Generic complexity in group theory

Alexei Miasnikov (McGill University)

Institute of Mathematics, Omsk, August, 2009

Outline

A. Generic case complexity.
   - Definitions.
   - Generic versus average.
   - Are the standard NP-complete problems generically easy?
   - Generic case NP-complete problems.
   - Generic complexity of undecidable problems.

B. Generic algorithms in group theory
   - Where do generic algorithms come from?
   - Search Word Problem revisited.
   - Why does Todd-Coxeter work?

C. With an eye on cryptography
   - Groups and semigroups with generically hard word problem.
   - Generic subgroups and cryptanalysis

Outline

A. Generic case complexity.
   - Definitions.
   - Generic versus average.
   - Are the standard NP-complete problems generically easy?
   - Generic case NP-complete problems.
   - Generic complexity of undecidable problems.

B. Generic algorithms in group theory
   - Where do generic algorithms come from?
   - Search Word Problem revisited.
   - Why does Todd-Coxeter work?

C. With an eye on cryptography
   - Groups and semigroups with generically hard word problem.
   - Generic subgroups and cryptanalysis

Outline

A. Generic case complexity.
   - Definitions.
   - Generic versus average.
   - Are the standard NP-complete problems generically easy?
   - Generic case NP-complete problems.
   - Generic complexity of undecidable problems.

B. Generic algorithms in group theory
   - Where do generic algorithms come from?
   - Search Word Problem revisited.
   - Why does Todd-Coxeter work?

C. With an eye on cryptography
   - Groups and semigroups with generically hard word problem.
   - Generic subgroups and cryptanalysis

Stratified computational problems

Recall, that a stratified distributional problem $\mathcal{D}$ is a computational problem $\mathcal{D} = (L, I)$ such that:

- The set of inputs I comes equipped with a size function $s : I \to \mathbb{N}$;
- The size function s gives rise to a spherical stratification of I:

$$I = \cup_n I_n$$

where $I_n = \{w \in I \mid s(w) = n\}$ is a sphere of radius n.

Stratified computational problems

Recall, that a stratified distributional problem $\mathcal{D}$ is a computational problem $\mathcal{D} = (L, I)$ such that:

- The set of inputs I comes equipped with a size function $s : I \to \mathbb{N}$;
- The size function s gives rise to a spherical stratification of I:

$$I = \cup_n I_n$$

where $I_n = \{w \in I \mid s(w) = n\}$ is a sphere of radius n.

Stratified computational problems

Recall, that a stratified distributional problem $\mathcal{D}$ is a computational problem $\mathcal{D} = (L, I)$ such that:

- The set of inputs I comes equipped with a size function $s : I \to \mathbb{N}$;
- The size function s gives rise to a spherical stratification of I:

$$I = \cup_n I_n$$

where $I_n = \{w \in I \mid s(w) = n\}$ is a sphere of radius n.

Stratified computational problems

- Every sphere I_n comes equipped with an (atomic) distribution $\mu_n : I_n \longrightarrow \mathbb{R}^+$.
- The set $\mu = \{\mu_n \mid n \in \mathbb{N}\}$ is called en ensemble of spherical distributions on I.

Similarly, one can defined a distributional problem with volume stratification with an ensemble of volume distributions.

From now on we fix a stratified distributional problem $\mathcal{D}$ with an ensemble of uniform distributions μ_n.

Stratified computational problems

- Every sphere I_n comes equipped with an (atomic) distribution $\mu_n : I_n \longrightarrow \mathbb{R}^+$.
- The set $\mu = \{\mu_n \mid n \in \mathbb{N}\}$ is called en ensemble of spherical distributions on I.

Similarly, one can defined a distributional problem with volume stratification with an ensemble of volume distributions.

From now on we fix a stratified distributional problem $\mathcal{D}$ with an ensemble of uniform distributions μ_n.

Stratified computational problems

- Every sphere I_n comes equipped with an (atomic) distribution $\mu_n : I_n \to \mathbb{R}^+$.
- The set $\mu = \{\mu_n \mid n \in \mathbb{N}\}$ is called en ensemble of spherical distributions on I.

Similarly, one can defined a distributional problem with volume stratification with an ensemble of volume distributions.

