On the Static Diffie-Hellman Problem on Elliptic Curves over Extension Fields

Robert Granger

rgranger@computing.dcu.ie
Claude Shannon Institute, UCD and DCU, Ireland

RHUL, 18th November 2010
Outline

1 Background and Motivation
   - The Static Diffie-Hellman Problem
   - Related Assumptions

2 Main Algorithm and Results
   - Algorithm Overview
   - Potentially Vulnerable Curves
   - Simulation results

3 A new method for binary curves
Let $\mathbb{G}$ be a cyclic group of prime order $r$ with generator $g$. 
Diffie-Hellman Key Agreement

Let $\mathbb{G}$ be a cyclic group of prime order $r$ with generator $g$.

- Alice chooses $x \overset{R}{\leftarrow} \mathbb{Z}_r$, computes $g^x$ and sends to Bob.
- Bob chooses $y \overset{R}{\leftarrow} \mathbb{Z}_r$, computes $g^y$ and sends to Alice.
- Alice computes $\left( g^y \right)^x$, Bob computes $\left( g^x \right)^y$ to give shared secret $g^{xy}$. 

A fundamental security requirement of DH Key Agreement is that the Computational Diffie-Hellman problem should be hard:

**Definition (CDH):**
Given $g$ and random $g^x$ and $g^y$, find $g^{xy}$. 

R. Granger

On the Static DHP on Elliptic Curves over Extension Fields
Diffie-Hellman Key Agreement

Let $\mathbb{G}$ be a cyclic group of prime order $r$ with generator $g$.

- Alice chooses $x \xleftarrow{R} \mathbb{Z}_r$, computes $g^x$ and sends to Bob
- Bob chooses $y \xleftarrow{R} \mathbb{Z}_r$, computes $g^y$ and sends to Alice
- Alice computes $(g^y)^x$, Bob computes $(g^x)^y$ to give shared secret $g^{xy}$

A fundamental security requirement of DH Key Agreement is that the Computational Diffie-Hellman problem should be hard:

**Definition**

(CDH): Given $g$ and random $g^x$ and $g^y$, find $g^{xy}$
The Static Diffie-Hellman Problem (Static DHP)

Suppose to minimise her exponentiation cost in multiple DH key agreements Alice repeatedly reuses \( x = d \).
The Static Diffie-Hellman Problem (Static DHP)

Suppose to minimise her exponentiation cost in multiple DH key agreements Alice repeatedly reuses $x = d$.

**Definition**

\((\text{Static DHP}_d)\): Given fixed $g$ and $g^d$, and random $g^y$, find $g^{dy}$
The Static Diffie-Hellman Problem (Static DHP)

Suppose to minimise her exponentiation cost in multiple DH key agreements Alice repeatedly reuses $x = d$.

Definition

$(\text{Static DHP}_d)$: Given fixed $g$ and $g^d$, and random $g^y$, find $g^{dy}$

- Set of problem instances in Static DHP is a tiny subset of CDH problem instances
- Not $a \text{ priori}$ clear that these instances should be hard, even if CDH is hard
- Hence Static DHP$_d$ better models the security of this scenario than CDH does
The Static DHP - inception and 1st result

Introduced by Brown and Gallant in 2004, who gave a reduction from the DLP for $d$ to the Static DHP$_d$
The Static DHP - inception and 1st result

Introduced by Brown and Gallant in 2004, who gave a reduction from the DLP for $d$ to the Static DHP$_d$

- Hence if the DLP for $d$ is hard, then so is the Static DHP$_d$
The Static DHP - inception and 1st result

Introduced by Brown and Gallant in 2004, who gave a reduction from the DLP for $d$ to the Static DHP$_d$

- Hence if the DLP for $d$ is hard, then so is the Static DHP$_d$
- Equivalently, given access to a Static DHP$_d$ oracle, one can find the associated DLP $d'$
The Static DHP - inception and 1st result

Introduced by Brown and Gallant in 2004, who gave a reduction from the DLP for \( d \) to the Static DHP\(_d\)

- Hence if the DLP for \( d \) is hard, then so is the Static DHP\(_d\)
- Equivalently, given access to a Static DHP\(_d\) oracle, one can find the associated DLP \( d \)

**Definition**

(Static DHP\(_d\) oracle): Let \( G \) be a cyclic group of prime order \( r \), written additively. For a fixed base element \( P \in G \) and a fixed element \( Q \in G \) let \( d \in \mathbb{Z}_r \) be such that \( Q = dP \). Then a Static DHP\(_d\) oracle (w.r.t. \((G, P, Q)\)) computes the function \( \delta : G \rightarrow G \) where

\[
\delta(X) = dX
\]
A Static DHP$_d$ algorithm is said to be oracle-assisted if during an initial learning phase, it can make a number of Static DHP$_d$ queries, after which, given a previously unseen challenge element $X$, it outputs $dX$. 
A Static DHP$_d$ algorithm is said to be *oracle-assisted* if during an initial learning phase, it can make a number of Static DHP$_d$ queries, after which, given a previously unseen challenge element $X$, it outputs $dX$.

