Unbalancing Pairing-Based Key Exchange Protocols

Michael Scott

Certivox Labs
mike.scott@certivox.com

Abstract. In many pairing-based protocols more than one party is involved, and some or all of them may be required to calculate pairings. Commonly it is the pairing calculation itself which takes most time. However some parties may be better equipped than others in terms of computational power. By exploiting the bilinearity property there are established ways to off-load the pairing calculation to an untrusted third party. Here we observe that this third party may in fact be one of the other participants in the protocol. In this way a protocol may be "unbalanced" by shifting the computational load from one participant to another, which may be an advantage in some circumstances. In this paper we focus on some simple key exchange protocols. Surprisingly we find that unbalancing a key exchange protocol can endow it with the property of full forward secrecy, even if it did not originally possess it. Finally we show that a new condition on the choice of pairing-friendly curve can help to minimize the overall computation.

1 Introduction

The bilinearity property of the pairing construct lies at the heart of its extraordinary flexibility, and is the reason that so many novel protocols and constructs are made possible using it. One downside is that the pairing calculation is relatively expensive compared to, say, a standard elliptic curve point multiplication. However Chevalier-Mames at al. [6] have shown that the pairing calculation can in all circumstances be safely offloaded to an untrusted device, and its calculation substituted by some point multiplications and exponentiations. Furthermore there is a range of choices for the optimal method to achieve the offloading, depending on the particular circumstances that apply.

Consider for example two-party pairing-based identity-based key exchange protocols, such as those considered by Chen, Cheng and Smart [4]. All of them require each of the parties to calculate a pairing. However all of them can in principle be "unbalanced" by using a suitable method from [6], resulting in a protocol where one party now calculates more than one pairing, and the other does not calculate any. But this has to done carefully if the full benefit is to be obtained.

2 Pairings

A type-1 pairing [7] on an elliptic curve is a mapping $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$, where $\mathbb{G}_1$ and $\mathbb{G}_T$ are groups of a prime order q, $\mathbb{G}_1$ are points on a pairing friendly elliptic curve over the base field $\mathbb{F}_p$, for p a prime or a prime power, and $\mathbb{G}_T$ are elements in the cyclotomic subgroup embedded in the finite extension field $\mathbb{F}_{p^k}$. Typically a distortion map in conjunction with a supersingular curve is exploited to implement a type-1 pairing. By "pairing-friendly" we mean that the curve has a relatively small embedding degree k, in fact restricted to be less than or equal to 6 for a supersingular curve.

A type-3 pairing is a mapping $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Again all groups are of a prime order q. For a typical pairing-friendly type-3 curve $\mathbb{G}_1$ are points on the curve over the base field $\mathbb{F}_q$, $\mathbb{G}_2$ are points on a degree t twist of the curve over a relatively small extension field $\mathbb{F}_{q^{k/t}}$, and $\mathbb{G}_T$ are elements in the cyclotomic subgroup embedded in the finite extension field $\mathbb{F}_{p^k}$.

The bilinearity property ensures that for both types of pairing $e(aP, Q) = e(P, aQ) = e(P, Q)^a$ where $a \in \mathbb{F}_q$. For type-1 pairings we have an additional symmetry property $e(P, Q) = e(Q, P)$

We will limit the scope of this research by only considering key exchange protocols that work on type-3 pairings. Our view would be that protocols which work only on a type-1 pairing, may not have a long-term future.

3 McCullagh-Barreto Key Exchange

We start with this simple protocol from McCullagh and Barreto [11]. A trusted authority (TA) has generated a master secret s, and made public the points $P \in \mathbb{G}_1$, $sP \in \mathbb{G}_1$, and $Q \in \mathbb{G}_2$ where P and Q are fixed points of order q. Alice and Bob have been issued with private keys $S_a = 1/(s + H(ID_a))Q \in \mathbb{G}_2$ and $S_b = 1/(s + H(ID_b))Q \in \mathbb{G}_2$ respectively, where ID_a and ID_b are Alice and Bob's identities, and $H(.)$ is a suitable hash function.