From now on we fix a stratified distributional problem $\mathcal{D}$ with an ensemble of uniform distributions μ_n.

Stratified computational problems

- Every sphere I_n comes equipped with an (atomic) distribution $\mu_n : I_n \to \mathbb{R}^+$.
- The set $\mu = \{\mu_n \mid n \in \mathbb{N}\}$ is called en ensemble of spherical distributions on I.

Similarly, one can defined a distributional problem with volume stratification with an ensemble of volume distributions.

From now on we fix a stratified distributional problem $\mathcal{D}$ with an ensemble of uniform distributions μ_n.

Convergence rates

For a subset R of I put

$$R_n = \frac{|R \cap I_n|}{|I_n|}$$

and define the asymptotic density of R by

$$\rho(R) = \lim_{n\to\infty} R_n = \lim_{n\to\infty} \frac{|R \cap I_n|}{|I_n|}$$

exists.

Convergence rates

For a subset R of I put

$$R_n = \frac{|R \cap I_n|}{|I_n|}$$

and define the asymptotic density of R by

$$\rho(R) = \lim_{n\to\infty} R_n = \lim_{n\to\infty} \frac{|R \cap I_n|}{|I_n|}$$

exists.

Differentiating by convergence rates

We say that R has asymptotic density $\rho(R)$ with the convergence rate

1) *polynomial* if $|\rho(R) - R_n| = O(n^{-k})$ for some constant k;

2) *superpolynomial* if $|\rho(R) - R_n| = o(n^{-k})$ for any natural k;

3) *exponential* if $|\rho(R) - R_n| = O(c^n)$ for some $0 < c < 1$.

Of course, one can introduce other degrees of convergence: subpolynomial, superexponential, etc.

Differentiating by convergence rates

We say that R has asymptotic density $\rho(R)$ with the convergence rate

1) *polynomial* if $|\rho(R) - R_n| = O(n^{-k})$ for some constant k;
2) *superpolynomial* if $|\rho(R) - R_n| = o(n^{-k})$ for any natural k;
3) *exponential* if $|\rho(R) - R_n| = O(c^n)$ for some $0 < c < 1$.

Of course, one can introduce other degrees of convergence: subpolynomial, superexponential, etc.

Differentiating by convergence rates

We say that R has asymptotic density $\rho(R)$ with the convergence rate

1) *polynomial* if $|\rho(R) - R_n| = O(n^{-k})$ for some constant k;

2) *superpolynomial* if $|\rho(R) - R_n| = o(n^{-k})$ for any natural k;

3) *exponential* if $|\rho(R) - R_n| = O(c^n)$ for some $0 < c < 1$.

Of course, one can introduce other degrees of convergence: subpolynomial, superexponential, etc.

Generic sets of different sizes

One can distinguish various generic sets with respect to their convergence rates.

We say that a generic set R is:

- *superpolynomial* if its convergence rate is superpolynomial.
- *exponential* if its convergence rate is exponential.

Sometimes we also use the following terminology:

superpolynomially generic $\Longleftrightarrow$ *strongly generic*,

exponentially generic $\Longleftrightarrow$ - *super generic*.

Generic sets of different sizes

One can distinguish various generic sets with respect to their convergence rates.

We say that a generic set R is:

- *superpolynomial* if its convergence rate is superpolynomial.
- *exponential* if its convergence rate is exponential.

Sometimes we also use the following terminology:

superpolynomially generic $\Longleftrightarrow$ *strongly generic*,

exponentially generic $\Longleftrightarrow$ - *super generic*.

Generic sets of different sizes

One can distinguish various generic sets with respect to their convergence rates.

We say that a generic set R is:

- *superpolynomial* if its convergence rate is superpolynomial.
- *exponential* if its convergence rate is exponential.

Sometimes we also use the following terminology:

superpolynomially generic $\Longleftrightarrow$ *strongly generic*,

exponentially generic $\Longleftrightarrow$ - *super generic*.

Generic sets of different sizes

One can distinguish various generic sets with respect to their convergence rates.

We say that a generic set R is:

- *superpolynomial* if its convergence rate is superpolynomial.
- *exponential* if its convergence rate is exponential.

Sometimes we also use the following terminology:

superpolynomially generic $\Longleftrightarrow$ *strongly generic*,

exponentially generic $\Longleftrightarrow$ - *super generic*.

