathbb{F}_{p^d}$ then our pairing computation can be performed efficiently [1]. The following lemma shows that there is essentially only one such subgroup G_2.

Lemma 1. *The only subgroups of $E(\mathbb{F}_{p^k})$ of order q which consist of elements with x-coordinates contained in $\mathbb{F}_{p^d}$ are given by G_1 and $G_1^{\perp}$.*

Proof: Let G be such a subgroup and let $P = (x, y)$ be a generator, then clearly all elements in G will have x-coordinate in $\mathbb{F}_{p^d}$ if and only if P does. If we set $P_1 = (1/k)\mathtt{Tr}(P) = (x_1, y_1)$ and $P_1^{\perp} = P - P_1 = (x_1^{\perp}, y_1^{\perp})$, then $\mathtt{Tr}(P_1^{\perp}) = \mathcal{O}$ and $P = P_1 + P_1^{\perp}$.

The element P_1 is contained in G_1 and $P_1^{\perp}$ is contained in $G_1^{\perp}$, see for instance [1, Corollary 1]. Looking at the addition formulae one sees that $x \in \mathbb{F}_{p^d}$ implies that $y_1 \cdot y_1^{\perp} \in \mathbb{F}_{p^d}$ and so y_1 is the conjugate of $y_1^{\perp}$, up to some multiple in

$\mathbb{F}_{p^d}$. However, since $y_1 \in \mathbb{F}_p$ and $y_1^{\perp} \in \mathbb{F}_{p^k} \setminus \mathbb{F}_{p^d}$, this would imply either $P_1 = \mathcal{O}$ or $P_1^{\perp} = \mathcal{O}$ and so the lemma follows. $\square$

Hence, if $G_2 = G_1^{\perp}$ then $\hat{t}$ is an efficiently computable pairing from $G_1 \times G_2$ to the subgroup of order q of $\mathbb{F}_{q^k}^*$. If however we select $G_2 \neq G_1^{\perp}$ then the speedups described in [1] no longer apply.

On the other hand, it is easy to see that $\mathtt{Tr}(P) = \mathcal{O}$ for all elements $P \in G_1^{\perp}$. Hence, if we select $G_2 = G_1^{\perp}$ then there does not exist a natural computable isomorphism from G_2 to G_1, no matter what choice of P_1 is made. However, as our earlier analysis points out, if no such computable isomorphism exists then either various pairing based schemes are not provably secure, or are provably secure relative to a new assumption (namely the adversary has access to an oracle to evaluate the isomorphism), in other cases the protocol simply cannot be computed without access to a computable isomorphism from G_2 to G_1.

4 Efficient Pairings and Computable Isomorphisms

In this section we present a new model for G_2 which allows us to achieve both an efficient pairing computation, plus an efficiently computable isomorphism from G_2 to G_1. However, this comes at the expense of G_2 not being randomly samplable. Thus, due to our discussion in an earlier section this restricts slightly the role of G_1 and G_2 in various protocols, in particular it will be impossible to construct hash functions with codomain equal to G_2. In this section we keep the elliptic curve notation of the previous section.

In setting up our system we define $G_1 = \langle P_1 \rangle$ as before. We then set

$$P_1' = [1/k \pmod{q}]P_1.$$

This implies that

$$\mathtt{Tr}(P_1') = kP_1' = P_1.$$

Our group G_2 we represent as the cyclic subgroup of $E(\mathbb{F}_p) \times E'(\mathbb{F}_{p^d})$ generated by (P_1', P_2') where P_2' is an element of $E'(\mathbb{F}_{p^d})$ of order q, as in the previous section. We shall denote this choice of G_2 by $\hat{G}_2$.

First we note that it appears impossible to construct a cryptographic hash function with codomain equal to $\hat{G}_2$, or to generate elements at random from $\hat{G}_2$ without creating a multiple of the generator, since it appears hard to generate completely at random two elements from different groups which have the same discrete logarithm with respect to the group generators without also computing the discrete logarithm.

However, it is easy to detect whether a pair (Q_1, Q_2) is actually an element of $\hat{G}_2$, given that $Q_1 \in E(\mathbb{F}_p)$ and $Q_2 \in E'(\mathbb{F}_{p^d})$. If $(Q_1, Q_2) \in \hat{G}_2$ then there is an $x \in \mathbb{F}_q$ such that $Q_1 = xP_1'$ and $Q_2 = xP_2'$. Hence, checking for inclusion in $\hat{G}_2$ comes down to testing whether

$$\hat{t}(Q_1, \Phi(P_2')) = \hat{t}(xP_1', \Phi(P_2')) = \hat{t}(P_1', \Phi(xP_2')) = \hat{t}(P_1', \Phi(Q_2)).$$

Note that these two Tate pairings are easy to compute since the second coordinate lies in the image of Φ and so the tricks of [1] can be applied.

