E-th roots and static Diffie-Hellman using index calculus

Antoine Joux[1]

Joint work with Reynald Lercier[2], David Naccache[3], Emmanuel Thomé[4]

Elliptic Curve Cryptography 2008
Utrecht

[1] DGA and UVSQ
[2] DGA and IRMAR
[3] ENS
[4] INRIA Lorraine

Key questions

Security of plain RSA ⇕ Factoring

?

Diffie-Hellman ⇕ Discrete Log.

Quick reminder: RSA

- RSA: Rivest, Shamir, Adleman (1977)
- Public key: N a large integer, e encryption exponent
- Private key: $N = pq$, p and q prime, d decryption exponent

$$ed = \lambda(p-1)(q-1) + 1.$$

$$\text{Encryption} \quad : \quad x \longrightarrow x^e \pmod N$$

$$\text{Decryption} \quad : \quad y \longrightarrow \sqrt[e]{y} \pmod N$$
$$y \longrightarrow y^d \pmod N$$

Quick reminder: Diffie-Hellman

- Invented by Diffie and Hellman (1976)
- Public parameters: p a large prime, g a generator (subgroup)
- Key exchange:

Alice		Bob
Choose a	$\longrightarrow$	g^a
g^b	$\longleftarrow$	Choose b

$$g^{ab}$$

- When $a = s$ is fixed: Static Diffie-Hellman

Quick reminder: RSA and factoring ?

- Pros:
  - Finding d is as difficult as factoring N
    - Probabilistic (already in RSA from Miller 1975)
    - Deterministic (May 2004)
  - Breaking RSA may be as difficult as factoring (Brown 2006)
- Cons:
  - Specific weaknesses:
    - Multiplicative attacks
    - Blinding
  - Breaking RSA may be easier than factoring (Boneh, Venkatesan, 1998)

Specific weaknesses

- Multiplicative attacks:
  - From $\sqrt[e]{a}$ and $\sqrt[e]{b}$, deduce $\sqrt[e]{ab}$.
- Blinding:
  - Ask $\sqrt[e]{ar^e}$. Deduce $\sqrt[e]{a}$.

Quick reminder: Diffie-Hellman and DLOG ?

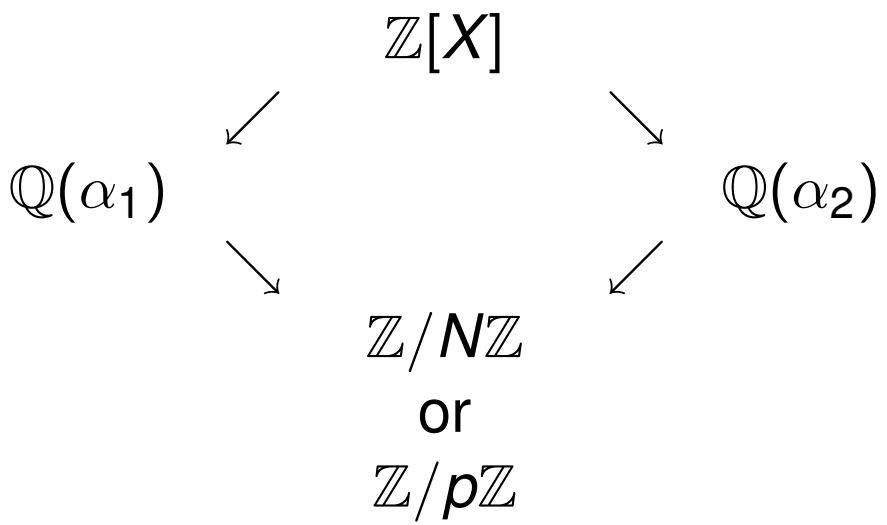
- Computational Diffie-Hellman and Discrete Log. (Maurer-Wolf 1996)
- Static Diffie-Hellman less clear (Brown-Gallant 2005)

Reformulating the key question

- RSA:
  - Given access to an e-th root oracle:
  - Can we learn to compute e-th roots ?
    - Efficienty (with a cost lower than factoring) ?

- Diffie-Hellman
  - Given access to a static Diffie-Hellman oracle:
  - Can we learn to raise to the secret power ?
    - Efficienty (with a cost lower than discrete log.) ?