Generic sets of different sizes

One can distinguish various generic sets with respect to their convergence rates.

We say that a generic set R is:

- *superpolynomial* if its convergence rate is superpolynomial.
- *exponential* if its convergence rate is exponential.

Sometimes we also use the following terminology:

superpolynomially generic $\Longleftrightarrow$ *strongly generic*,

exponentially generic $\Longleftrightarrow$ - *super generic*.

Generic sets of different sizes

One can distinguish various generic sets with respect to their convergence rates.

We say that a generic set R is:

- *superpolynomial* if its convergence rate is superpolynomial.
- *exponential* if its convergence rate is exponential.

Sometimes we also use the following terminology:

superpolynomially generic $\Longleftrightarrow$ *strongly generic*,

exponentially generic $\Longleftrightarrow$ - *super generic*.

Generic sets of different sizes

One can distinguish various generic sets with respect to their convergence rates.

We say that a generic set R is:

- *superpolynomial* if its convergence rate is superpolynomial.
- *exponential* if its convergence rate is exponential.

Sometimes we also use the following terminology:

superpolynomially generic $\Longleftrightarrow$ *strongly generic*,

exponentially generic $\Longleftrightarrow$ - *super generic*.

Where do they come from?

"Generic" algorithms are the usual algorithms, but they come from a different, **generic world**.

This world is much more simple and tractable, in fact, it is the same old world with all the singularities removed.

The world through the eyes of a very happy and very naive person.

To design an algorithm which is generically fast:

- Study the structure and properties of generic objects,
- design an algorithm based on these properties.

Where do they come from?

"Generic" algorithms are the usual algorithms, but they come from a different, **generic world**.

This world is much more simple and tractable, in fact, it is the same old world with all the singularities removed.

The world through the eyes of a very happy and very naive person.

To design an algorithm which is generically fast:

- Study the structure and properties of generic objects,
- design an algorithm based on these properties.

Where do they come from?

"Generic" algorithms are the usual algorithms, but they come from a different, **generic world**.

This world is much more simple and tractable, in fact, it is the same old world with all the singularities removed.

The world through the eyes of a very happy and very naive person.

To design an algorithm which is generically fast:

- Study the structure and properties of generic objects,
- design an algorithm based on these properties.

Coset enumeration

One of the most important question about a group $G = \langle X \mid R \rangle$ is whether or not G is finite.

This problem is undecidable.

However, the well known coset enumeration procedure (Todd-Coxeter algorithms) gives the right answer when G is finite by constructing the Cayley diagram of G.

If G is infinite, coset enumeration runs forever.

Thus, coset enumeration is the Search variation of an undecidable problem, so there is no any computable upper bound on its time complexity.

Coset enumeration

One of the most important question about a group $G = \langle X \mid R \rangle$ is whether or not G is finite.

This problem is undecidable.

However, the well known coset enumeration procedure (Todd-Coxeter algorithms) gives the right answer when G is finite by constructing the Cayley diagram of G.

If G is infinite, coset enumeration runs forever.

Thus, coset enumeration is the Search variation of an undecidable problem, so there is no any computable upper bound on its time complexity.

Coset enumeration

One of the most important question about a group $G = \langle X \mid R \rangle$ is whether or not G is finite.

This problem is undecidable.

However, the well known coset enumeration procedure (Todd-Coxeter algorithms) gives the right answer when G is finite by constructing the Cayley diagram of G.

If G is infinite, coset enumeration runs forever.

Thus, coset enumeration is the Search variation of an undecidable problem, so there is no any computable upper bound on its time complexity.

Coset enumeration

One of the most important question about a group $G = \langle X \mid R \rangle$ is whether or not G is finite.

This problem is undecidable.

However, the well known coset enumeration procedure (Todd-Coxeter algorithms) gives the right answer when G is finite by constructing the Cayley diagram of G.

If G is infinite, coset enumeration runs forever.

Thus, coset enumeration is the Search variation of an undecidable problem, so there is no any computable upper bound on its time complexity.

Coset enumeration

One of the most important question about a group $G = \langle X \mid R \rangle$ is whether or not G is finite.

This problem is undecidable.

However, the well known coset enumeration procedure (Todd-Coxeter algorithms) gives the right answer when G is finite by constructing the Cayley diagram of G.

