# Reducing the Discrete Logarithm Problem to the Diffie–Hellman Problem Using Auxiliary Compression

**Sergey Agievich**

Research Institute for Applied Problems
of Mathematics and Informatics

Belarusian State University

CTCrypt-2013, Ekaterinburg, 25 June 2013

# DL and DH

Let $G$ be a cyclic group of prime order.

descr($G$):
an order $p = |G|$;
a generator $g$;
a program for multiplication in $G$ (can be done in time $(\log p)^{O(1)}$).

**Discrete Logarithm problem (DL)**:
$(\text{descr}(G), g^a) \mapsto a$ $(a \in \mathbb{F}_p = \{0, 1, \ldots, p-1\})$

**(computational) Diffie-Hellman Problem (DH)**:
$(\text{descr}(G), g^a, g^b) \mapsto g^{ab}$

**Refined problems (for a fixed $G$)**:
$\mathbf{DL}(G)$: $g^a \mapsto a$
$\mathbf{DH}(G)$: $(g^a, g^b) \mapsto g^{ab}$

# Reductions

Let $P_1$, $P_2$ be computational problems.

**Reduction (probabilistic polynomial-time) of $P_1$ to $P_2$ ($P_1 \leqslant P_2$)**:

a probabilistic algorithm that solves $P_1$;

a) with a non-negligible probability;

b) in polynomial time;

c) in polynomially many calls to an algorithm which solves $P_2$.

$\mathbf{DH}(G) \leqslant \mathbf{DL}(G)$:

$a \leftarrow \mathbf{DL}(G)(g^a)$

$g^{ab} \leftarrow (g^b)^a$

$\mathbf{DL}(G) \overset{?}{\leqslant} \mathbf{DH}(G)$ (the unsolved problem)

# Auxiliary problems

**The compound problem** $P_1 \wedge P_2$:

to solve $P_1 \wedge P_2$ one must be able to solve $P_1$ as well $P_2$.

**Theorem** (Cherepnev). $\mathbf{DL}(G) \leqslant \mathbf{Factor} \wedge \mathbf{DH}$.

[More precisely, solving $\mathbf{Factor}$ polynomially many times, one can define family $\mathcal{H}$ of cyclic groups of prime orders $\leqslant p$ and then solve $\mathbf{DL}(G)$ with a non-negligible probability in polynomial time and in polynomially many calls to an algorithm which solves $\mathbf{DH}(H)$, $H \in \mathcal{H}$.]

**Theorem** (Maurer–Wolf). $\mathbf{DL}(G) \leqslant \mathbf{AuxGroup} \wedge \mathbf{DH}(G)$.

$\mathbf{Factor}$ (integer **Factor**ization) and $\mathbf{AuxGroup}$ (**Aux**iliary **Group**) are *auxiliary* problems which help to solve $\mathbf{DL}$ using $\mathbf{DH}$ as subroutine.

# Results

We introduce a new problem "Auxiliary compression" ($\mathbf{AuxCompr}$) and obtain the following results.

**Proposition 1**. $\mathbf{DL}(G) \leqslant \mathbf{AuxCompr} \wedge \mathbf{DH}(G)$.

**Proposition 2**. $\mathbf{AuxCompr} \leqslant \mathbf{AuxGroup}$.

These propositions yield that for reducing $\mathbf{DL}(G)$ to $\mathbf{DH}(G)$ it is sufficient to solve the auxiliary problem $\mathbf{AuxCompr}$ that is not harder than the previously used problem $\mathbf{AuxGroup}$.

