DL systems over finite fields II

Index calculus attacks

Tanja Lange

Eindhoven University of Technology

2MMC10 – Cryptology

Rough idea

- Translate solving $\log_g(h)$ into solving many smaller DLs

Rough idea

- Translate solving $\log_g(h)$ into solving many smaller DLs which we can solve using linear relations.

Rough idea

- Translate solving $\log_g(h)$ into solving many smaller DLs which we can solve using linear relations.
- Assume we know $a_i = \log_g(p_i)$ for lots of small primes p_i.
  Assume for simplicity that $\mathbf{F}_p^* = \langle g \rangle$, else restrict to primes in $\langle g \rangle$.

Rough idea

- Translate solving $\log_g(h)$ into solving many smaller DLs which we can solve using linear relations.
- Assume we know $a_i = \log_g(p_i)$ for lots of small primes p_i.
  Assume for simplicity that $\mathbf{F}_p^* = \langle g \rangle$, else restrict to primes in $\langle g \rangle$.
- If $h = \prod p_i^{e_i}$ and DLs are known for all p_i then

$$\log_g(h) = \log_g\left(\prod p_i^{e_i}\right) = \sum e_i \log_g(p_i) = \sum e_i a_i.$$

Rough idea

- Translate solving $\log_g(h)$ into solving many smaller DLs which we can solve using linear relations.
- Assume we know $a_i = \log_g(p_i)$ for lots of small primes p_i.
  Assume for simplicity that $\mathbf{F}_p^* = \langle g \rangle$, else restrict to primes in $\langle g \rangle$.
- If $h = \prod p_i^{e_i}$ and DLs are known for all p_i then

$$\log_g(h) = \log_g\left(\prod p_i^{e_i}\right) = \sum e_i \log_g(p_i) = \sum e_i a_i.$$

  Else pick random k and check whether $g^k h$, taken as integer in $[1, p-1]$ factors as $\prod p_i^{e_i}$.

Rough idea

- Translate solving $\log_g(h)$ into solving many smaller DLs which we can solve using linear relations.
- Assume we know $a_i = \log_g(p_i)$ for lots of small primes p_i.
  Assume for simplicity that $\mathbf{F}_p^* = \langle g \rangle$, else restrict to primes in $\langle g \rangle$.
- If $h = \prod p_i^{e_i}$ and DLs are known for all p_i then

$$\log_g(h) = \log_g\left(\prod p_i^{e_i}\right) = \sum e_i \log_g(p_i) = \sum e_i a_i.$$

  Else pick random k and check whether $g^k h$, taken as integer in $[1, p-1]$ factors as $\prod p_i^{e_i}$.
  If so, $\log_g(h) + k = \sum e_i a_i$, else repeat with different choice of k.
- Stage 1 collects relations to get the a_i.
  This is independent of the target h.
- Stage 2 Some random choices of k allow to recover $\log_g(h)$.

Rough idea

- Translate solving $\log_g(h)$ into solving many smaller DLs which we can solve using linear relations.
- Assume we know $a_i = \log_g(p_i)$ for lots of small primes p_i.
  Assume for simplicity that $\mathbf{F}_p^* = \langle g \rangle$, else restrict to primes in $\langle g \rangle$.
- If $h = \prod p_i^{e_i}$ and DLs are known for all p_i then

  $$\log_g(h) = \log_g\left(\prod p_i^{e_i}\right) = \sum e_i \log_g(p_i) = \sum e_i a_i.$$

  Else pick random k and check whether $g^k h$, taken as integer in $[1, p-1]$ factors as $\prod p_i^{e_i}$.
  If so, $\log_g(h) + k = \sum e_i a_i$, else repeat with different choice of k.
- Stage 1 collects relations to get the a_i.
  This is independent of the target h.
- Stage 2 Some random choices of k allow to recover $\log_g(h)$.
- Optimized attacks work with these 2 stages but differ in details from schoolbook method.
- See `https://weakdh.org/` for an optimized attack aiming to break many targets (think of nation state attacker).