If G is infinite, coset enumeration runs forever.

Thus, coset enumeration is the Search variation of an undecidable problem, so there is no any computable upper bound on its time complexity.

Coset enumeration

Nevertheless, under the circumstances, coset enumeration is surprisingly practical and has been used successfully since well before the advent of computers.

Why?

The answer, it seems, comes from generic complexity.

For simplicity, I will assume that $G = 1$.

Coset enumeration

Nevertheless, under the circumstances, coset enumeration is surprisingly practical and has been used successfully since well before the advent of computers.

Why?

The answer, it seems, comes from generic complexity.

For simplicity, I will assume that $G = 1$.

Coset enumeration

Nevertheless, under the circumstances, coset enumeration is surprisingly practical and has been used successfully since well before the advent of computers.

Why?

The answer, it seems, comes from generic complexity.

For simplicity, I will assume that $G = 1$.

Coset enumeration

Nevertheless, under the circumstances, coset enumeration is surprisingly practical and has been used successfully since well before the advent of computers.

Why?

The answer, it seems, comes from generic complexity.

For simplicity, I will assume that $G = 1$.

Coset enumeration

Let $\{a_1, \ldots\}$ be a countable set of generators.

The set of inputs, I, for our coset enumeration procedure, $\mathcal{P}$, consists of all finite sequences, $r_1, \ldots, r_t$ of nontrivial freely reduced words such that

$$G = \langle a_1, \ldots, a_s \mid r_1, \ldots, r_t \rangle$$

is the trivial group.

Coset enumeration

There are many different coset enumeration procedures, but they are all roughly the same. Here is a simple one.

Given a presentation (**??**), the coset enumeration procedure $\mathcal{P}$ works by producing a sequence of approximations $\Gamma_0, \Gamma_1 \ldots$ to Γ, the Cayley diagram of G. Γ_0 consists of a single unvisited vertex v_0 and no edges.

$\mathcal{P}$ constructs Γ_{i+1} from Γ_i by visiting each unvisited vertex and for each cyclic permutation r' of every relator r adding a cycle with label r' beginning at that vertex.

The unvisited vertex is then marked as visited.

After this process in complete, all new vertices in the cycles which have been added are marked as unvisited.

The graph is then folded to obtain Γ_{i+1}.

During the folding the property of being visited or unvisited is

Coset enumeration

Because the relators include all generators, the process of adding cycles insures that every visited vertex of Γ_{i+1} has exactly one inedge and one outedge labeled by each generator.

At each step there is a homomorphism of labeled graphs from Γ_i to Γ with v_0 mapping to the identity vertex.

Coset enumeration

Since $G = 1$, each generator is freely equal to a product of conjugates of relators.

Let ℓ be the maximum length of the conjugating elements required to express each generator in this way.

All conjugates of relators by words of length at most ℓ are traceable in $\Gamma_{\ell+1}$.

Lemma

If every generator is freely equal to a product of conjugates of relators with conjugating elements all of length at most ℓ, then $\mathcal{P}$ verifies $G = 1$ in at most $\ell + 1$ passes.

Generic complexity of coset enumeration

Theorem [Gilman]

On a generic subset of I coset enumeration converges in at most $(1 + 2m)$ passes, where m is the maximum relator length.

Corollary

On a generic subset of I the time complexity of $\mathcal{P}$ for an input with k relators of length at most m is $O((km^2)^{2m+2})$.

The classical semigroups with undecidable WP

There are several famous finitely presented semigroups with undecidable word problem due to Markov, Post, Tseitin, Matiyasevich, Makanin.

The Tseitin semigroup $\mathfrak{T}$ presented by the generators a, b, c, d, e and defining relations

$$ac = ca, ad = da, bc = cb, bd = db,$$

$$ce = eca, de = edb, cca = ccae.$$

is the shortest finitely presented semigroup with undecidable Word Problem.

I will refer to all these known examples as to the **"Classical Monsters"**.

The classical semigroups with undecidable WP

There are several famous finitely presented semigroups with undecidable word problem due to Markov, Post, Tseitin, Matiyasevich, Makanin.