# Arithmetic circuits

**Arithmetic circuit over $\mathbb{F}_p$:**

a directed acyclic graph $\Phi$ such that

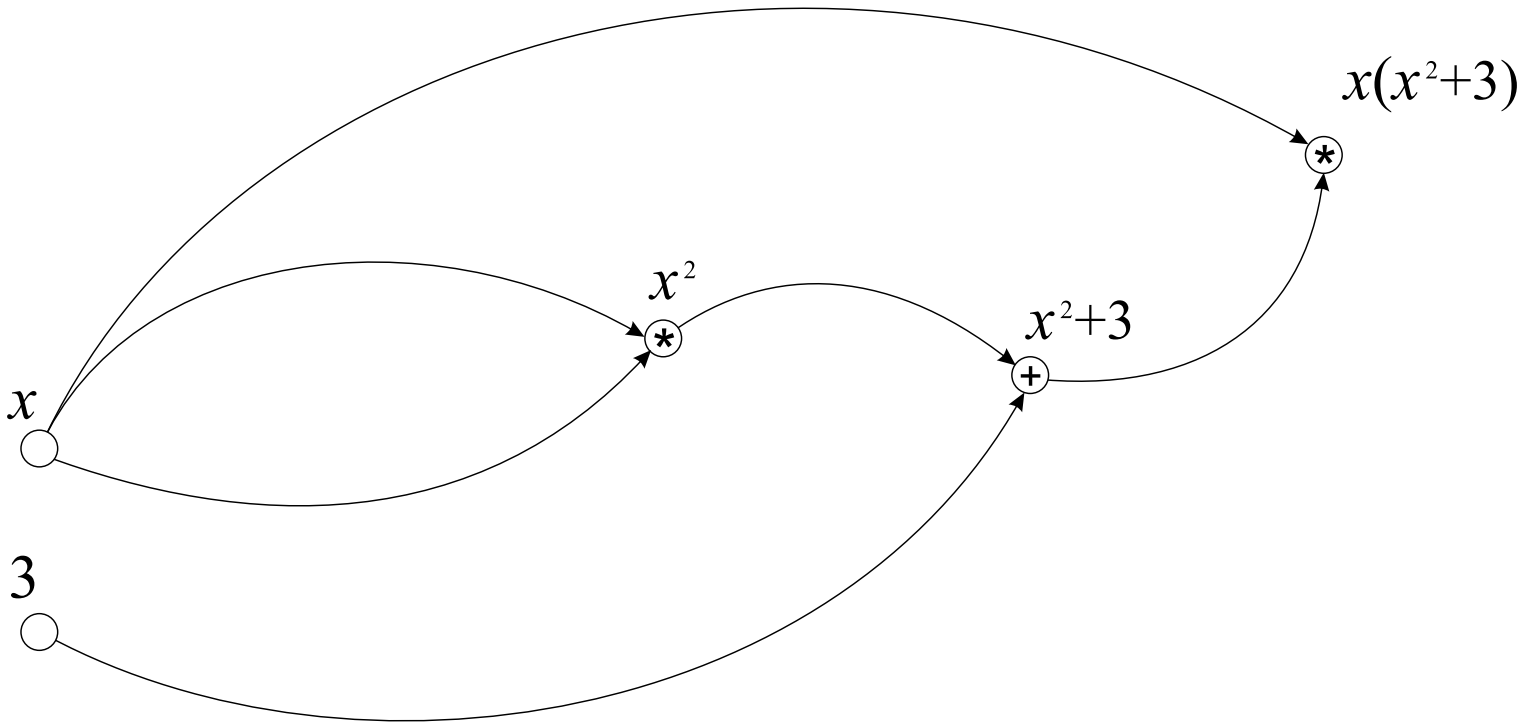
1) every vertice of $\Phi$ (*gate*) has in-degree 0 or 2;
2) a gate of in-degree 0 (*leaf*) is labeled by either a variable $x$; or a field element $c \in \mathbb{F}_p$;
3) a gate of in-degree 2 is labeled by either $*$ or $+$.

**Size of $\Phi$** (denoted by $|\Phi|$):

the number of edges in $\Phi$.

Every gate $v$ of $\Phi$ represents a polynomial $f_v(x) \in \mathbb{F}_p[x]$. Let $\Phi_v$ be the sub-circuit of $\Phi$ rooted at $v$. Choosing $a \in \mathbb{F}_p$ and setting $x = a$ we can calculate $f_v(a)$ by $\frac{1}{2}|\Phi_v|$ additions and multiplications in $\mathbb{F}_p$.

# Arithmetic circuits: an example



# Implicit calculations

$a, b \in \mathbb{F}_p$

$a \rightsquigarrow A = g^a$, $b \rightsquigarrow B = g^b$

$A$ is an **implicit representation** of $a$.