We let the pairing of elements of G_1 and $\hat{G_2}$ be denoted by $\hat{p}$, and the isomorphism from $\hat{G_2}$ to G_1 be denoted by ϕ. We also define an injective homomorphism from $\hat{G_2}$ to $E(\mathbb{F}_{p^k})$ defined by

$$\xi : \begin{cases} \hat{G_2} & \longrightarrow \quad E(\mathbb{F}_{p^k}) \\ (Q_1, Q_2) & \longmapsto Q_1 + \Phi(Q_2). \end{cases}$$

We define for $P \in G_1$ and $Q = (Q_1, Q_2) \in \hat{G_2}$ that

$$\begin{aligned} \hat{p}(P, Q) &= \hat{t}(P, \xi(Q)) \\ &= \hat{t}(P, Q_1 + \Phi(Q_2)) \\ &= \hat{t}(P, Q_1) \cdot \hat{t}(P, \Phi(Q_2)) \\ &= \hat{t}(P, \Phi(Q_2)). \end{aligned}$$

The last equality follows since Q_1 is linearly dependent on P, so we can simply set $\hat{p}(P, Q) = \hat{t}(P, \Phi(Q_2))$. We note that since the second coordinate of the Tate pairing is in the image of Φ that the pairing $\hat{p}(\cdot, \cdot)$ can be computed efficiently. We also note that all the properties of a pairing in cryptography are satisfied by our pairing $\hat{p}(\cdot, \cdot)$.

To define the isomorphism from $\hat{G_2}$ to G_1 we use the trace map as in the previous section, which gives:

$$\begin{aligned} \phi(Q) &= \mathtt{Tr}(\xi(Q)) = \mathtt{Tr}(Q_1 + \Phi(Q_2)) \\ &= \mathtt{Tr}(Q_1) + \mathtt{Tr}(\Phi(Q_2)) = \mathtt{Tr}(Q_1) \\ &= kQ_1. \end{aligned}$$

This follows since the image of Φ always has trace zero and since Q_1 is in $E(\mathbb{F}_p)$ its trace is easily calculated by taking the kth multiple of Q_1. As another alternative one could take $\phi(Q) = Q_1$.

Hence, with this representation $G_2 = \hat{G_2}$, we achieve both an efficiently computable pairing and an efficiently computable isomorphism from G_2 to G_1.

5 Comparison

In this section we return to the Boneh–Franklin encryption scheme; a similar analysis applies to the other schemes mentioned earlier. We wish to explain the effect of the different choices for G_2, namely $G_1^{\perp}$ and $\hat{G_2}$ and a general G_2. A general G_2, by which we mean an arbitrary subgroup of order q of $E(\mathbb{F}_{q^{2d}})$ which is not equal to $G_1^{\perp}$ we shall denote by G_2'. The pairing can be computed in an efficient manner by using the following trick from [7]

$$\hat{p}(P, Q) = \hat{t}(P, Q - \sigma(Q))$$

where σ is the q^d-power Frobenius automorphism. Note, that the point $Q-\sigma(Q)$ has x-coordinate in $\mathbb{F}_{q^d}$ and so the pairing can be computed efficiently.

We summarise the main computational and bandwidth considerations in Table 1. Bandwidth, i.e. the size of elements of G_2, is measured in multiples of $\log_2 p$. We recall that when we have a computable isomorphism the $\mathtt{BDH}_{i,j,k}$ problem is equivalent to the $\mathtt{BDH}^{\varphi}_{i,j,k}$ problem.

G_2	Size of $g \in G_2$ in $\log_2 p$	Computable Isomorphism	Efficient Pairing	Hash to G_2	Underlying Hard Problem
$G_1^{\perp}$	d	N	Y	Y	$\mathtt{BDH}^{\varphi}_{2,1,2}$ or $\mathtt{coBDH}_{1,2}$
$\hat{G}_2$	$d+1$	Y	Y	N	$\mathtt{BDH}_{2,2,1}$ or $\mathtt{BDH}_{2,1,2}$
G_2'	$2d$	Y	Y	N	$\mathtt{BDH}_{2,2,1}$ or $\mathtt{BDH}_{2,1,2}$

Table 1. Comparison of different representations of the group G_2

Note, by reducing the hardness in the last two cases to $\mathtt{BDH}_{2,2,1}$ rather than $\mathtt{BDH}_{2,1,2}$ we can select ciphertext sizes which are smaller, i.e. $k = 1$ in our previous discussion of Section 2.1. However, this would require a mechanism to hash into the group G_2, which appears impossible. Hence, it is only practical to reduce the security to $\mathtt{BDH}_{2,1,2}$ which implies that ciphertexts are larger.

