Dlog is Practically as Hard (or Easy) as DH – Solving Dlogs via DH Oracles on EC Standards

Alexander May, Richard Schneider

Ruhr-University Bochum, Germany
CASA - casa.rub.de/en/

TCHES ’23

Dlogs and Diffie-Hellman (DH)

Setting: Consider elliptic curve $E(\mathbb{F}_p)$ with generator P of prime order q.

Dlog problem

Given: kP for $k \in \mathbb{Z}_q$
Find: k

Diffie-Hellman (DH) problem

Given: xP, yP for $x, y \in \mathbb{Z}_q$
Find: xyP

Notice: $xP + yP = (x + y)P$. So *addition* of x, y is easy, but *multiplication* requires DH.

Relation between Dlog and DH

Observe that Dlog $\Rightarrow$ DH. In general, DH may be easier than Dlog.
Problem: Crypto usually requires hardness of DH, e.g. Diffie-Hellman protocol, ElGamal.

Hardness of DH?

Our goal

Give strong security guarantee for Diffie-Hellman (DH) on elliptic curves.

Elliptic Curve Standards: NIST P-256, BLS12-381, BN(2,254), brainpoolP256t1, Curve25519, GOST R 34.10, SM2, secp256k1, NIST P-384, NIST P-521, etc.

- Define groups of bit-sizes 256, 384, 521.
- We assume that the best Dlog algorithms have square root complexity (Pollard Rho).
- Therefore we achieve security levels 128, 192, 260 bit against Dlog attacks.
- **Again:** How hard is DH then?

Main result

We provide reductions DH $\Rightarrow$ Dlog that are **tight in practice** (at least for 256 bit curves).

Maurer-Wolf reduction (1994, '96, '99)

Input:

1. Dlog instance $Q = kP$ on $E(\mathbb{F}_p)$ with $\mathrm{ord}(P) = q$
2. DH oracle $\mathcal{O}(xP, yP) = xyP$
3. *Auxiliary curve* $\hat{E}(\mathbb{F}_q)$ with generator $\hat{P}$ of smooth order

Output: $k \in \mathbb{Z}_q$

Maurer-Wolf: DH $\Rightarrow$ Dlog

1. **Lifting:** Construct point $\hat{Q} = (k, \cdot) \in \hat{E}(\mathbb{F}_q)$ such that Q has x-coordinate k.
   Important: We only implicitly represent $Q = [kP, \cdot]$.
2. **Auxiliary Dlog:** Let $\hat{Q} = u\hat{P}$. Compute u on auxiliary curve $\hat{E}(\mathbb{F}_q)$.
   Computation on implicit representation $\hat{Q} = [kP, \cdot]$ requires DH oracle $\mathcal{O}(\cdot, \cdot)$ calls.
3. **Dlog extraction:** Compute $\hat{Q} = u\hat{P} = (k, \cdot)$ explicitly, revealing $k \in \mathbb{Z}_q$.

Maurer-Wolf goes Real World Crypto: Our Contributions

1 Auxiliary Curve Construction

We explicitly compute auxiliary curves $\hat{E}(\mathbb{F}_q)$ with **smooth order** for most elliptic curve standards (focus on 256-bit).

2 Efficient Auxiliary Dlog Computation

For 256-bit curves our auxiliary curves allows us to precompute and store a **codebook of all dlogs** for $\hat{E}(\mathbb{F}_q)$.

3 Minimizing DH-Oracle Calls

We optimize for minimal number of oracle calls, thereby achieving a **practically tight reduction** DH $\Rightarrow$ Dlog.

Auxiliary Curve Construction

Auxiliary Curve Algorithm

Repeat

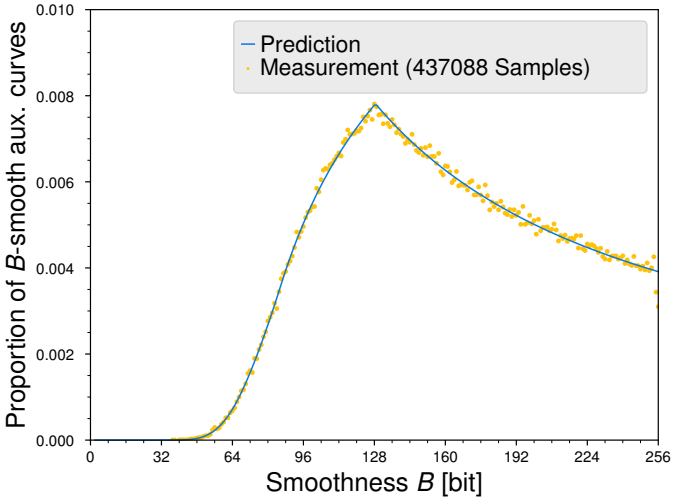
1. Pick a random auxiliary cyclic curve $\hat{E}(\mathbb{F}_q)$.
2. Compute $|\hat{E}(\mathbb{F}_q)|$ together with a generator $\hat{P}$ using Schoof's algorithm.
3. Factor $\mathrm{ord}(\hat{P}) = |\hat{E}(\mathbb{F}_q)| = \prod_i p_i^{e_i}$.