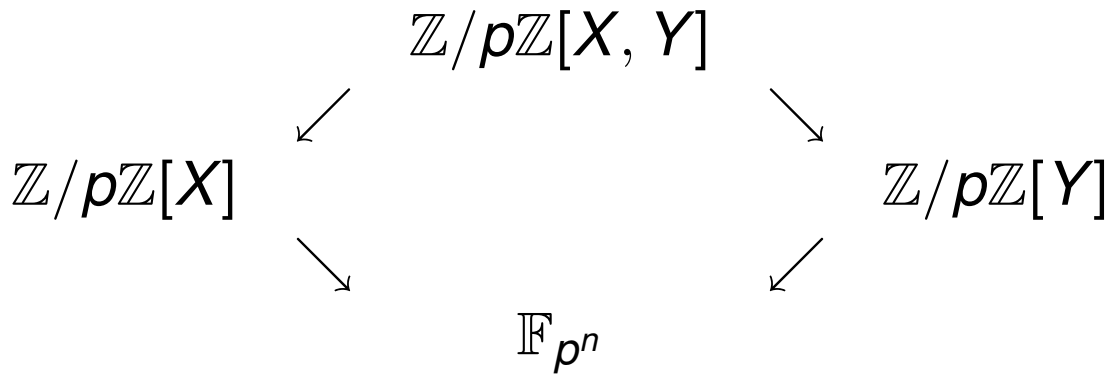
Reminder: Number Field Sieve



- Number fields defined from two polynomials: f_1 and f_2
- Relies on multiplicative relations over smoothness bases
- Applicable to factoring and discrete logarithms
- Complexity:

$$L_N(1/3, (64/9)^{1/3}) = e^{((64/9)^{1/3}+o(1))\log^{1/3} N \log\log^{2/3} N)}$$

Reminder: simplified Function Field Sieve



- Function fields defined from two polynomials: $x = f_1(y)$ and $y = f_2(x)$
- Applicable to discrete logarithms in small characteristic
- Complexity:

$$L_N(1/3, (32/9)^{1/3}) = e^{((32/9)^{1/3}+o(1))\log^{1/3} N \log\log^{2/3} N)}$$

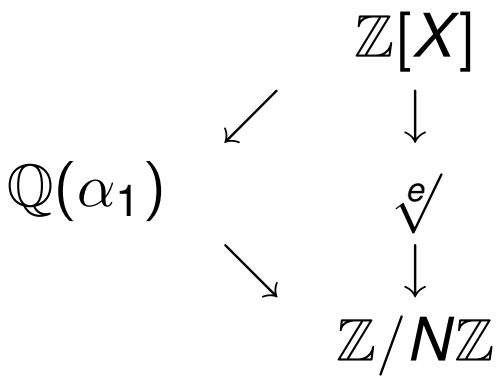
Reminder[1]: NFS and FFS

1. Find smooth objects and write multiplicative relations
2. Do linear algebra
3. Final stage
   - Finish factorization: Square root of ideal (Montgomery)
   - Compute individual discrete logarithms: Descent

[1] Another reminder: both are heuristic algorithms

A special case for RSA: Affine modular roots (AMR)

- Special oracle $\sqrt[e]{c+x}$ (c fixed, x small)
  - Multiplicative attack ?
    - Known attacks when $x \geq N^{1/3}$ is allowed
  - Arbitrary e-th roots ?

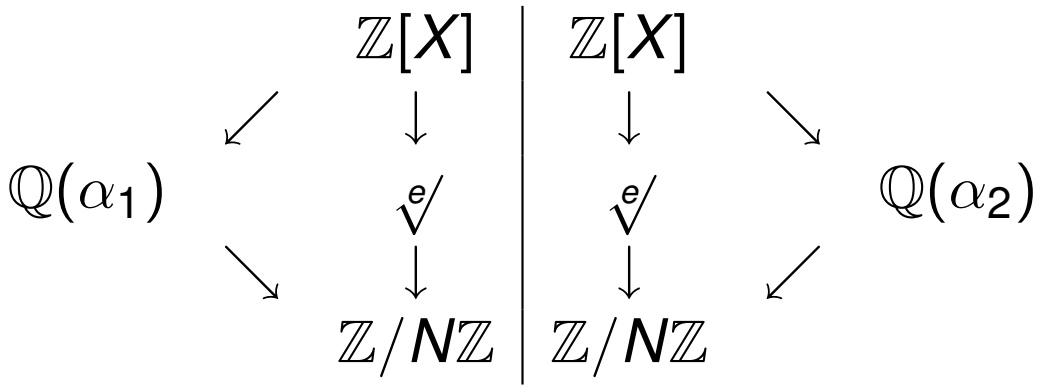


A special case for RSA: Affine modular roots

1. One sided smooth objects: multiplicative relations with $\sqrt[e]{\ }$
2. Do linear algebra: $\sqrt[e]{\ }$ of basis elements
3. Final stage
   - Get multiplicative relation
     - Existential forgery
   - Compute arbitrary e-th roots (with additional queries)
     - Universal forgery
     - One sided descent

Answering the key question

- General oracle $\sqrt[e]{x}$ or x^s
- Collect two sides
  - Sieving on one side. Twice.
  - Same complexity !