Alice - identity ID_a	**Bob - identity** ID_b
Generates random $x < q$	Generates random $y < q$
$ID_a \rightarrow$	$\leftarrow ID_b$
$b = H(ID_b)$	$a = H(ID_a)$
$A = x(bP + sP) \rightarrow$	$\leftarrow B = y(aP + sP)$
$k = e(xB, S_a)$	$k = e(yA, S_b)$

Table 1. McCullagh-Barreto

The shared key is $e(P, Q)^{xy}$. We assume that the protocol finishes with a standard key confirmation step, where the parties hash their mutual key with transmitted values and pass such values in both directions so that both parties are satisfied that they have the same key, without revealing it to an eavesdropper. We also assume that all values received are checked for the correct group membership [4], and that if this is not the case the protocol will fail.

The cost of this protocol is $\mathcal{P} + 3\mathcal{M}_1$, where $\mathcal{M}_1$ is the cost of a point multiplication in $\mathbb{G}_1$ and $\mathcal{P}$ is the cost of the pairing. The cost of the pairing clearly stands out as the major contributor.

Our next move is to consider ideas from [6], and offload the pairing calculation from Alice to Bob, which we will call the assisting party. While in [6] many scenarios are considered, here we will try to use the simplest method. So if the pairing $e(P, Q)$ is to be calculated, and P is public and Q a secret, then P and rQ (where r is randomly generated) are passed to the third party who responds with $e(P, rQ)$, and the pairing value is recovered as $e(P, rQ)^{1/r}$.

We also take the opportunity to simplify the protocol and combine some steps. Observe that the non-secret parameter to Alice's pairing B which was transmitted by Bob to Alice, can now be used directly by Bob. This brings us to our final version of the unbalanced protocol.

Alice - identity ID_a	**Bob - identity** ID_b
Generates random $r, x < q$	Generates random $y < q$
$ID_a \rightarrow$	$\leftarrow ID_b$
$b = H(ID_b)$	$a = H(ID_a)$
$A = x(bP + sP) \rightarrow$	$B = y(aP + sP)$
$R = rS_a \rightarrow$	
	$\leftarrow g = e(B, R)$
$k = g^{x/r}$	$k = e(yA, S_b)$

Table 2. Unbalanced McCullagh-Barreto Protocol

The cost of the protocol for Alice is now $\mathcal{E} + 2\mathcal{M}_1 + \mathcal{M}_2$, where $\mathcal{E}$ is the cost of an exponentiation in $\mathbb{G}_T$ and $\mathcal{M}_2$ is the cost of a point multiplication in $\mathbb{G}_2$. For Bob the cost is $2\mathcal{P} + 3\mathcal{M}_1$. Clearly, for Alice at least, unbalancing has worked.

This protocol suffers from vulnerability to a Key Compromise Impersonation attack (KCI) [15], whereby if Bob has stolen Alice's private key, not only can he masquerade as Alice to any third party, he can also pretend to be any third party and complete the key exchange with Alice. This is clearly undesirable.

3.1 Assisting party impersonation attack

What if Bob as the assisting party passes back to Alice a pairing value specially concocted so that Bob can complete the key exchange without having possession of a valid secret key?

Now there is clearly no point in Bob just transmitting a random value for g of the correct order. The joint key is calculated directly from this pairing value, and so if it is wrong the protocol will simply fail in a way indistinguisable from any other cause of protocol failure and this will be caught by the key confirmation step. However Bob might plausibly come up with values for C and D such that $g = e(C, D)^{x/r}$ is a value that he too can calculate without possessing his own legitimate private key.

However to cancel the effect of r, Bob must set $D = R$ (as this is the only multiple of r in his possession). Similarly the only value Bob knows, where he also knows its multiple times x (required as g is raised to the power of x) is $bP + sP$. So using these values Bob can succeed by sending $g = e(bP + sP, R)$, in which case Alice calculates $g^{x/r} = e(A, S_a)$, which Bob can also calculate if he knows Alices private key, without knowing S_b. But we are already aware that this protocol suffers from the KCI vulnerability, so this does not constitute a new attack.