**Theorem**

Let $r = uv + 1$. Then $d$ can be found with $u$ calls to a Static DHP$_d$ oracle, and off-line computational work of about $(\sqrt{u} + \sqrt{v})$ group operations.
The complexity of the attack is minimised when $u \approx r^{1/3}$.

Depending on the factorisation of $r - 1$, can lead to a real attack which is quicker than solving the DLP.
DLP to Static DHP\textsubscript{d} reduction

- The complexity of the attack is minimised when $u \approx r^{1/3}$
- Depending on the factorisation of $r - 1$, can lead to a real attack which is quicker than solving the DLP

Brown and Gallant showed that a system entity acts as a Static DHP\textsubscript{d} oracle, transforming their reduction into a DLP solver, for the following protocols:

- textbook El Gamal encryption
- Ford-Kaliski key retrieval
- Chaum-Van Antwerpen’s undeniable signatures
Static $DHP_d$ example: textbook El Gamal

- Alice has public key $g^d$. To encrypt a message $m$, Bob picks a random $x \leftarrow \mathbb{Z}_r$ and computes
  \[
  c = (c_1, c_2) = (g^x, mg^{dx})
  \]

- To decrypt Alice computes $m = c_2/c_1^d$. So if one can compute $g^{dx}$ for any $g^x$ one can decrypt.

- Furthermore, in a chosen-ciphertext attack an adversary has access to a decryption oracle.

- If adversary chooses $c = (g^x, c_2)$ the decryption oracle returns $m = c_2/g^{dx}$.

- Adversary computes $g^{dx} = c_2/m$, which solves the Static $DHP_d$ for instance $g^x$, giving a Static $DHP_d$ oracle.
DLP to $l$-Strong DHP reduction

Attack was rediscovered by Cheon in 2006, when the requisite information is provided in the guise of the $l$-Strong DHP:

**Definition**

$l$-Strong Diffie-Hellman problem: Given $P$ and $d^i P$ in $\mathbb{G}$ for $i = 1, 2, \ldots, l$, compute $d^{l+1} P$.
The Static Diffie-Hellman Problem

Attack was rediscovered by Cheon in 2006, when the requisite information is provided in the guise of the \(l\)-Strong DHP:

**Definition**

\(l\)-Strong Diffie-Hellman problem: Given \(P\) and \(d^iP\) in \(G\) for \(i = 1, 2, \ldots, l\), compute \(d^{l+1}P\)

- Cheon also formulated an algorithm when \(l \mid (r + 1)\)
- Both can be seen as using the DLP to DHP reduction due to den Boer, Maurer, Wolf et al, but with limited access to a limited CDH oracle
Delayed Target DHP

Freeman [05] — ‘Pairing-based identification schemes’
Delayed Target DHP

Freeman [05] — ‘Pairing-based identification schemes’

**Definition**

A solver is given initial access to a Static DHP$_d$ oracle for the element $Q = dP \in \mathbb{G}$; when the oracle is removed, the solver is given a random challenge $X \in \mathbb{G}$ and must solve the CDH for input $(Q, X)$, i.e., output $dX$. 
**Delayed Target DHP**

Freeman [05] — ‘Pairing-based identification schemes’

**Definition**

A solver is given initial access to a Static DHP$_d$ oracle for the element $Q = dP \in G$; when the oracle is removed, the solver is given a random challenge $X \in G$ and must solve the CDH for input $(Q, X)$, i.e., output $dX$.

- Situation identical to oracle-assisted Static DHP
- Security of scheme equivalent to Delayed Target DHP
In ‘Another look at non-standard discrete log and Diffie-Hellman problems’ [07], Koblitz and Menezes studied a set of problems in the Jacobian of small genus hyperelliptic curves
Results of Koblitz and Menezes

In ‘Another look at non-standard discrete log and Diffie-Hellman problems’ [07], Koblitz and Menezes studied a set of problems in the Jacobian of small genus hyperelliptic curves

- **Delayed Target** DLP/DHP, **One-More** DLP/DHP, and DLP1/DHP1

- Using ‘Index Calculus’ or Brown/Gallant/Cheon show that some are easier than DLP - hardness separation

- Argue that problems which are either interactive or have complicated inputs can produce weaknesses

- Conclude that security assurances provided by such assumptions should be reassessed/are difficult to assess
Assuming index calculus methodology applies, Koblitz-Menezes used the following simple algorithm:

- Construct a factor base $\mathcal{F}$ over which a non-negligible proportion of group elements factor
- Call the Static DHP$_d$ oracle $\delta$ on all $f \in \mathcal{F}$
- For a target element $X$ attempt to write random multiples $aX$ as a sum of elements of $\mathcal{F}$, i.e., $aX = P_1 + \cdots + P_n$
- Then $dX = (a^{-1} \mod r)(\delta(P_1) + \cdots + \delta(P_n))$

Applied algorithm to finite fields and small genus hyperelliptic curves — resulting in a hardness separation from DLP
Let $H(\mathbb{F}_q)$ be a genus $g$ hyperelliptic curve and $\text{Jac}_H(\mathbb{F}_q)$ its Jacobian.

