A Closer Look at Pairings

Luther Martin
June 3, 2008

Questions to answer

- What is the Tate pairing?
- What types of elliptic curves can be used to calculate pairings?
- How can we calculate pairings faster?
- What is the ate pairing?
- What are the security implications for this?

Pairings

- A special function called a *pairing* is needed to implement most IBE algorithms
- The benefits of IBE don't come for free – pairings are more expensive (computationally) that operations that are used in other traditional public-key algorithms
- Best optimized pairing is roughly comparable to an RSA decryption (within roughly 20 percent)
- Research is finding new ways to optimize pairing calculations, but there's still work to do
- The security implications of the optimizations are still not fully understood
  - Some require special structure which an attacker might or might not be able to take advantage of

Structures used and notation summary

- Finite field
  - Can add and multiply
  - If q is a prime number and k is a positive integer, there is only one finite field with q^k elements which we write GF(q^k)
    - Example: GF(7) = {0,1,2,3,4,5,6}
    - For $k > 1$ this gives us a way to multiply and divide vectors
- Multiplicative group of a finite field
  - Non-zero points in a finite field that we can multiply which we write as GF(q^k)*
  - Example GF(7)* = {1,2,3,4,5,6}

Structures used and notation summary

- Elliptic curve group
  - Points on an elliptic curve E: $y^2 = x^3 + ax + b$ that we can add using the usual connect-the-dots method
  - If the coefficients a and b of the elliptic curve E are from GF(q^k) we write E(GF(q^k)) for this

Voltage security

Bilinear mappings

- $e{:}G_1 \times G_2 \rightarrow G_T$
  - First input comes from G_1
  - Second input comes from G_2
  - Output is in G_T
  - So we might write $g = e(P,Q)$
- Usually think of G_1 and G_2 being elliptic curve groups so we write the operation there as addition
  - $P_3 = P_1 + P_2$
- Usually think of G_T as being in $GF(q^k)^*$ so we write the operation there as multiplication
  - $g_3 = g_1 \times g_2 = g_1 g_2$

Voltage security

Bilinearity

- A function e is *bilinear* if it's linear in both inputs
  - $e(aP,Q)=e(P,Q)^a$
  - $e(P,bQ)=e(P,Q)^b$
  - Can combine to get $e(aP,bQ)=e(P,Q)^{ab}$
- Can pull constants out of either input
- Note that we're writing some operations like they're addition and others as if they're multiplication
  - Addition in an elliptic curve group
  - Multiplication in a finite field

Pairings

- Just being bilinear isn't enough
- $f(x,y) = 1$ is bilinear but not very interesting or useful
- The trace map of $GF(q^k)$ over $GF(q)$ is bilinear but tricky to compute
- A mapping which is bilinear, non-degenerate and efficiently-computable is called a *pairing*
  - A "useful" bilinear mapping
- A very useful pairing is the Tate pairing
  - First cryptographic use was actually to attack elliptic curve systems (MOV reduction, 1993)
  - Now it's been rehabilitated

Calculating the Tate pairing

- Idea: to calculate $e(P,Q)$, do the following:
  - Find a rational function that's defined by P
  - Evaluate this function at Q
- If the point P is of order p, we can get the Tate pairing like this:

```
f = 1
for i = 1 to p
  f = f * f_i(Q) // we get f_i from iP
end for
```

9 Voltage

Miller's algorithm

- For cryptographic uses, p is typically 2^{160} or greater
  - Iterating from 1 to 2^{160} will take essentially forever
- We can also calculate the Tate pairing using a double-and-add technique
  - Iterate over the binary expansion of p
    - Repeatedly double
    - Add when the bit of p that we're at is a '1'
    - Accumulate the factors of the rational function as we do
  - Loop 160 times instead of 2^{160}
- This gives us Miller's algorithm (1986)
- A straightforward implementation is fairly slow

10 Voltage

Making Miller's algorithm faster

- It's possible to speed up Miller's algorithm using a number of computational tricks
- Some of these require the creation of pairings that are much like the Tate pairing
  - The ate pairing is the most important
  - Shorter version of "Tate"
- If $e(P,Q)$ is the Tate pairing, the ate pairing calculates $e(P,Q)^r$ for some integer r
- This requires special structure
- This structure lets you decrease the length of the loop in Miller's algorithm
- This structure may or may not make its use cryptographically weak (probably not)
- More research is probably needed in this area

Embedding degree

- Because we need to multiply to calculate it, the Tate pairing requires calculations to be done in a *field*
- We can only add in G_1
  - We want to be able to multiply to implement Miller's algorithm
  - Solution: embed G_1 in $GF(q^k)^*$ where multiplication *is* defined
  - The embedding degree (MOV degree) k is the degree of the extension field where we can do this
- This means that we have vectors with k components, each one an element of $GF(q)$
- We need for k to be relatively small to make this practical
- Most elliptic curve groups have embedding degrees that are much too big
  - Roughly the same as the order of G_1
  - Ouch: $|G_1| = 2^{160}$ means roughly 2^{160} coordinates

Low embedding degree

- Not many elliptic curves give us groups with a low embedding degree
- A few types that do:
  - Supersingular curves (k = 1, 2, 3, 4, 6)
    - k = 2 the most useful
    - $y^2 = x^3 + 1$; $q \equiv 2 \bmod 3$ (easier to hash to point)
    - $y^2 = x^3 + x$; $q \equiv 1 \bmod 3$ (faster pairing calculation)
  - MNT curves (k = 3, 4, 6)
  - BN curves (k = 12)
- A low embedding degree makes a MOV attack possible
  - If calculating a pairing is feasible then an MOV attack is also feasible
- So we need to account for this when we pick parameters