4 McCullagh and Barreto second protocol

To counter the KCI vulnerability McCullagh and Barreto came up with a second protocol [11]. This is immune to a KCI attack. The secret keys are computed in the same way, and the protocol differs only in the way in which the final key is calculated.

Alice - identity ID_a	**Bob - identity** ID_b
Generates random $x < q$	Generates random $y < q$
$ID_a \rightarrow$	$\leftarrow ID_b$
$b = H(ID_b)$	$a = H(ID_a)$
$A = x(bP + sP) \rightarrow$	$\leftarrow B = y(aP + sP)$
$k = e(P, Q)^x.e(B, S_a)$	$k = e(P, Q)^y.e(A, S_b)$

Table 3. McCullagh-Barreto Protocol 2

The compute cost for each participant is $\mathcal{P} + \mathcal{E} + 2\mathcal{M}_1$, assuming that the pairing value $e(P, Q)$ is precalculated. See Table 3. The shared key this time is $e(P, Q)^{x+y}$. Next is our unbalanced version of this protocol (Table 4).

Alice - identity ID_a	**Bob - identity ID_b**
Generates random $r, x < q$	Generates random $y < q$
$ID_a \rightarrow$	$\leftarrow ID_b$
$b = H(ID_b)$	$a = H(ID_a)$
$A = x(bP + sP) \rightarrow$	$B = y(aP + sP)$
$R = rS_a \rightarrow$	$\leftarrow g = e(B, R)$
$k = (e(P, Q)^{xr}.g)^{1/r}$	$k = e(P, Q)^y.e(A, S_b)$

Table 4. Unbalanced McCullagh-Barreto Protocol 2

Now the cost for Alice is $2\mathcal{E} + 2\mathcal{M}_1 + \mathcal{M}_2$, and for Bob is $\mathcal{E} + 2\mathcal{P} + 2\mathcal{M}_1$.

4.1 Assisting party impersonation attack

Bob might plausibly come up with values for C and D such that $e(P, Q)^x.e(C, D)^{1/r}$ is a value that he too can calculate, without possessing his own legitimate private key. However to cancel the effect of r, again Bob must set $D = R$, which implies that Bob must know Alice's private key in order to succeed. But since KCI resistance is now assumed, this must be allowed. However Bob still cannot calculate a suitable value for C as he has no way of controlling Alice's random choice of x.

5 Chen-Kudla protocol

Next we consider protocol 2 from [5]. This protocol as described will only work with a type-1 pairing. However by moving from a peer-to-peer key exchange context to a client-server context, where client identities are mapped to $\mathbb{G}_1$ and servers to $\mathbb{G}_2$, we can implement it on a type-3 pairing. This context is of particular interest as it is quite likely that a client will have less compute power than a server, so unbalancing may be particularly useful. If we reasonably assume that each server and its clients are associated with a different TA master secret, then there is a redundancy in having both a unique server identity and a unique master secret associated with it. So instead we introduce a system wide constant $Q \in \mathbb{G}_2$ and differentiate between servers using only the uniqueness of the master secret. In this case the protocol is only "identity-based" as far as the clients are concerned, but we consider this as a realistic option. A client secret for Alice is calculated by the TA as $S_a = s.H_1(ID_a) \in \mathbb{G}_1$, where ID_a is Alice's identity, and $H(.)$ is a hash function which hashes an identity string to an elliptic curve point in $\mathbb{G}_1$. The server secret is calculated as $S_s = s.Q \in \mathbb{G}_2$,

Alice - identity ID_a	**Server**
Generates random $x < q$	Generates random $y < q$
$ID_a \rightarrow$	
$A = H(ID_a)$	$A = H(ID_a)$
$W_a = xA \rightarrow$	$\leftarrow W_s = yQ$
$k = e(S_a, W_s + xQ)$	$k = e(W_a + yA, S_s)$

Table 5. Chen-Kudla Protocol

The agreed key is $e(A, Q)^{s(x+y)}$. The cost for both parties is $\mathcal{P} + \mathcal{M}_1 + \mathcal{M}_2$.