The Tseitin semigroup $\mathfrak{T}$ presented by the generators a, b, c, d, e and defining relations

$$ac = ca, ad = da, bc = cb, bd = db,$$

$$ce = eca, de = edb, cca = ccae.$$

is the shortest finitely presented semigroup with undecidable Word Problem.

I will refer to all these known examples as to the **"Classical Monsters"**.

The classical semigroups with undecidable WP

There are several famous finitely presented semigroups with undecidable word problem due to Markov, Post, Tseitin, Matiyasevich, Makanin.

The Tseitin semigroup $\mathfrak{T}$ presented by the generators a, b, c, d, e and defining relations

$$ac = ca, ad = da, bc = cb, bd = db,$$

$$ce = eca, de = edb, cca = ccae.$$

is the shortest finitely presented semigroup with undecidable Word Problem.

I will refer to all these known examples as to the **"Classical Monsters"**.

Generic complexity of WP in the standard monsters

Theorem [Myasnikov, Ushakov, Wook Won]

The Word Problem in the classical monsters has polynomial time generic case complexity.

A philosophical explanation for this phenomenon

The main reason for this phenomenon is that to interpret a hard problem (say, the Halting Problem for Turing machines) one brings into the finite presentation of a semigroup a lot of "garbage" (extra generators and relators that serve to simulate the Turing machine computation) that makes the Word problem easy on most inputs.

Though, the generic decision algorithm is almost obvious, it is surprisingly hard to show that it works on a generic subset.

A philosophical explanation for this phenomenon

The main reason for this phenomenon is that to interpret a hard problem (say, the Halting Problem for Turing machines) one brings into the finite presentation of a semigroup a lot of "garbage" (extra generators and relators that serve to simulate the Turing machine computation) that makes the Word problem easy on most inputs.

Though, the generic decision algorithm is almost obvious, it is surprisingly hard to show that it works on a generic subset.

Cloning of the Word Problem in semigroups

For a semigroup

$$\mathfrak{S} = \langle a_1, \ldots, a_n | r_1 = s_1, \ldots, r_k = s_k \rangle = \langle A \mid R \rangle$$

define a new semigroup

$$\mathfrak{S}_x = \langle A, x | R, x = xa_1, \ldots, x = xa_n, x = xx \rangle,$$

where x is a letter $x \notin A$. Denote $A_x = A \cup \{x\}$.

The inclusion map $A \to A_x$ extends to an embedding of semigroups $\mathfrak{S} \to \mathfrak{S}_x$.

Cloning of the Word Problem in semigroups

For a semigroup

$$\mathfrak{S} = \langle a_1, \ldots, a_n | r_1 = s_1, \ldots, r_k = s_k \rangle = \langle A \mid R \rangle$$

define a new semigroup

$$\mathfrak{S}_x = \langle A, x | R, x = xa_1, \ldots, x = xa_n, x = xx \rangle,$$

where x is a letter $x \notin A$. Denote $A_x = A \cup \{x\}$.

The inclusion map $A \to A_x$ extends to an embedding of semigroups $\mathfrak{S} \to \mathfrak{S}_x$.

Semigroups with super-undecidable WP

Theorem [Myasnikov, Rybalov]

If the word problem in $\mathfrak{S}$ is undecidable then the word problem in $\mathfrak{S}_x$ is super-undecidable (undecidable on every generic subset of $\mathfrak{S}_x$).

Example

The clone $\mathfrak{T}_x$ of the Tseitin semigroup has 6 generators and 13 relators whose total length is equal to 49. Its WP is super-undecidable.

Semigroups with super-undecidable WP

Theorem [Myasnikov, Rybalov]

If the word problem in $\mathfrak{S}$ is undecidable then the word problem in $\mathfrak{S}_x$ is super-undecidable (undecidable on every generic subset of $\mathfrak{S}_x$).

Example

The clone $\mathfrak{T}_x$ of the Tseitin semigroup has 6 generators and 13 relators whose total length is equal to 49. Its WP is super-undecidable.

Reference

Alexei G. Myasnikov, Alexander N. Rybalov,

Generic complexity of undecidable problems,

Journal of Symbolic Logic, v. 73, 2, 2008, pp.656-673.

The classical monsters in groups

There are several famous finitely presented groups with undecidable word problem (the **classical monsters**) due to Novikov, Boon, Borisov, and Collins.