- Let $\mathcal{F}$ be a proportion $q^\alpha$ of degree one divisors for $0 < \alpha \leq 1$
- Call the Static DHP$_d$ oracle for $Q = dP$ for all $D \in \mathcal{F}$
- Prob. random $aX$ factors over $\mathcal{F}$ is $q^{g(\alpha-1)}/g!$
- Hence expected number of trials to obtain an $\mathcal{F}$-smooth element $aX$ is $q^{g(1-\alpha)}g!$
- Balancing this with the oracle calls gives

$$\alpha = \frac{g + \log_q g!}{g + 1} \approx 1 - \frac{1}{g+1}$$
For DLP, there are four basic variants:

- Gaudry (2000): basic index calculus — $O(q^2)$
- Harley (2000): reduce factor base — $O(q^{2-2/(g+1)})$
- Thériault (2003): large-prime variation — $O(q^{2-2/(g+1/2)})$
- GTTD (2007): double large-prime variation — $O(q^{2-2/g})$

The *Delayed Target* DHP algorithm is $O(q^{1-1/(g+1)})$ — the square root of Harley’s algorithm:

- No linear algebra
- Only one relation so can only balance the two stages
For DLP, there are four basic variants:

- Gaudry (2000): basic index calculus — $O(q^2)$
- Harley (2000): reduce factor base — $O(q^{2-2/(g+1)})$
- Thériault (2003): large-prime variation — $O(q^{2-2/(g+1/2)})$
- GTTD (2007): double large-prime variation — $O(q^{2-2/g})$

The *Delayed Target* DHP algorithm is $O(q^{1-1/(g+1)})$ — the square root of Harley’s algorithm:

- No linear algebra
- Only one relation so can only balance the two stages

**Question:** For $g = 1$ have $O(q^{1/2})$, so can we do better?
Joux, Naccache and Thomé [08] showed that initial access to an \( e \)-th root oracle in RSA enables later \( e \)-th root computations — faster than one can factor the modulus

- Ports easily over to Static DHP \(_d\) in \( \mathbb{F}_q \) (+Lercier [09])
- The \( L_{q^n}(1/3, \sqrt[3]{x}) \) complexities of the JLNT algorithm are

<table>
<thead>
<tr>
<th>variant</th>
<th>oracle access</th>
<th>learning phase</th>
<th>post-learning phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFS</td>
<td>4/9</td>
<td>-</td>
<td>4/9</td>
</tr>
<tr>
<td>NFS-HD</td>
<td>48/91</td>
<td>384/91</td>
<td>384/91</td>
</tr>
<tr>
<td>NFS</td>
<td>4/9</td>
<td>32/9</td>
<td>3</td>
</tr>
</tbody>
</table>

- Each is faster than the DLP in the corresponding fields
Oracle-assisted Static DHP for elliptic curves?

- Problem is that one needs a factor base to beat the Brown/Gallant/Cheon complexity
- For ECs over $\mathbb{F}_p$, currently no known useful factor base
Oracle-assisted Static DHP for elliptic curves?

- Problem is that one needs a factor base to beat the Brown/Gallant/Cheon complexity.
- For ECs over $\mathbb{F}_p$, currently no known useful factor base.
- Basic insight is that for ECs over extension fields, one already has a native factorisation via Gaudry/Semaev ECDLP algorithm $\Rightarrow$ can use the KM methodology.
Oracle-assisted Static DHP for elliptic curves?

- Problem is that one needs a factor base to beat the Brown/Gallant/Cheon complexity
- For ECs over $\mathbb{F}_p$, currently no known useful factor base
- Basic insight is that for ECs over extension fields, one already has a native factorisation via Gaudry/Semaev ECDLP algorithm $\Rightarrow$ can use the KM methodology
- Obvious in hindsight and could have been observed in 2004 when Gaudry had his idea
Oracle-assisted Static DHP for elliptic curves?

- Problem is that one needs a factor base to beat the Brown/Gallant/Cheon complexity.
- For ECs over $\mathbb{F}_p$, currently no known useful factor base.
- Basic insight is that for ECs over extension fields, one already has a native factorisation via Gaudry/Semaev ECDLP algorithm $\implies$ can use the KM methodology.
- Obvious in hindsight and could have been observed in 2004 when Gaudry had his idea.
- Basic observation made independently by Joux and Vitse.
Semaev’s summation polynomials

Let $E : Y^2 = X^3 + aX + b$, over a field $\mathbb{F}_q$ with $\text{char}(\mathbb{F}_q) > 3$.