To unbalance this protocol we assume that the client can precalculate the pairing $e(S_a, Q)$. Then the protocol proceeds as follows.

Alice - identity ID_a	**Server**
Generates random $r, x < q$	Generates random $y < q$
$ID_a \rightarrow$	
$A = H(ID_a)$	$A = H(ID_a)$
$R = rS_a \rightarrow$	
$W_a = xA \rightarrow$	$\leftarrow g = e(yR, Q)$
$k = (e(S_a, Q)^{xr}.g)^{1/r}$	$k = e(W_a + yA, S_s)$

Table 6. Unbalanced Chen-Kudla

Now the cost for Alice is $2\mathcal{E} + 2\mathcal{M}_1$, and the cost for the server is $2\mathcal{P} + 2\mathcal{M}_1$.

5.1 Assisting party impersonation attack

The analysis for this protocol is similar to that for the second McCullagh-Barreto protocol. A false Server with no secret S_s, even if it is in possession of Alice's private key, cannot control the value of g to force Alice to compute a key which he can also calculate, as he has no way of controlling Alice's random choice of x.

6 Forward Security

In each of the protocols described above extra measures must be taken to ensure support for full forward secrecy. Recall that full forward secrecy requires that if all communications are recorded and if sometime in the future all secrets are revealed (including TA secrets), then it should still not be possible to calculate the keys used to protect the past communications.

For the first McCullagh-Barreto protocol full forward secrecy is achieved by implementing on a type-3 pairing [11]. The second McCullagh-Barreto protocol does not support full forward secrecy at all, and is even vulnerable if TA secrets remain intact but both of Alice's and Bob's personal secrets are revealed. The Chen-Kudla protocol requires the explicit inclusion of a Diffie-Hellman component in order to achieve full forward secrecy.

However it turns out that by simply unbalancing these protocols as described above, this is sufficient to achieve full forward secrecy without taking any further action. The reason is quite simple in all cases – an ephemeral value required by Bob or the Server to calculate the pairing for Alice no longer needs to be transmitted to Alice. So it is no longer available to a potential attacker.

7 Precomputation

Virtually all pairing based protocols can benefit from extensive precomputation [12]. Fixed system-wide parameters can be precomputed on, as can individual private keys. For example for the McCullagh-Barreto protocols considered above the computation of $B = y(aP + sP)$ can be calculated as $B = (ya)P + y.sP$ where P and sP are system wide constants. Secret keys can be precomputed on, both for point multiplication and when they appear as a constant input to the pairing. In the second McCullagh-Barreto protocol and the Chen-Kudla protocol a significant speed-up can be obtained by precomputing multiples of a fixed pairing value.

8 Working in $\mathbb{G}_T$

Unbalancing will only pay off if we can optimise the exponentiation in $\mathbb{G}_T$, which now substitutes for the pairing calculation. Here we focus on the specific case of BN curves [2]. This popular family of pairing-friendly curves with embedding degree $k = 12$ is defined over the field $\mathbb{F}_p$, where

$$p = 36x^4 + 36x^3 + 24x^2 + 6x + 1$$

Further we initially consider a particular BN curve generated from the popular choice of parameter $x = -4080000000000001_{16}$, which produces a 254-bit prime p corresponding closely to the standard AES-128 bit level of security. This curve supports a pairing friendly group of prime order $q = p + 1 - t$, and $t = 6x^2 + 1$ is the trace of the Frobenius of the associated elliptic curve. The sparseness of x speeds up the pairing calculation.