6 Another choice of G_2

One can obtain a group for G_2 which has both a computable homomorphism to G_1, efficient pairings and for which one can derive a hash function with codomain equal to G_2. However, one now has to select a group which is not cyclic of order q, but has exponent q.

It is easy to construct hash functions from an arbitrary bit string to both G_1 and $G_1^{\perp}$. As G_2 we now select the subgroup of $G_1 \times E'(\mathbb{F}_{p^d})$ of order q^2 and exponent q. There are two ways to represent elements of G_2 which give performance improvements.

1. In the first method we hold elements in G_2 as the pair of elements (Q_1, Q_2) where $Q_1 \in G_1$ and $Q_2 \in E'(\mathbb{F}_{p^d})$ where Q_2 is of order q. Representing Q in this way require roughly $(d+1)\log_2 p$ bits. Hashing into G_2 is now (theoretically) trivial, and computing the homomorphism to G_1 is obtained via
$$\phi((Q_1, Q_2)) = \mathtt{Tr}(Q_1 + \Phi(Q_2)) = kQ_1 \quad \text{ or } \quad \phi((Q_1, Q_2)) = Q_1.$$
The pairing is also computed in a similar way via
$$\hat{p}(P, (Q_1, Q_2)) = \hat{t}(P, Q_1 + \Phi(Q_2)) = \hat{t}(P, \Phi(Q_2)).$$

2. In the second method we hold elements in G_2 as a point $Q \in E(\mathbb{F}_{q^k})$, where Q has order q. Representing Q in this way requires roughly $2d \log_2 p$ bits. Hashing into G_2 is now (theoretically) trivial, and computing the homomorphism to G_1 is obtained via $\phi(Q) = \mathtt{Tr}(Q)$. The pairing can be computed in an efficient manner using the previous trick we presented for the group G_2'.

In both cases hashing is performed by generating an element at random on the appropriate curve (using a standard construction from a cryptographic hash function) followed by multiplication by the appropriate cofactor. This is rather an expensive operation in both cases. The cofactor in the first case has size roughly p^d/q, whilst in the second case the cofactor has size roughly $p^k/q = p^{2d}/q$. Hence, hashing into G_2 may become a computational bottleneck of any scheme.

In addition, the security proofs and protocols may now need to be slightly altered as almost all security proofs are written on the assumption that G_2 is cyclic of order q. Another problem is that there is a small probability, namely $1-1/q$, that the pairing of a non-trivial element of G_1 with a non-trivial element of G_2 may result in a trivial value for the pairing. This eventuality will also need to be considered in the various protocols which require pairings. We do not discuss these two issues further except to point out that for all protocols we have looked at such problems can be easily fixed with this choice of G_2.

7 Conclusion

We have described the various hard problems that occur in the asymmetric pairing setting and some of the relationships between them. In particular we have seen how the precise proof techniques used in pairing protocols can have subtle implications for the nature of the underlying hard problem in the asymmetric setting. This is particularly the case in relation to the existence of a computable isomorphism from the second to the first group.

We have also elaborated on efficient representations for the second group in the asymmetric setting which try to address some of the issues presented in the first part of the paper. However, the choice for the second group raises one of the following problems: it either affects whether one can implement certain protocols, whether the protocol is efficient and whether it has a security proof relative to a standard, non-relativised, problem. We hope our paper provides an impetus to researchers to find sets of pairing parameters which do not have these problems.

References

1. P.S.L.M. Barreto, B. Lynn and M. Scott. On the selection of pairing-friendly groups. *Selected Areas in Cryptography – SAC 2003*, Springer-Verlag LNCS 3006, 17–25, 2004.
2. D. Boneh, X. Boyen and H. Shacham. Short group signatures. *Advances in Cryptology – CRYPTO 2004*, Springer-Verlag LNCS 3152, 41–55, 2004.

3. D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, **32**, 586–615, 2003.
4. D. Boneh, B. Lynn and H. Shacham. Short signatures from the Weil pairing. *Advances in Cryptology – ASIACRYPT 2001*, Springer-Verlag LNCS 2248, 514–532, 2001.
5. D. Boneh and H. Shacham. Group signatures with verifier-local revocation. *11'th ACM conference on Computer and Communications Security – CCS*, 168–177, 2004.
6. A. Miyaji, M. Nakabayashi and S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **E84-A**, 1234–1243, 2001.
7. M. Scott. Faster Identity Based Encryption *Electronic Letters*, **40**, 861, 2004.