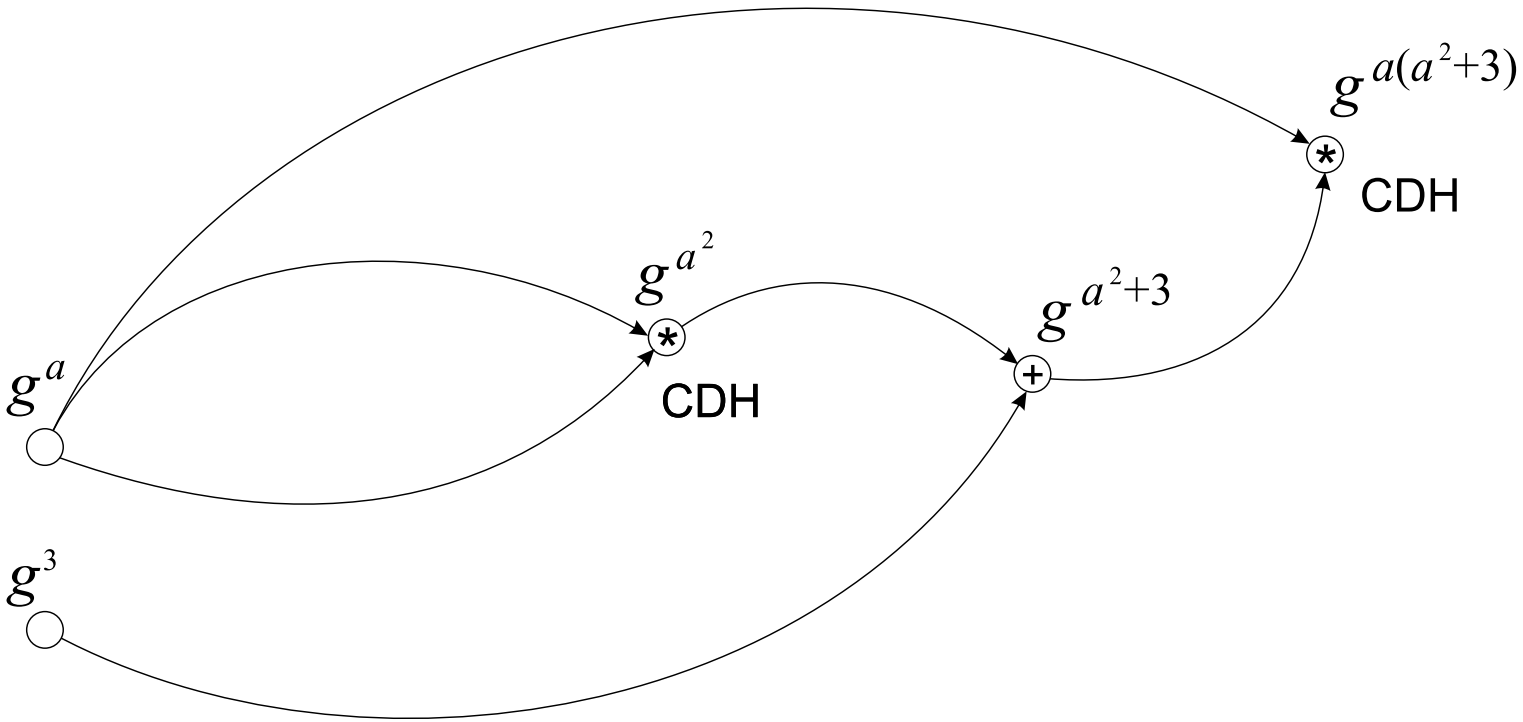
**Implicit calculations** in $\Phi$:

1) $a + b \rightsquigarrow A * B$;
2) $a * b \rightsquigarrow \mathbf{DH}(G, A, B)$;
3) $a = b \Leftrightarrow A = B$.

To determine an implicit representation of $f_v(a)$ for a given $A = g^a$ it is sufficient to use algorithms that multiplies in $G$ and solves $\mathbf{DH}(G)$. We need to make $\frac{1}{2}|\Phi_v|$ calls to these algorithms in total.