Until $\max_i\{p_i\}$ is sufficiently small.

Remarks:

- We call an iteration of our algorithm a sample.
- Every sample requires the factorization of a q-bit number.
- We have good predictions for the largest prime factor distribution (smoothness).
- This allows us to accurately predict the number of samples.

Running for NIST P-256



NIST P-256

NIST P-256's Auxiliary Curve $y^2 = x^3 + Ax + b \bmod q$

- A = 0x11c877b751dcab93a3dc546a7af6f26a4a7506a0f648d54b143b9cdbb100025a
- B = 0xee378847ae23546d5c23ab9585090d957271f40cb0cec939df7e2de74b6322f7
- $x(\hat{P})$ = 0xcd7ca16b05e3dd64c99da4a31cef71bdf7d48798d213a40ba4ec3a4d137bab30
- $y(\hat{P})$ = 0x445c8a8c21843080bf651958a5c26df5f9ad5bd73f4684d1ecb1026ec59c161f
- $\mathrm{ord}\hat{P} = (2 \cdot 3 \cdot 626663) \cdot 6487813 \cdot 17752487 \cdot 30034813 \cdot 620378903 \cdot 1316356273 \cdot 4747815593 \cdot 17399156003 \cdot 131964961211$

Achieved Smoothness

Curve	**q** [bit]	**B** [bit]	Sample
NIST P-256	**256**	**37**	437,088
BLS12-381	255	36	3,829,640
BN(2,254)	254	39	7,060
brainpoolP256t1	256	39	498,440

Curve	**q** [bit]	**B** [bit]	Sample
Curve25519	253	37	104,806
GOST R 34.10	256	37	113,350
secp256k1	256	37	991,302
SM2	256	39	840,273

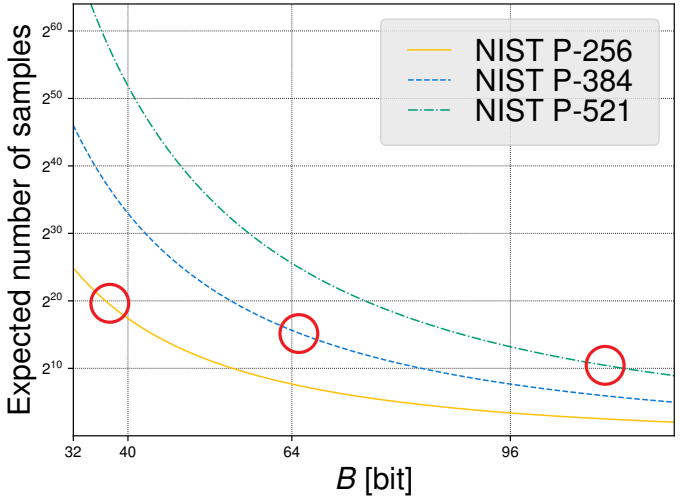
Takeaways: Auxiliary Curve Construction

- We run our algorithm until we achieve at most 39-bit smoothness.
- This requires on expectation roughly $500,000 \approx 2^{19}$ samples.
- Such smoothness allows for **efficient** Auxiliary Dlog computation.

Question: How does our construction scale for 384- and 521-bit elliptic curves?

Larger Groups

Curve	B [bit]	Sample
P-256	37	2^{19}
P-386	65	2^{14}
P-521	110	2^{11}



Maurer-Wolf goes Real World Crypto: Contribution 2

1 Auxiliary Curve Construction

We explicitly compute auxiliary curves $\hat{E}(\mathbb{F}_q)$ with **smooth order** for most elliptic curve standards (focus on 256-bit).

2 Efficient Auxiliary Dlog Computation

For 256-bit curves our auxiliary curves allows us to precompute and store a **codebook of all dlogs** for $\hat{E}(\mathbb{F}_q)$.

3 Minimizing DH-Oracle Calls

We optimize for minimal number of oracle calls, thereby achieving a **practically tight reduction** DH $\Rightarrow$ Dlog.

Storing und Computing Auxiliary Dlogs

Recall: $\hat{Q} = u\hat{P}$ with $u \in \mathbb{Z}_{\mathrm{ord}(\hat{P})}$ and $\mathrm{ord}(\hat{P}) = \prod_{i=1}^{n} p_i^{e_i}$. We have

$$\left(\frac{\mathrm{ord}(\hat{P}_i)}{p_i^{e_i}}\right)\hat{Q} = u\left(\frac{\mathrm{ord}(\hat{P}_i)}{p_i^{e_i}}\right)\hat{P} = (u \bmod p_i^{e_i})\left(\frac{\mathrm{ord}(\hat{P}_i)}{p_i^{e_i}}\right)\hat{P}.$$

Algorithm 1 Silver-Pohlig-Hellman algorithm

Input: $\hat{Q} = u\hat{P}$, $\hat{P}$

1: **for** $i = 1$ to n **do**
2: Compute $u_i := u \bmod p_i^{e_i}$ in G_i
3: **end for**
4: Compute $u = CRT(u_1, \ldots, u_n)$.