MOV attack

- Suppose that we want to find the discrete logarithm of aP
- Suppose that we have a pairing e that we can use
- Say $e(P,Q) = g$
- Note that $e(aP,Q) = e(P,Q)^a = g^a$
- We can find the discrete log a from either aP or g^a
- aP might be in elliptic curve group and g^a in a finite field
  - Embedding degree $k = 2$ for $E(GF(q))$ means that we can calculate discrete logs in $GF(q^2)^*$
    - Index calculus with 320 bits (weak) instead of Pollard's rho with 160 bits (strong)

MOV attack

- If you can implement a pairing, you can do an MOV attack
- You need to pick parameters so that this doesn't matter
- In the previous example we could calculate discrete logs in either $GF(q^k)^*$ of order 2^{320} or a group G_1 of order 2^{160}
- If we make q big enough so that the $GF(q^k)^*$ has order 2^{1024}, we're done
  - 512-bit q instead of 160-bit q

Security considerations

- With supersingular curves, the embedding degree is always low ($k \leq 6$)
  - This has been fairly well studied
  - But they certainly "sound weak," don't they?
  - Bad reputation because of MOV attack
- With ordinary curves, additional structure is needed to get a low embedding degree
  - This has not been well studied
  - More research is needed
- The conservative choice for implementing a pairing-based algorithm is to use a supersingular curve

Underlying computational problems

- Diffie-Hellman problem
  - Given g, g^a, g^b, find g^{ab}
  - We assume that we need to calculate discrete log of either g^a or g^b to do this
- Bilinear Diffie-Hellman problem
  - Given P, aP, bP, cP, find $e(P,P)^{abc}$
  - Note that we can also calculate $e(P,aP) = g^a$ (also g^b, g^c)
  - We assume that we need to calculate the discrete logs of aP, bP, cP, g^a, g^b, g^c to do this

Picking parameters

- To attack IBE systems with a pairing $e:G_1 \times G_2 \rightarrow G_T$ whose security depends on the bilinear Diffie-Hellman problem, we assume that you need to calculate a discrete log in G_1, G_2, or G_T
  - Just like we assume that calculating discrete logs is the only way to solve the Diffie-Hellman problem
- G_1 and G_2 are easy to understand if they're elliptic curve groups of prime order
  - Just look at SP 800-57 to see how big they need to be for a particular security level
- G_T is slightly more complicated
  - It's the same order as G_1 and G_2, but it's in a finite field
  - We can find discrete logs in G_T in two different ways

Security in G_T

- If $e{:}G_1{\times}G_2{\rightarrow}G_T$ is a pairing, the output is in $GF(q^k)^*$
- We can calculate discrete logs in G_T in two ways
  - Pollard's rho in G_T
  - Index calculus in $GF(q^k)^*$
- We need to pick parameters so that both of these are difficult enough
  - Just like with Diffie-Hellman with $GF(p)$ replaced by $GF(q^k)$

Voltage security

Parameter sizes

- Example: 80 bits of security
  - Need $p = |G_1| \geq 2^{160}$
  - Need $|GF(q^k)^*| \geq 2^{1024}$ or $|GF(q)^*| \geq 2^{1024/k}$
  - If $k = 2$, need 512-bit q ($1024 = 2 \times 512$)
    - A supersingular curve can be used to implement this
  - If k = 6, need 171-bit q (rounded up from 1024 / 6 = 170.67) and $|GF(q^k)^*| = 2^{1026}$ (6 x 171 = 1026)
    - An MNT curve can be used to implement this

Voltage security

Parameter sizes

- Example: 128 bits of security
  - $|G_1| \geq 2^{256}$, need $|GF(q^k)^*| \geq 2^{3072}$
  - If $k = 12$, need 256-bit q ($3072 = 12 \times 256$)
  - A BN curve can be used to implement this

Voltage security

Parameters to get comparable strengths

Bits of security	FFC	ECC	PBC
80	$L = 1024$ $N = 160$	$f = 160\text{-}223$	$f = 160\text{-}223$ $k \times L \geq 1024$
112	$L = 2048$ $N = 224$	$f = 224\text{-}255$	$f = 224\text{-}255$ $k \times L \geq 2048$
128	$L = 3072$ $N = 256$	$f = 256\text{-}333$	$F = 256\text{-}333$ $k \times L \geq 3072$
192	$L = 7680$ $N = 334$	$f = 384\text{-}511$	$F = 384\text{-}511$ $k \times L \geq 7680$
256	$L = 15360$ $N = 512$	$f = 512+$	$F = 512+$ $k \times L \geq 15360$

Voltage security

Selecting parameters

- Select bit security level
  - Determines size of p, k x $log_2 q$
- Select curve type
  - Supersingular curve or ordinary curve
    - Select curve family if ordinary
- Select curve
- Select appropriate pairing
- Select q
- Find p so that $E(GF(q))$ has a subgroup of order p
  - Should be a Solinas prime for best efficiency

23 Voltage security

Summary

- What is the Tate pairing?
- What types of elliptic curves can be used to calculate pairings?
- How can we calculate pairings faster?
- What is the ate pairing?
- What are the security implications for this?

24 Voltage security


Voltage
security