Retrieving $f_v(a)$ from its implicit representation is the hard discrete logarithm problem. But we can compare $f_v(a)$ with an arbitrary $c \in \mathbb{F}_p$.

# Implicit calculations: an example



# Auxiliary compression

**Definition**. A circuit $\Phi$ over $\mathbb{F}_p$ defines *invertible compression*, if it has polynomial (in $\log p$) size and if there exists a probabilistic polynomial algorithm $I$, which with a non-negligible probability determines an input $a \in \mathbb{F}_p$ using comparisons $f_v(a) \stackrel{?}{=} c$, where $v$, a gate of $\Phi$, and $c \in \mathbb{F}_p$ can be chosen by the algorithm.

$S_v = \{f_v(a) : a \in \mathbb{F}_p\}$

**Compression**:

$$\exists u \text{ s.t. } |S_u| = (\log p)^{O(1)}$$

[If such gate does not exist and all $|S_u|$ are superpolynomial, then $I$ can not determine any $f_u(a)$ (and, therefore, $a$) with a non-negligible probability.]

**AuxCompr**:

given $p$, build an invertible compression circuit $\Phi$ over $\mathbb{F}_p$.

# Auxiliary group

**Auxiliary group** $H$ (sketch):

1) elements of $H$ are represented by vectors of $\mathbb{F}_p^n$, $n = (\log p)^{O(1)}$;
2) $H$ is a cyclic group with a known generator $(h_1, \ldots, h_n)$;
3) the order of $H$ is $B$-smooth, where $B = (\log p)^{O(1)}$;
4) there is a circuit $\Phi_0$ over $\mathbb{F}_p$ that defines a mapping $\mathbb{F}_p \to H$;
5) the circuit $\Phi_0$ is invertible;
6) there is a circuit $\Phi_1$ for multiplication in $H$;
7) there is a circuit $\Phi_2$ for comparison in $H$

(all circuits have polynomial size).

**AuxGroup**:

given $p$, build an auxiliary group over $\mathbb{F}_p$.

**Examples**:

$\mathbb{F}_p^*$ (if $p-1$ is smooth, denBoer);
$E(\mathbb{F}_p)$ (if an appropriate elliptic curve is founded, Maurer-Wolf).

## $\mathbf{AuxCompr} \leqslant \mathbf{AuxGroup}$

Let $H$ is a solution of $\mathbf{AuxGroup}$.

Let $|H| = \prod_{i=1}^{k} q_i$, where $q_i$ is small primes.

**Compression**:

1) $b_0 = \Phi_0(a)$;
2) $b_i = b_i^{q_i}$, $i = 1, \ldots, k$;
3) $b_k = b_0^{|H|}$ must be the neutral element of $H$.

**Inversion** $(i = k, k-1, \ldots, 1)$:

1) determine $q_i$th roots of $b_i$;
2) compare each root with $b_{i-1}$;
3) choose an appropriate root.

# Complexity of AuxCompr

**Theorem** (Boneh–Lipton). **AuxGroup** (as well as **AuxCompr**) can be solved in heuristically expected subexponential time

$$L_p(1/2, 2) = \exp((2 + o(1))\sqrt{\ln p \ln\ln p}).$$

Can we solve **AuxCompr** in polynomial time?

**Particular form of AuxCompr**:

given $p$, find polynomials $f_1, \ldots, f_k \in \mathbb{F}_p[x]$ s.t.

1) $k = (\log p)^{O(1)}$;
2) $\deg f_i = (\log p)^{O(1)}$;
3) $x^p - x \mid F_k(x)$,

where $F_i(x) = f_i(F_{i-1}(x))$, $i = k, k-1, \ldots, 1$, $F_0(x) = x$.

**Compression**: $F_k(a) = 0 \ \forall a \in \mathbb{F}_p$.

**Inversion**: starting from $b_k = 0$ find roots of $f_i(x) - b_i$ and choose an appropriate root $b_{i-1}$, $i = k, k-1, \ldots, 1$ ($a = b_0$).