For $m \geq 2$ define $f_m = f_m(X_1, \ldots, X_m) \in \mathbb{F}_q[X_1, \ldots, X_m]$ by the following property:

- for $x_1, \ldots, x_m \in \overline{\mathbb{F}}_q$, $f_m(x_1, \ldots, x_m) = 0$ is equivalent to
  
  $$\exists y_1, \ldots, y_m \in \overline{\mathbb{F}}_q \mid (x_i, y_i) \in E \text{ and } (x_1, y_1) + \cdots + (x_m, y_m) = \mathcal{O} \in E(\overline{\mathbb{F}}_q)$$

- We have $f_2(X_1, X_2) = X_1 - X_2$, and $f_3(X_1, X_2, X_3) =$
  
  $$(X_1 - X_2)^2 X_3^2 - 2((X_1 + X_2)(X_1X_2 + a) + 2b)X_3$$
  
  $$+ ((X_1X_2 - a)^2 - 4b(X_1 + X_2))$$
Semaev’s summation polynomials

- In general, for any $m \geq 4$, and $m - 3 \geq k \geq 1$,
  
  $$f_m(X_1, \ldots, X_m) =$$
  
  $$\text{Res}_X(f_{m-k}(X_1, \ldots, X_{m-k-1}, X), f_{k+2}(X_{m-k}, \ldots, X_m, X))$$

- Degree of $f_m$ in each $X_i$ is $2^{m-2}$ for $m \geq 3$.
- In the case prime fields, a natural factor base is
  
  $$\mathcal{F} = \{ P = (x, y) \in E \ s.t. \ x < p^{1/m} \}$$

- However no known way to efficiently find such small roots $x_1, \ldots, x_m$ of $f_{m+1}(x_1, \ldots, x_m, x_R) = 0$ corresponding to
  
  $$R = P_{i_1} + \cdots + P_{i_m}$$

- For $m \geq 5$ would give sub-square-root DLP algorithm
Assume now that $E$ is over a degree $n$ extension $\mathbb{F}_{q^n}$.

- Fix a poly basis $\{t^{n-1}, \ldots, t, 1\}$ for $\mathbb{F}_{q^n}/\mathbb{F}_q$
- Define $\mathcal{F} = \{P = (x, y) \in E(\mathbb{F}_{q^n}) \text{ s.t. } x \in \mathbb{F}_q\}$
- Note $|\mathcal{F}| \approx q$
- Observe that $f_{n+1}(x_1, \ldots, x_n, x_r) = 0$ now has $n$ components:

$$f_{n+1,0} + f_{n+1,1} t + \cdots + f_{n+1,n-1} t^{n-1} = 0 \in \mathbb{F}_{q^n}$$

- System of $n$ equations over $\mathbb{F}_q$ in $n$ variables in $\mathbb{F}_q$
- Solved via resultants, or Grobner basis computation
ECDLP complexity with Gaudry/Semaev

- Decomposition complexity $O(Poly(2^{n(n-1)}))$
- Decomposition probability is $1/n!$
- For fixed $n$, $q \rightarrow \infty$, complexity is $O(q^2)$, rho is $O(q^{n/2})$
- Using double large-prime variation reduces to $O(q^{2-2/n})$
- Works for all curves over any extension field, even of prime extension degree
- Computationally far more intensive than Weil descent
- Subexponential attack for a large class of fields (Diem)

$$e^{O((\log q^n)^{2/3})}$$
Oracle-assisted Static DHP Algorithm in full

- Define $\mathcal{F} = \{P = (x, y) \in E(\mathbb{F}_{q^n}) \text{ s.t. } x \in \mathbb{F}_q\}$
- For all $P \in \mathcal{F}$ compute $\delta(P) = dP$
- For a given $R \in E(\mathbb{F}_q)$ add random linear combinations $P_r$ of elements of $\mathcal{F}$ to $R$ until it can be written

$$R + P_r = P_1 + \cdots + P_n \iff f_{n+1}(x_1, \ldots, x_n, x_R) = 0$$

- Then $dR = \delta(P_1) + \cdots + \delta(P_n) - \delta(P_r)$
Heuristic Result 1. For any elliptic curve $E(\mathbb{F}_{q^n})$, by making $O(q)$ queries to a Static DHP$_d$ oracle during an initial learning phase, for fixed $n > 1$ and $q \to \infty$, an adversary can solve any further instance of the Static DHP$_d$ in time $O(\text{Poly}(\log q))$. 

Can reduce the factor base à la Harley:

Heuristic Result 2. For any elliptic curve $E(\mathbb{F}_{q^n})$, by making $O(q^{1 - 1/n + 1})$ queries to a Static DHP$_d$ oracle during an initial learning phase, for fixed $n > 1$ and $q \to \infty$, an adversary can solve any further instance of the Static DHP$_d$ in time $\tilde{O}(q^{1 - 1/n + 1})$. 

Can also obtain subexponential algorithm à la Diem.
Heuristic Result 1. For any elliptic curve $E(\mathbb{F}_{q^n})$, by making $O(q)$ queries to a Static DHP$_d$ oracle during an initial learning phase, for fixed $n > 1$ and $q \to \infty$, an adversary can solve any further instance of the Static DHP$_d$ in time $O(\text{Poly}(\log q))$.

Can reduce the factor base à la Harley:
Heuristic Result 1. For any elliptic curve $E(\mathbb{F}_{q^n})$, by making $O(q)$ queries to a Static DHP$_d$ oracle during an initial learning phase, for fixed $n > 1$ and $q \to \infty$, an adversary can solve any further instance of the Static DHP$_d$ in time $O(\text{Poly}(\log q))$.