1. **Codebook:** We compute and store all $u \bmod p_i^{e_i}$, sorted (in implicit representation).
2. **Dlog computation:** We search for $\left(\frac{\mathrm{ord}(\hat{P}_i)}{p_i^{e_i}}\right)\hat{Q}$ in our codebook.

Codebook for all subgroups of NIST P-256's Auxiliary Curve

Factor p_i	Bit-size	Codebook[GB]
$2 \cdot 3 \cdot 626663$	22	0.07
6487813	23	0.12
17752487	25	0.33
30034813	25	0.56
620378903	30	11.48
1316356273	31	24.35
4747815593	33	90.21
17399156003	35	330.58
131964961211	37	2507.33

Codebooks for Auxiliary Curves

Curve	B	Codebook
NIST P-256	37 bit	**3.0 TB**
BLS12-381	36 bit	1.9 TB
BN(2,254)	39 bit	6.0 TB
brainpoolP256t1	39 bit	28.2 TB

Curve	B	Codebook
Curve25519	37	4.1 TB
GOST R 34.10	37	3.0 TB
secp256k1	37	3.1 TB
SM2	39	10.0 TB

Curve	B	Codebook
NIST P-384	65	613,705,033 TB
NIST P-521	110	29,923,937,044,117,456,000,000 TB

Takeaway: Codebook Construction

- Having smoothness below 40 bit gives highly practical codebooks.

Maurer-Wolf goes Real World Crypto: Contribution 3

1 Auxiliary Curve Construction

We explicitly compute auxiliary curves $\hat{E}(\mathbb{F}_q)$ with **smooth order** for most elliptic curve standards (focus on 256-bit).

2 Efficient Auxiliary Dlog Computation

For 256-bit curves our auxiliary curves allows us to precompute and store a **codebook of all dlogs** for $\hat{E}(\mathbb{F}_q)$.

3 Minimizing DH-Oracle Calls

We optimize for minimal number of oracle calls, thereby achieving a **practically tight reduction** DH $\Rightarrow$ Dlog.

Dlog computation / Minimizing DH-oracle calls

- Recall that dlog extraction requires computation of $\left(\frac{\mathrm{ord}(\hat{P}_i)}{p_i^{e_i}}\right)\hat{Q}$.
- This computation needs DH-oracle calls $\mathcal{O}(\cdot,\cdot)$, that we simulate efficiently.
- We minimize number of DH-oracle calls by grouping $p_i^{e_i}$ to balance sizes.
- Example for Curve25519:

$$\mathrm{ord}(\hat{P}) = 255833749 \cdot \left(2^2 \cdot 53 \cdot 1563739\right) \cdot (3 \cdot 7 \cdot 1013 \cdot 26339) \cdot (23 \cdot 25395859) \cdot 1073269973 \cdot 36776837081 \cdot 49009622279 \cdot 134777522111$$

Tightness of DH $\Rightarrow$ Dlog

$$\mathbf{T}_{\mathrm{Dlog}} \approx \textbf{DH-oracle calls} \cdot \mathbf{T}_{\mathrm{DH}}$$

Efficiency of Our Approach

Curve	DH-calls	T_{Dlog}
NIST P-256	**23,656**	**32s**
BLS12-381	20,397	28s
BN(2,254)	22,036	30s
brainpoolP256t1	16,542	22s

Curve	DH-calls	T_{Dlog}
Curve25519	19,091	26s
GOST R 34.10	17,519	24s
SM2	19,592	27s
secp256k1	21,868	30s

Takeaway: Dlog computation & Tightness

- All Dlog computations require less than 2^{15} DH-oracle calls.
- Experimentally, we observe $\mathbf{T}_{\text{Dlog}} \approx \textbf{DH-oracle calls} \cdot \mathbf{T}_{\text{DH}}$.
- Assume that any Dlog algorithm has at least 128 bit complexity.
- **Then any DH algorithm has at least 113 bit complexity.**

Wrap-Up

Lessons Learned

- 256-bit curves allow for efficient Auxiliary Curve constructions.
- This leads to concrete security guarantees for the DH problem.
- *Personal Opinion:* We need more security reductions that go practical.

Open Problems

- Our method provides less strong DH guarantees for larger group orders.
- Could be resolved by more computing power / smarter algorithms(?).
- *Dark side:* Are there DH-oracles in practice for Dlog computation?