Theorem

The Word Problem in the classical monsters has polynomial time generic case complexity.

Groups with strongly undecidable WP

Theorem [Gilman, Myasnikov, Osin]

If G is a finitely generated recursively presented amenable group with an unsolvable word problem, then the word problem is not decidable on any exponentially generic set.

Kharlampovich has shown that there exists a finitely presented solvable group K with unsolvable word problem.

Corollary

The word problem in K is undecidable on any exponentially generic set.

Groups with strongly undecidable WP

Theorem [Gilman, Myasnikov, Osin]

If G is a finitely generated recursively presented amenable group with an unsolvable word problem, then the word problem is not decidable on any exponentially generic set.

Kharlampovich has shown that there exists a finitely presented solvable group K with unsolvable word problem.

Corollary

The word problem in K is undecidable on any exponentially generic set.

Problem

Problem

Is there a finitely presented group with a super-undecidable word problem?

Commutator Key Exchange

The Commutator Key Exchange was introduced by Anshel-Anshel-Goldfeld in 1990's (published in 1999).

Ingenious idea.

Generated a lot of research in braid groups (and not only).

Never been seriously studied as a crypto scheme.

Easily broken (or claimed to be broken).

Abandoned.

Currently, in the Rehabilitation process.

Commutator Key Exchange

The Commutator Key Exchange was introduced by Anshel-Anshel-Goldfeld in 1990's (published in 1999).

Ingenious idea.

Generated a lot of research in braid groups (and not only).

Never been seriously studied as a crypto scheme.

Easily broken (or claimed to be broken).

Abandoned.

Currently, in the Rehabilitation process.

Commutator Key Exchange

The Commutator Key Exchange was introduced by Anshel-Anshel-Goldfeld in 1990's (published in 1999).

Ingenious idea.

Generated a lot of research in braid groups (and not only).

Never been seriously studied as a crypto scheme.

Easily broken (or claimed to be broken).

Abandoned.

Currently, in the Rehabilitation process.

Commutator Key Exchange

The Commutator Key Exchange was introduced by Anshel-Anshel-Goldfeld in 1990's (published in 1999).

Ingenious idea.

Generated a lot of research in braid groups (and not only).

Never been seriously studied as a crypto scheme.

Easily broken (or claimed to be broken).

Abandoned.

Currently, in the Rehabilitation process.

Commutator Key Exchange

The Commutator Key Exchange was introduced by Anshel-Anshel-Goldfeld in 1990's (published in 1999).

Ingenious idea.

Generated a lot of research in braid groups (and not only).

Never been seriously studied as a crypto scheme.

Easily broken (or claimed to be broken).

Abandoned.

Currently, in the Rehabilitation process.

Commutator Key Exchange

The Commutator Key Exchange was introduced by Anshel-Anshel-Goldfeld in 1990's (published in 1999).

Ingenious idea.

Generated a lot of research in braid groups (and not only).

Never been seriously studied as a crypto scheme.

Easily broken (or claimed to be broken).

Abandoned.

Currently, in the Rehabilitation process.

Commutator Key Exchange

The Commutator Key Exchange was introduced by Anshel-Anshel-Goldfeld in 1990's (published in 1999).

Ingenious idea.

Generated a lot of research in braid groups (and not only).

Never been seriously studied as a crypto scheme.

Easily broken (or claimed to be broken).

Abandoned.

Currently, in the Rehabilitation process.

Commutator Key Exchange

Public:

A group $G = \langle X \mid R \rangle$ with a normal form $N : w \longrightarrow N(w)$ and two subgroups $A = \langle a_1, \ldots, a_m \rangle, \quad B = \langle b_1, \ldots, b_n \rangle$

Alice:

takes a (secret) word $a = u(a_1, \ldots, a_m)$ in alphabet $A^{\pm 1}$, computes $a^{-1} b_1 a, \ldots, a^{-1} b_n a$, encodes $a^{-1} b_i a$ by $N(a^{-1} b_i a)$, and makes public:

$$N(a^{-1} b_1 a), \ldots, N(a^{-1} b_n a)$$

Bob:

takes a (secret) word $b = v(b_1, \ldots, b_n)$ in alphabet $B^{\pm 1}$, encodes the conjugates $b^{-1} a_i b$ by their normal forms, and makes public:

$$N(b^{-1} a_1 b), \ldots, N(b^{-1} a_m b).$$

Commutator Key Exchange

Public:

A group $G = \langle X \mid R \rangle$ with a normal form $N : w \longrightarrow N(w)$ and two subgroups $A = \langle a_1, \ldots, a_m \rangle, \quad B = \langle b_1, \ldots, b_n \rangle$

Alice:

takes a (secret) word $a = u(a_1, \ldots, a_m)$ in alphabet $A^{\pm 1}$, computes $a^{-1}b_1a, \ldots, a^{-1}b_na$, encodes $a^{-1}b_ia$ by $N(a^{-1}b_ia)$, and makes public:

$$N(a^{-1}b_1a), \ldots, N(a^{-1}b_na)$$

Bob:

takes a (secret) word $b = v(b_1, \ldots, b_n)$ in alphabet $B^{\pm 1}$, encodes the conjugates $b^{-1}a_ib$ by their normal forms, and makes public:

$$N(b^{-1}a_1b), \ldots, N(b^{-1}a_mb).$$

Commutator Key Exchange

Public:

A group $G = \langle X \mid R \rangle$ with a normal form $N : w \longrightarrow N(w)$ and two subgroups $A = \langle a_1, \ldots, a_m \rangle, \quad B = \langle b_1, \ldots, b_n \rangle$

Alice:

takes a (secret) word $a = u(a_1, \ldots, a_m)$ in alphabet $A^{\pm 1}$, computes $a^{-1}b_1a, \ldots, a^{-1}b_na$, encodes $a^{-1}b_ia$ by $N(a^{-1}b_ia)$, and makes public:

$$N(a^{-1}b_1a), \ldots, N(a^{-1}b_na)$$

Bob:

takes a (secret) word $b = v(b_1, \ldots, b_n)$ in alphabet $B^{\pm 1}$, encodes the conjugates $b^{-1}a_ib$ by their normal forms, and makes public:

$$N(b^{-1}a_1b), \ldots, N(b^{-1}a_mb).$$

Commutator Key Exchange

Alice:

has $a = u(a_1, \ldots, a_m)$ and can compute
$b^{-1}ab = u(b^{-1}a_1b, \ldots, b^{-1}a_mb)$

Bob:

has $b = v(b_1, \ldots, b_n)$ and can compute
$a^{-1}ba = v(a^{-1}b_1a, \ldots, a^{-1}b_na)$

Shared secret key:

$a^{-1}b^{-1}ab = [a, b] = (a^{-1}ba)^{-1}b$

Commutator Key Exchange

Alice:

has $a = u(a_1, \ldots, a_m)$ and can compute
$b^{-1}ab = u(b^{-1}a_1b, \ldots, b^{-1}a_mb)$

Bob:

has $b = v(b_1, \ldots, b_n)$ and can compute
$a^{-1}ba = v(a^{-1}b_1a, \ldots, a^{-1}b_na)$

Shared secret key:

$a^{-1}b^{-1}ab = [a, b] = (a^{-1}ba)^{-1}b$

Commutator Key Exchange

Alice:

has $a = u(a_1, \ldots, a_m)$ and can compute
$b^{-1}ab = u(b^{-1}a_1b, \ldots, b^{-1}a_mb)$

Bob:

has $b = v(b_1, \ldots, b_n)$ and can compute
$a^{-1}ba = v(a^{-1}b_1a, \ldots, a^{-1}b_na)$

Shared secret key:

$a^{-1}b^{-1}ab = [a, b] = (a^{-1}ba)^{-1}b$

AAG scheme

In the original AAG scheme G is the braid group B_n.

$$B_n = \left\langle \sigma_1, \ldots, \sigma_{n-1} \;\middle|\; \begin{array}{ll} \sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j & \text{if } |i-j| = 1 \\ \sigma_i\sigma_j = \sigma_j\sigma_i & \text{if } |i-j| > 1 \end{array} \right\rangle.$$

The normal form is the Dehornoy form $w \to D(w)$ ("diffusion" in B_n).

The subgroups A, B in B_n are chosen randomly.

The private keys $a \in A, b \in B$ are chosen randomly.