Can reduce the factor base à la Harley:

Heuristic Result 2. For any elliptic curve $E(\mathbb{F}_{q^n})$, by making $O(q^{1 - \frac{1}{n+1}})$ queries to a Static DHP$_d$ oracle during an initial learning phase, for fixed $n > 1$ and $q \to \infty$, an adversary can solve any further instance of the Static DHP$_d$ in time $\tilde{O}(q^{1 - \frac{1}{n+1}})$
Heuristic Result 1. *For any elliptic curve* $E(\mathbb{F}_{q^n})$, *by making* $O(q)$ *queries to a Static DHP\(_d\) oracle during an initial learning phase, for fixed* $n > 1$ *and* $q \to \infty$, *an adversary can solve any further instance of the Static DHP\(_d\) in time* $O(\text{Poly}(\log q))$.

- Can reduce the factor base à la Harley:

Heuristic Result 2. *For any elliptic curve* $E(\mathbb{F}_{q^n})$, *by making* $O(q^{1-\frac{1}{n+1}})$ *queries to a Static DHP\(_d\) oracle during an initial learning phase, for fixed* $n > 1$ *and* $q \to \infty$, *an adversary can solve any further instance of the Static DHP\(_d\) in time* $\tilde{O}(q^{1-\frac{1}{n+1}})$.

- Can also obtain subexponential algorithm à la Diem.
The Galbraith-Lin-Scott Curves

At EUROCRYPT 2009 the use of curves defined over extension fields with degree a power of 2 were proposed.

- Exploits the existence of efficiently computable homomorphism to enable use of the GLV fast point multiplication method
- GLV: if $\psi$ is an efficiently computable endomorphism of $E$ then one can compute $[n]P = [n_0]P + [n_1]\psi(P)$ with $|n_i| \approx \sqrt{\#E}$
- Over $\mathbb{F}_{p^2}$ method takes about 0.75 the time of the previous best methods
- Performance over $\mathbb{F}_{p^4}$ currently uninvestigated, but subject to Gaudry’s ECDLP attack
The Oakley key determination protocol curves
‘Well-Known Group’ 3

Group 3 is defined over the field \( \mathbb{F}_{2^{155}} = \mathbb{F}_2[\omega]/(\omega^{155} + \omega^{62} + 1) \), by the equation

\[
Y^2 + XY = X^3 + \beta,
\]

where

\[
\beta = \omega^{18} + \omega^{17} + \omega^{16} + \omega^{13} + \omega^{12} + \omega^{9} + \omega^{8} + \omega^{7} + \omega^{3} + \omega^{2} + \omega + 1.
\]

\[
\#E(\mathbb{F}_{2^{155}}) = 12 \cdot r, \text{ with } r = 3805993847215893016155463826195386266397436443
\]

Subject to several unsuccessful DLP attacks via Weil descent: Jacobson/Menezes/Stein [01], Gaudry/Hess/Smart [00], Galbraith/Hess/Smart [02], Hess [03].
The Oakley key determination protocol curves
‘Well-Known Group’ 4

Group 4 is defined over the field $\mathbb{F}_{2^{185}} = \mathbb{F}_2[\omega]/(\omega^{185} + \omega^{69} + 1)$, by the equation

$$Y^2 + XY = X^3 + \beta,$$

where

$$\beta = \omega^{12} + \omega^{11} + \omega^{10} + \omega^9 + \omega^7 + \omega^6 + \omega^5 + \omega^3 + 1.$$  

- $\#E(\mathbb{F}_{2^{185}}) = 4 \cdot r$, with $r = 12259964326927110866866776214413170562013096\backslash$
  
  250261263279

- DLP studied by Maurer/Menezes/Teske [01] and Menezes/Teske/Weng [04], the latter concluding that the fields $\mathbb{F}_{2^l}$ for $l > 37$ are ‘weak’ while the security of ECs over $\mathbb{F}_{2^{185}}$ is questionable.
Large prime characteristic

For each of \( n = 2, 3, 4 \) and 5 we used curves of the form

\[
E(\mathbb{F}_{p^n}) : y^2 = x^3 + ax + b,
\]

for \( a \) and \( b \) randomly chosen elements of \( \mathbb{F}_{p^n} \), such that \( \#E(\mathbb{F}_{p^n}) \) was a prime of bitlength 256.

- Implemented in MAGMA (V2.16-5) run on a 3.16 GHz Intel Xeon with 32G RAM

Data for testing and decomposing points for elliptic curves over extension fields (times in s):

<table>
<thead>
<tr>
<th>( n )</th>
<th>( \log p )</th>
<th>( # f_{n+1} )</th>
<th>( # \text{ symf}_{n+1} )</th>
<th>( T(\text{GB}) )</th>
<th>( T(\text{roots}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>128</td>
<td>13</td>
<td>5</td>
<td>0.001</td>
<td>0.009</td>
</tr>
<tr>
<td>3</td>
<td>85.3</td>
<td>439</td>
<td>43</td>
<td>0.029</td>
<td>0.027</td>
</tr>
<tr>
<td>4</td>
<td>64</td>
<td>54777</td>
<td>1100</td>
<td>5363</td>
<td>3.68</td>
</tr>
</tbody>
</table>
Given data, compute $\alpha$ such that:

$$p^{n(1-\alpha)} \cdot n! \cdot (T(\text{GB}) + T(\text{roots})) = p^{\alpha} \cdot T(\text{scalar})$$
Given data, compute $\alpha$ such that:

$$p^{n(1-\alpha)} \cdot n! \cdot (T(GB) + T(\text{roots})) = p^\alpha \cdot T(\text{scalar})$$