8.1 Exponentiation

For Alice the pairing calculation has been swapped for a general exponentiation in $\mathbb{G}_T$. At first glance the best algorithm would appear to be that proposed by Galbraith and Scott [8]. However if the pairing value is to be immediately consumed to produce a protocol outcome, this approach may not be optimal. Instead it is faster to compress the pairing value as described in [13] to an element in $\mathbb{F}_{p^4}$, and then use trace-based methods as originally proposed for the XTR crypto scheme [14], a possibility not fully explored in [8]. For BN curves the XTR based method ([14] section 3.1) costs approximately 9 $\mathbb{F}_p$ multiplications per exponent bit. So an exponentiation in $\mathbb{G}_T$ can cost just twice as much as a point multiplication in $\mathbb{G}_1$, which is in sharp contrast with the much poorer ratio recently reported in Table 4 of [3]. The non-trace-based method used there (from [8]), despite having access to a better times-4 decomposition of the exponent, costs nearly twice as much. Furthermore the XTR method requires much less space, and so would appear to be a better choice for a constrained environment.

8.2 Sub-Group membership

There is a simple BN-curve-specific trick that somewhat speeds up a sub-group membership test (which tests $g^q \overset{?}{=} 1$) in $\mathbb{G}_T$. For a BN curve this test becomes

$$g^q = g^{(p+1-t)} \overset{?}{=} 1$$

which reduces to the condition that

$$g^p \overset{?}{=} g^{6x^2}$$

This last can be quickly tested using the Frobenius to calculate g^p, and an exponentiation to a very sparse 128-bit exponent.

8.3 Small Subgroup Attacks

Consider the extension field $\mathbb{F}_{p^{12}}$. In fact relatively few of its elements will be in the pairing friendly group of prime order q. The possible group orders in this field are the divisors of $p^{12} - 1$. We have

$$p^{12} - 1 = (p^6 - 1).(p^2 + 1).((p^4 - p^2 + 1)/q).q$$

where $p^4 - p^2 + 1 = \Phi_{12}(p)$ is the 12-th cyclotomic polynomial. In fact it is easy to check that an element g is of order $\Phi_{12}(p)$, as this can be done at almost no cost using the Frobenius. However how can we be sure that an element is in fact of order q and not of an order which is some other small divisor of $(p^4 - p^2 + 1)/q$, the component in the context of pairings often referred to as the "hard part of the final exponentiation"?

For the popular choice of $x = -4080000000000001_{16}$ many such small subgroups exist. In this case $(p^4 - p^2 + 1)/q$ has a factorisation

13.3793.29173.716953.569360689.C_{205}. So in this regard this is a particularly poor choice.

Consider now the protocol of Table 2 implemented using this curve. An entity purporting to be Bob, but without possession of Bob's secret key, transmits an element g of order 13 to Alice. Alice quickly confirms that this is indeed a member of the cyclotomic sub-group, and raises it to a large power to determine the key. But of course this just results in another element from the small group of size 13, all of which Bob has enumerated off-line. From the key confirmation hash, Bob can quickly determine which one Alice has calculated, and they use the same key. The protocol has failed.

Clearly this can be prevented by Alice first fully checking the order of g as shown above. But this has a small but significant cost associated with it.

9 "GT-Strong" Curves

As pointed out by Lim and Lee [10] in a different context, an alternative solution is to simply make sure that no small subgroups exist, in which case a membership test is redundant. This can, for example, be achieved by choosing the BN curve parameter such that $(p^4-p^2+1)/q$ is a prime, clearly much greater than q. In fact this is not hard to do, at the sacrifice of the extremely low Hamming weight for x. We call a pairing-friendly curve chosen with this extra feature "GT-strong". For example $x = -4000806000004081_{16}$ is a nice choice. This value can be used as a drop-in replacement for the value above in contexts where this feature is deemed to be desirable. We observe however that such a choice may negatively impact on some assumptions used to set pairing calculation records [1].