Easy case: FFS in small characteristic

- Two linear sides: No sieving and no linear algebra
- Descent (compute s-th power for $h(x)$)
  - Randomize until:
    $$h(x) = \frac{A(x)}{B(x)}$$
    is smooth enough.
  - For each factor $q(x)$, choose $l(x, y)$ to find:
    $$q(x)C(x) = D(y),$$
    with $C(x)$ and $D(y)$ smooth enough
- Finally backtrack from known s-th powers

Special q: How to

- We want $q(x)$ to divide $l(x, f_2(x))$
- That's $\deg(q)$ linear conditions
- Precompute $1, x, \ldots, x^{d_x}$ (modulo $q(x)$)
- Precompute $f_2(x), xf_2(x), \ldots, x^{d_x} f_2(x)$
- $\vdots$
- Precompute $f_2(x)^{d_y}, xf_2(x)^{d_y}, \ldots, x^{d_x} f_2(x)^{d_y}$
- Construct matrix and find kernel

FFS experiment in $\mathbb{F}_{2^{1025}}$

- Two polynomials:

$$\begin{aligned} y &= x^{171} + x^4 + x^3 + x^2 + 1 \\ x &= y^6 + y + 1 \end{aligned}$$

- 77 millions calls to oracle (deg. up to 29)
- Total runtime less than a week (single computer[2])
- For details, see IACR eprint 2008-217

[2] Intel Core-2 at 3.6 GHz

General case

1. Collect relations:
   - On side 1: sieving
   - On side 2: directly obtain e-th roots or s-th powers
2. Do linear algebra: $\sqrt[e]{\ }$ or s-th powers of basis elements (side 1)
   - Possibly delayed
3. Optionally enlarge smoothness bases
4. Final stage
   - Descent as in discrete logs
   - Recover e-th root or s power

General case: linear algebra

- Type of linear algebra:
  - Modulo e (or $p-1$) with Schirokauer's maps
    - Alternatively: Exact
  - Before the final stage: "Multiplicative"
  - Or postponed to backtrack of final phase

General case: Descent

- Descent for H:
  - Randomize until:
    $$H = \frac{A}{B}$$
    is smooth enough in $\mathbb{Z}$.
  - For each factor q, choose $ax + b$ to find:
    $$q.C = D(\alpha),$$
    with C and norm of $D(\alpha)$ smooth enough
- Backtrack (postponed linear algebra here)
- If modulo e, need variant of Montgomery's square root

RSA experiment on 512 bits

With public exponent $e = 65537$.

- 400 millions calls to oracle
- Initial sieving: 2 CPU hours[1]
- Bases extension: 44 CPU hours
- Descent time: around one hour
- Linear algebra[2]: 6 hours on 4 proc.
- Montgomery e-th root: five minutes
- For details, see IACR eprint 2007-424

Reminder: Factoring this number took 8000 mips.years

[1] AMD Opteron 2.4GHz.
[2] Intel Core 2 at 2.667GHZ

Dlog experiment on 516 bits

- Using $p = \lfloor 10^{155}\pi + 88896 \rfloor$
- 140 millions calls to oracle
- Initial sieving: 4 minutes on 128 proc.[1] (FB 2^{19})
- Base extension: 24 more minutes (FB 2^{32})
- Linear algebra[2]: 8 hours on 4 proc.
- Descent time: around two hours
- For details, see IACR eprint 2008-217

[1] Intel Core 2 at 1.6 GHz
[2] Intel Core 2 at 2.4 GHZ

Asymptotic complexity

All complexities are $L(1/3, \sqrt[3]{\cdot})$:

variant	calls	lin. alg.	descent	Dlog
FFS	$4/9$	-	$4/9$	$32/9$
NFS-HD	$48/91$	$384/91$	$384/91$	$128/9$
NFS[3]	$4/9$	$32/9$	3	$64/9$

Reminder, range of algorithms:

Algorithm	From p	To p
FFS	2	$L_{p^n}(1/3, \cdot)$
NFS-HD	$L_{p^n}(1/3, \cdot)$	$L_{p^n}(2/3, \cdot)$
NFS	$L_{p^n}(2/3,)$	$p = p^n$

[3] Requires Montgomery algorithm for RSA

Open problems

- Use oracle to factor/compute discrete log. faster ?
- What about static Diffie-Hellman on curves ?

Conclusion

Questions ?