Attack time estimates for our implementation (times in s):

<table>
<thead>
<tr>
<th>$n$</th>
<th>$\alpha$</th>
<th>Attack time</th>
<th>Pollard rho</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.6701 (2/3)</td>
<td>$2^{79.8}$</td>
<td>$2^{111.3}$</td>
</tr>
<tr>
<td>3</td>
<td>0.7645 (3/4)</td>
<td>$2^{59.7}$</td>
<td>$2^{111.4}$</td>
</tr>
<tr>
<td>4</td>
<td>0.8730 (4/5)</td>
<td>$2^{50.5}$</td>
<td>$2^{111.4}$</td>
</tr>
</tbody>
</table>
Characteristic two

For each of \( n = 2, 3, 4 \) and 5 we used curves of the form

\[
E(\mathbb{F}_{2^n}) : y^2 + xy = x^3 + b,
\]

for \( b \) a randomly chosen element of \( \mathbb{F}_{2^n} \), such that \( \#E(\mathbb{F}_{2^n}) \) was a four times a prime of bitlength 256.
For each of $n = 2, 3, 4$ and 5 we used curves of the form

$$E(\mathbb{F}_{2^n}) : y^2 + xy = x^3 + b,$$

(1)

for $b$ a randomly chosen element of $\mathbb{F}_{2^n}$, such that $\#E(\mathbb{F}_{2^n})$ was a four times a prime of bitlength 256.

Data for testing and decomposing points for elliptic curves over binary extension fields and attack time estimates (times in s):

<table>
<thead>
<tr>
<th>$n$</th>
<th>$#f_{n+1}$</th>
<th>$# \text{sym} f_{n+1}$</th>
<th>Time GB</th>
<th>$\alpha$</th>
<th>Attack time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>3</td>
<td>0.000</td>
<td>0.6672</td>
<td>$2^{80.9}$</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
<td>6</td>
<td>0.005</td>
<td>0.7572</td>
<td>$2^{60.0}$</td>
</tr>
<tr>
<td>4</td>
<td>729</td>
<td>39</td>
<td>247</td>
<td>0.8575</td>
<td>$2^{50.6}$</td>
</tr>
<tr>
<td>5</td>
<td>148300</td>
<td>638</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
The Joux-Vitse variation

- Joux-Vitse[10] gave a variant of Gaudry’s algorithm which improves ECDLP complexity for $n \geq c^3\sqrt{\log p}$
- Noted the same algorithm as *Heuristic Result 1* for the oracle-assisted Static DHP
- Observed that the obstacle to finding relations for $n \geq 5$ is the degree of the summation poly $(2^{n-1})$ and resulting system $(2^n(n-1))$
- To circumvent this, they add not $n$ points of $\mathcal{F}$ but $n-1$, i.e.,
  \[ R = P_1 + \cdots + P_{n-1} \]
- This reduces the degree to $2^{n-2}$, and results in an overdetermined system since one has $n$ equations
The Joux-Vitse variation

- Developed a new version of Faugère’s $F4$ algorithm to exploit solving a system of the same shape many times
- Prob. of a random element being representable is reduced to $1/(p \cdot (n - 1)!)$
- For prime base fields with $\log_2 p \approx 32$ and $n = 5$ they can test a decomposition in about 8.5s on a 2.6 GHz Intel Core 2 Duo (Magma takes 1046s)
- Implemented their method for binary fields using the F4 algorithm in Magma: $\approx 1000$ times faster than large $p$
The Joux-Vitse variation

- Developed a new version of Faugère’s $F4$ algorithm to exploit solving a system of the same shape many times.
- Prob. of a random element being representable is reduced to $1/(p \cdot (n-1)!)$
- For prime base fields with $\log_2 p \approx 32$ and $n = 5$ they can test a decomposition in about 8.5s on a 2.6 GHz Intel Core 2 Duo (Magma takes 1046s).
- Implemented their method for binary fields using the F4 algorithm in Magma: $\approx 1000$ times faster than large $p$.
- Vanessa’s implementation: Decomposition test time is $22.95 ms$ on a 2.93 GHz Intel Xeon processor.
The Joux-Vitse variation

- Developed a new version of Faugère’s $F4$ algorithm to exploit solving a system of the same shape many times
- Prob. of a random element being representable is reduced to $1/(p \cdot (n - 1)!)$
- For prime base fields with $\log_2 p \approx 32$ and $n = 5$ they can test a decomposition in about $8.5s$ on a 2.6 GHz Intel Core 2 Duo (Magma takes 1046s)
- Implemented their method for binary fields using the F4 algorithm in Magma: $\approx 1000$ times faster than large $p$
- Vanessa’s implementation: Decomposition test time is $22.95ms$ on a 2.93 GHz Intel Xeon processor
- Total time (excluding $\approx 2^{30}$ oracle queries) is $\approx 40.4$ years
Let $E : y^2 + xy = x^3 + \beta$ be an elliptic curve over $\mathbb{F}_{q^n}$