Schoolbook version stage 1

Define factor base $\mathcal{F} = \{p_i | p_i \text{ prime }, p_i < B\}$ for some bound B.
Let $f = |\mathcal{F}|$.

Repeat the following until $f + 4$ relations are collected.

1. Pick random integer j.
2. Compute g^j in $\mathbf{F}_p$. Consider result as integer $b \in [0, p-1]$.
3. Check whether b factors over the factor base, i.e. whether

$$b = \prod_{i=1}^{f} p_i^{e_i} \text{ for } p_i \in \mathcal{F}, e_i \in \mathbf{N}$$

   If so, $j = \sum e_i \log_g(p_i)$. Store relation $(e_1, e_2, \ldots, e_f, j)$

Put the relations in a matrix. Note, inhomogenous system.
Use linear algebra to compute a solution to the system modulo $\mathrm{ord}(g)$.
Output result $(a_1, a_2, \ldots, a_f)$.

If system underdetermined, collect more relations.

Schoolbook version stage 2

This part uses the target h.

Repeat the following until successful

1. Pick random integer k.
2. Compute $g^k h$ in $\mathbf{F}_p$. Consider result as integer $b \in [0, p-1]$.
3. Check whether b factors over the factor base, i.e. whether

$$b = \prod_{i=1}^{f} p_i^{e_i} \text{ for } p_i \in \mathcal{F}, e_i \in \mathbf{N}$$

   If so, output $-k + \sum e_i a_i$ modulo $\mathrm{ord}(g)$.

Schoolbook version stage 2

This part uses the target h.

Repeat the following until successful

1. Pick random integer k.
2. Compute $g^k h$ in $\mathbf{F}_p$. Consider result as integer $b \in [0, p-1]$.
3. Check whether b factors over the factor base, i.e. whether

$$b = \prod_{i=1}^{f} p_i^{e_i} \text{ for } p_i \in \mathcal{F}, e_i \in \mathbf{N}$$

   If so, output $-k + \sum e_i a_i$ modulo $\operatorname{ord}(g)$.

- Many optimizations to improve smoothness chance for b in stage 1.
- Make structured choices of j to enable sieving.

Schoolbook version stage 2

This part uses the target h.

Repeat the following until successful

1. Pick random integer k.
2. Compute $g^k h$ in $\mathbf{F}_p$. Consider result as integer $b \in [0, p-1]$.
3. Check whether b factors over the factor base, i.e. whether

$$b = \prod_{i=1}^{f} p_i^{e_i} \text{ for } p_i \in \mathcal{F}, e_i \in \mathbf{N}$$

If so, output $-k + \sum e_i a_i$ modulo $\mathrm{ord}(g)$.

- Many optimizations to improve smoothness chance for b in stage 1.
- Make structured choices of j to enable sieving.
- Many optimizations of number-field sieve for factoring carry over. Best index calculus attack for $\mathbf{F}_p$ also called number-field sieve and uses number fields and sieving.
- Asymptotic cost $L^{c+o(1)}$ for constant c where $L = \exp((\ln n)^{1/3}(\ln \ln n)^{2/3})$

Index-calculus attack for $\mathbf{F}_q = \mathbf{F}_{p^n}$

Easiest example: $p = 2, n$ large.

$\mathbf{F}_{2^n} \cong \mathbf{F}_2[x]/f(x)$, with $f(x) \in \mathbf{F}_2[x]$ monic, irreducible and $\deg(f) = n$.
Thus $\mathbf{F}_{2^n} \cong \{\sum_{i=0}^{n-1} c_i x^i | c_i \in \mathbf{F}_2\}$.

- Put $\mathcal{F} = \{p_i(x) | p_i(x) \in \mathbf{F}_2[x], \deg(p_i) \leq b, p_i$ is irreducible$\}$.
- Compute g^j in $\mathbf{F}_{2^n}$, consider result in $\mathbf{F}_2[x]$ and factor there.
- Factorization in $\mathbf{F}_2[x]$ even faster than in $\mathbf{Z}$,
  all sieving ideas work the same.