AAG scheme

In the original AAG scheme G is the braid group B_n.

$$B_n = \left\langle \sigma_1, \ldots, \sigma_{n-1} \;\middle|\; \begin{array}{ll} \sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j & \text{if } |i-j| = 1 \\ \sigma_i\sigma_j = \sigma_j\sigma_i & \text{if } |i-j| > 1 \end{array} \right\rangle.$$

The normal form is the Dehornoy form $w \longrightarrow D(w)$ ("diffusion" in B_n).

The subgroups A, B in B_n are chosen randomly.

The private keys $a \in A, b \in B$ are chosen randomly.

AAG scheme

In the original AAG scheme G is the braid group B_n.

$$B_n = \left\langle \sigma_1, \ldots, \sigma_{n-1} \;\middle|\; \begin{array}{ll} \sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j & \text{if } |i-j| = 1 \\ \sigma_i\sigma_j = \sigma_j\sigma_i & \text{if } |i-j| > 1 \end{array} \right\rangle.$$

The normal form is the Dehornoy form $w \longrightarrow D(w)$ ("diffusion" in B_n).

The subgroups A, B in B_n are chosen randomly.

The private keys $a \in A, b \in B$ are chosen randomly.

AAG scheme

In the original AAG scheme G is the braid group B_n.

$$B_n = \left\langle \sigma_1, \ldots, \sigma_{n-1} \;\middle|\; \begin{array}{ll} \sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j & \text{if } |i-j| = 1 \\ \sigma_i\sigma_j = \sigma_j\sigma_i & \text{if } |i-j| > 1 \end{array} \right\rangle.$$

The normal form is the Dehornoy form $w \longrightarrow D(w)$ ("diffusion" in B_n).

The subgroups A, B in B_n are chosen randomly.

The private keys $a \in A, b \in B$ are chosen randomly.

Security assumptions

The **Algorithmic Security Assumption:** CKE problem is "hard" in G.

The CKE Problem:

Given the public information in the scheme CKE find the shared secret key $[a, b]$.

Complexity of CKE is unknown.

The danger of random subgroups: LBA in B_n

Almost a theorem [A. Miasnikov, A. Ushakov]

Let $\bar{a} = (a_1, \ldots, a_m)$ be "randomly chosen" elements in B_n. Then:

- The subgroup $A = \langle a_1, \ldots, a_m \rangle$ is a free group with basis $\bar{a}$;
- The subgroup A is quasi-isometrically embedded into B_n, so for every $w \in A$ the length $l_A(w)$ is about the same as $l_{B_n}(w)$.

Conjecture [Dynnikov]

The length of $D(w)$ is a "good approximation" of $l_{B_n}(w)$.

Heuristics [A. D. Miasnikov, A. Ushakov]

There is a good heuristic algorithm to compute $l_{B_n}(w)$.

The danger of random subgroups: LBA in B_n

Almost a theorem [A. Miasnikov, A. Ushakov]

Let $\bar{a} = (a_1, \ldots, a_m)$ be "randomly chosen" elements in B_n. Then:

- The subgroup $A = \langle a_1, \ldots, a_m \rangle$ is a free group with basis $\bar{a}$;
- The subgroup A is quasi-isometrically embedded into B_n, so for every $w \in A$ the length $l_A(w)$ is about the same as $l_{B_n}(w)$.

Conjecture [Dynnikov]

The length of $D(w)$ is a "good approximation" of $l_{B_n}(w)$.

Heuristics [A. D. Miasnikov, A. Ushakov]

There is a good heuristic algorithm to compute $l_{B_n}(w)$.

The danger of random subgroups: LBA in B_n

Almost a theorem [A. Miasnikov, A. Ushakov]

Let $\bar{a} = (a_1, \ldots, a_m)$ be "randomly chosen" elements in B_n. Then:

- The subgroup $A = \langle a_1, \ldots, a_m \rangle$ is a free group with basis $\bar{a}$;
- The subgroup A is quasi-isometrically embedded into B_n, so for every $w \in A$ the length $l_A(w)$ is about the same as $l_{B_n}(w)$.

Conjecture [Dynnikov]

The length of $D(w)$ is a "good approximation" of $l_{B_n}(w)$.

Heuristics [A. D. Miasnikov, A. Ushakov]

There is a good heuristic algorithm to compute $l_{B_n}(w)$.