Fix a basis \{${t^{n-1}, \ldots, t, 1}$\} for $\mathbb{F}_{q^n}/\mathbb{F}_q$

Writing

$$\begin{align*}
\beta &= b_0 + b_1 t + \cdots + b_{n-1} t^{n-1}, \\
x &= x_0 + x_1 t + \cdots + x_{n-1} t^{n-1}, \\
y &= y_0 + y_1 t + \cdots + y_{n-1} t^{n-1},
\end{align*}$$

upon substituting into equation for $E$ and equating coefficients of $t$, one obtains a variety $W$ of dimension $n$ over $\mathbb{F}_q$.

$W$ is called the Weil restriction of $E$
Weil descent and the GHS attack

- If $E/F_{q^k}$ contains a cryptographically interesting group of prime order $r$ then $W$ contains an irreducible subvariety $V$ with group order divisible by $r$
- GHS attack finds a hyperelliptic curve $H$ in $W$ whose Jacobian contains a subvariety isogenous to $V$
- One can then map the DLP

$$\phi : E(F_{q^k}) \rightarrow \text{Jac}_H(F_q),$$

and apply index calculus to $\text{Jac}_H(F_q)$
- In GHS attack elements of $E(F_{2^{ln}})[r]$ map to Jacobian of hyperelliptic curve $H(F_{2^l})$ of genus at most $2^{n-1}$
One can define $\mathcal{F}$ as before to be the set of degree one divisors in $\text{Jac}_H(\mathbb{F}_q)$.
One can define $\mathcal{F}$ as before to be the set of degree one divisors in $\text{Jac}_H(\mathbb{F}_q)$.

**Problem 1:** Can not call Static DHP oracle on elements of $\text{Jac}_H(\mathbb{F}_q)$!
Oracle-assisted Static DHP via GHS attack?

- One can define $\mathcal{F}$ as before to be the set of degree one divisors in $\text{Jac}_H(\mathbb{F}_q)$
- **Problem 1**: Can not call Static DHP oracle on elements of $\text{Jac}_H(\mathbb{F}_q)$!
- **Solution 1**: $\phi$ is easily invertible: just a conorm and norm computation
One can define $\mathcal{F}$ as before to be the set of degree one divisors in $\text{Jac}_H(\mathbb{F}_q)$.

**Problem 1:** Can not call Static DHP oracle on elements of $\text{Jac}_H(\mathbb{F}_q)$!

**Solution 1:** $\phi$ is easily invertible: just a conorm and norm computation.

**Problem 2:** Elements of $\mathcal{F}$ are not in $\text{im}(\phi)$!
One can define $\mathcal{F}$ as before to be the set of degree one divisors in $\text{Jac}_H(\mathbb{F}_q)$.

**Problem 1:** Can not call Static DHP oracle on elements of $\text{Jac}_H(\mathbb{F}_q)$!

**Solution 1:** $\phi$ is easily invertible: just a conorm and norm computation.

**Problem 2:** Elements of $\mathcal{F}$ are not in $\text{im}(\phi)$!

**Solution 2:** No problem if $(\#\text{Jac}_H(\mathbb{F}_{2^l})/r, r) = 1$.
Oracle-assisted Static DHP via GHS attack

- Let $\mathcal{F}$ be the set of degree one divisors in $\text{Jac}_H(\mathbb{F}_{2^l})$
- Let $N = \#\text{Jac}_H(\mathbb{F}_{2^l})$ and $h = N/r$
- Project each $D \in \mathcal{F}$ into $\text{im}(\phi)$ by multiplying by $h$
- Compute $\phi^{-1}(hD)$ for each $D \in \mathcal{F}$
- Call the Static DHP$_d$ oracle $\delta$ on each $\phi^{-1}(hD)$ in $E(\mathbb{F}_{2^ln})$
- For a target $X \in E(\mathbb{F}_{2^ln})$ take random multiples until $\phi(aX) = \sum D_i$ with each $D_i \in \mathcal{F}$
- Then assuming $(h, r) = 1$ one computes

$$\delta(X) = (a^{-1} \mod r)(h^{-1} \mod r) \sum \delta(\phi^{-1}(hD_i))$$
We have $\phi : E(\mathbb{F}_{2^{155}})[r] \longrightarrow \text{Jac}_H(\mathbb{F}_{2^{31}})$ for hyperelliptic

$$H : Y^2 + h(X) \cdot Y = f(X),$$

with $\mathbb{F}_{2^{31}} = \mathbb{F}_2[\omega]/(\omega^{31} + \omega^3 + 1)$ and

$$h(X) = 289804524X^{16} + 607247628X^8 + 1798965180X^4$$
$$+ 1103766465X^2 + 742287012X,$$

$$f(X) = 505223067X^{33} + 1000507042X^{17} + 1992775259X^{16}$$
$$+ 1146351457X^9 + 1078048302X^8 + 284388091X^5$$
$$+ 518998412X^4 + 1875045691X^3 + 2001664187X^2$$
$$+ 1973705837X,$$

and genus$(H) = 16 = 2^{155/31} - 1$
Using Florian’s LMS J. Comput. Math paper (or a magma computation), one finds $N = \#\text{Jac}_H(\mathbb{F}_{2^{31}})$ which has bitlength 497.