As an aside we note that for parameterised pairing friendly curves there is the possibility that the hard part of the final exponentiation represented as a polynomial in the parameter x may have a polynomial factorisation, and hence might never represent primes. For a BN curve this polynomial is

$$46656x^{12}+139968x^{11}+241056x^{10}+272160x^9+225504x^8+138672x^7+65448x^6+23112x^5+6264x^4+1188x^3+174x^2+6x+1$$

which is irreducible. This may not necessarily the case, but for the great majority of pairing friendly curves it is true, and certainly for all of the currently fashionable curves that we have tested.

10 Implementation

In the following table we show the cost in field multiplications for the considered protocols as implemented on a BN curve, with and without unbalancing. We fully exploit any possibilities for precomputation, precomputing and storing 256 multiples of constants where possible. In all cases we use the optimal R-ate pairing [9].

As can be seen unbalancing approximately halves the workload for one of the parties, while doubling it for the other. Note that we have not attempted to perform the modifications required to support full forward secrecy for the balanced versions of the protocols.

Table 7. Number of field multiplications

Protocol	Balanced	Unbalanced (Alice)	Unbalanced (Bob/Server)
McCullagh-Barreto 1	12400	5604	24278
McCullagh-Barreto 2	13081	7885	25026
Chen-Kudla	13955	6346	26040

References

1. D. F. Aranha, K. Karabina, P. Longa, C. H. Gebotys, and J. Lopez. Faster explicit formulas for computing pairings over ordinary curves. In *Eurocrypt 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 48–68. Springer-Verlag, 2011.
2. P.S.L.M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptology – SAC 2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer-Verlag, 2006.
3. J. Bos, C. Costello, and M. Naehrig. Exponentiating in pairing groups. Cryptology ePrint Archive, Report 2013/458, 2013. `http://eprint.iacr.org/2013/458`.
4. L. Chen, Z. Cheng, and N. Smart. Identity-based key agreement protocols from pairings. *Int. J. Inf. Sec.*, 6(4):213–241, 2007.
5. L. Chen and C. Kudla. Identity based authenticated key agreement from pairings. Cryptology ePrint Archive, Report 2002/184, 2002. `http://eprint.iacr.org/2002/184`.
6. B. Chevalier-Mames, J-S. Coron, N. McCullagh, D. Naccache, and M. Scott. Secure delegation of elliptic curve pairing. Cryptology ePrint Archive, Report 2005/150, 2005. `http://eprint.iacr.org/2005/150`.
7. S. Galbraith, K. Paterson, and N. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156:3113–3121, 2008.
8. S. Galbraith and M. Scott. Exponentiation in pairing-friendly groups using homomorphisms. In *Pairing 2008*, volume 5209 of *Lecture Notes in Computer Science*, pages 211–224. Springer-Verlag, 2008.
9. E. Lee, H. Lee, and C. Park. Efficient and generalized pairing computation on abelian varieties. *IEEE Trans. Information Theory*, 55:1793–1803, 2009.
10. C. H. Lim and P. J. Lee. A key recovery attack on discrete log-based schemes using a prime order subgroup. In *Crypto 1994*, volume 1294 of *Lecture Notes in Computer Science*, pages 249–263. Springer-Verlag, 1994.
11. N. McCullagh and P.S.L.M. Barreto. A new two-party identity-based authentication agreement. In *CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 262–274. Springer-Verlag, 2005.
12. M. Scott. On the efficient implementation of pairing-based protocols. In *Cryptography and Coding*, volume 7089 of *Lecture Notes in Computer Science*, pages 296–308. Springer-Verlag, 2011.

13. M. Scott and P. Barreto. Compressed pairings. In *Advances in Cryptology – Crypto' 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 140–156. Springer-Verlag, 2004. Also available from `http://eprint.iacr.org/2004/032/`.
14. M. Stam and A. K. Lenstra. Speeding up XTR. In *Asiacrypt 2005*, volume 2248 of *Lecture Notes in Computer Science*, pages 125–143. Springer-Verlag, 2001.
15. G. Xie. Cryptanalysis of Noel McCullagh and Paulo S.L.M Barretos's two party identity-based key agreement. Cryptology ePrint Archive, Report 2004/308, 2004. `http://eprint.iacr.org/2004/308`.