Index-calculus attack for $\mathbf{F}_q = \mathbf{F}_{p^n}$

Easiest example: $p = 2, n$ large.

$\mathbf{F}_{2^n} \cong \mathbf{F}_2[x]/f(x)$, with $f(x) \in \mathbf{F}_2[x]$ monic, irreducible and $\deg(f) = n$.
Thus $\mathbf{F}_{2^n} \cong \{\sum_{i=0}^{n-1} c_i x^i | c_i \in \mathbf{F}_2\}$.

- Put $\mathcal{F} = \{p_i(x) | p_i(x) \in \mathbf{F}_2[x], \deg(p_i) \leq b, p_i$ is irreducible$\}$.
- Compute g^j in $\mathbf{F}_{2^n}$, consider result in $\mathbf{F}_2[x]$ and factor there.
- Factorization in $\mathbf{F}_2[x]$ even faster than in $\mathbf{Z}$,
  all sieving ideas work the same.
- Coppersmith showed asymptotic cost $L^{c'+o(1)}$ where
  $L = \exp((\ln n)^{1/3}(\ln \ln n)^{2/3})$ and $c' < c$.
  Attack called function-field sieve.

Index-calculus attack for $\mathbf{F}_q = \mathbf{F}_{p^n}$

Easiest example: $p = 2, n$ large.

$\mathbf{F}_{2^n} \cong \mathbf{F}_2[x]/f(x)$, with $f(x) \in \mathbf{F}_2[x]$ monic, irreducible and $\deg(f) = n$.
Thus $\mathbf{F}_{2^n} \cong \{\sum_{i=0}^{n-1} c_i x^i | c_i \in \mathbf{F}_2\}$.

- Put $\mathcal{F} = \{p_i(x) | p_i(x) \in \mathbf{F}_2[x], \deg(p_i) \le b, p_i$ is irreducible$\}$.
- Compute g^j in $\mathbf{F}_{2^n}$, consider result in $\mathbf{F}_2[x]$ and factor there.
- Factorization in $\mathbf{F}_2[x]$ even faster than in $\mathbf{Z}$,
  all sieving ideas work the same.
- Coppersmith showed asymptotic cost $L^{c'+o(1)}$ where
  $L = \exp((\ln n)^{1/3}(\ln \ln n)^{2/3})$ and $c' < c$.
  Attack called function-field sieve.

For $\mathbf{F}_{p^n}$ with small p use function-field sieve,
for large p and small n use number-field sieve.

Index-calculus attack for $\mathbf{F}_q = \mathbf{F}_{p^n}$

Easiest example: $p = 2, n$ large.

$\mathbf{F}_{2^n} \cong \mathbf{F}_2[x]/f(x)$, with $f(x) \in \mathbf{F}_2[x]$ monic, irreducible and $\deg(f) = n$.
Thus $\mathbf{F}_{2^n} \cong \{\sum_{i=0}^{n-1} c_i x^i | c_i \in \mathbf{F}_2\}$.

- Put $\mathcal{F} = \{p_i(x) | p_i(x) \in \mathbf{F}_2[x], \deg(p_i) \leq b, p_i \text{ is irreducible}\}$.
- Compute g^j in $\mathbf{F}_{2^n}$, consider result in $\mathbf{F}_2[x]$ and factor there.
- Factorization in $\mathbf{F}_2[x]$ even faster than in $\mathbf{Z}$, all sieving ideas work the same.
- Coppersmith showed asymptotic cost $L^{c'+o(1)}$ where $L = \exp((\ln n)^{1/3}(\ln \ln n)^{2/3})$ and $c' < c$.
  Attack called function-field sieve.

For $\mathbf{F}_{p^n}$ with small p use function-field sieve,
for large p and small n use number-field sieve.

For small p security has much degraded in 2012 – 2014
with new attacks reaching quasi-polynomial time.
Many improvements, but not as dramatic, for large p and small $n > 1$.

Granger and Joux recently wrote a survey of DL attacks,
see page 13 onwards for finite fields.