Furthermore $(N/r, r) = 1$ and so attack can proceed.

Using Victor Shoup’s Number Theory Library on a 3.16GHz Intel Xeon, testing 1-smoothness of a random multiple of $\phi(P)$ takes $\approx 0.690\, ms$.

Other basic cost is a point addition in the Jacobian; Jacobson estimates this to be $< 1/2.3$ the cost of smoothness test using NUCOMP.

Hence expected time to find a relation using a single processor is $\approx 650$ years.
We have $\phi : E(\mathbb{F}_{2^{185}})[r] \longrightarrow \text{Jac}_H(\mathbb{F}_{2^{37}})$ for hyperelliptic

$$H : Y^2 + h(X) \cdot Y = f(X),$$

with $\mathbb{F}_{2^{37}} = \mathbb{F}_2[\omega]/(\omega^{37} + \omega^9 + \omega^2 + \omega + 1)$ and

$$h(X) = 73994877348X^{16} + 113350789030X^8 + 86827085475X^4$$
$$+ 21964938327X^2 + 125543309305X,$$

$$f(X) = 49045248530X^{33} + 40737336296X^{17} + 45140903646X^{16}$$
$$+ 120039047741X^9 + 105120752497X^8 + 72787224919X^5$$
$$+ 25040887869X^4 + 72047225547X^3 + 94586877616X^2$$
$$+ 68639477599X,$$

and genus$(H) = 16 = 2^{185/37}-1$
\( N = \#\text{Jac}_H(\mathbb{F}_{2^{37}}) \) has bitlength 592
Again \((N/r, r) = 1\) and so attack can proceed
Using NTL on the same processor testing 1-smoothness of a random multiple of \(\phi(P)\) takes \(\approx 0.854\,ms\)
Cost of point addition in the Jacobian \(\approx 1/2.3\) the cost of smoothness test using NUCOMP
Hence expected time to find a relation using a single processor is \(\approx 810\) years
Components of learning phase:

- Construct factor base $\mathcal{F}$ of degree 1 divisors: $\approx 2^{l-1}$ such divisors ignoring negatives
- Map each $D \in \mathcal{F}$ to an element of $\text{im}(\phi)$ via multiplication by $h = \#\text{Jac}_H(\mathbb{F}_{2^l}) / r \approx 2^{l(2^{n-1} - n)}$
- Compute $\phi^{-1}(hD)$ for each $D \in \mathcal{F}$
- Call the Static DHP$_d$ oracle $\delta$ on each $\phi^{-1}(hD)$ in $E(\mathbb{F}_{2^l})$

Expected cost of relation find:

- Cost of each smoothness test $\approx (128l - 288) \mathbb{F}_{2^l}$ multiplications
- Hence total cost is $\approx (2^{n-1})! \cdot (128l - 288) \mathbb{F}_{2^l}$ multiplications
Consider asymptotics for fixed $n$ and $l \to \infty$. Write $g = 2^{n-1}$.

- For $2^l > g!$ the dominant cost is the oracle calls
- Hence should reduce $\mathcal{F}$ to balance the two stages
- Let $q = 2^l$ and let $|\mathcal{F}_s| = q^\alpha$ with $0 < \alpha \leq 1$
- Probability that a random point decomposes over $\mathcal{F}_s$ is $q^{g(\alpha-1)}/g!$

Solving $g! \cdot q^{g(1-\alpha)} = q^\alpha$ gives $\alpha = \frac{g+\log q \cdot g!}{g+1}$ and so complexity of algorithm is

$$\tilde{O}(q^{1-\frac{1}{g+1}}).$$

This is the square-root of the balanced DLP algorithm complexity for fixed genus (Gaudry/Harley)
Comparison with the Gaudry/Semaev-based method

- For fixed \( n \) and increasing \( q \) first algorithm is asymptotically faster: \( \tilde{O}(q^{1 - \frac{1}{n+1}}) \) vs \( \tilde{O}(q^{1 - \frac{1}{g+1}}) \)

- In practice, smoothness test is much easier than a decomposition — have a trade-off between decomposition probability and ease of decomposition test — so may even be better for \( n = 2, 3, 4, \) as well as 5

- Method is really tailored for when Gaudry/Semaev decompositions are impractical

- Limitation: details are only clear in characteristic 2
Some problems occurring in security proofs are easier than DLP, especially when index calculus applies.
Some problems occurring in security proofs are easier than DLP, especially when index calculus applies.

Elliptic curves defined over extension fields may be unsuitable in some scenarios.
Some problems occurring in security proofs are easier than DLP, especially when index calculus applies.

Elliptic curves defined over extension fields may be unsuitable in some scenarios.

Interesting use of auxiliary groups when an efficiently computable two-way map present — no need for a native factorisation/decomposition method at all.