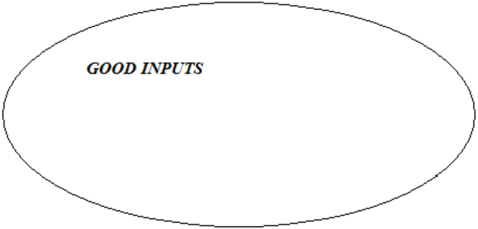
Amplification of algorithmic problems: cryptographic problems

Alexander Rybalov

Sobolev Institute of Mathematics, Omsk

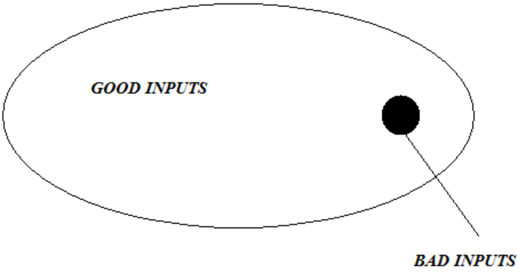
September, 2016

Classical approach



Algorithm works correctly on **all** inputs.

Generic approach



Algorithm works correctly on **almost all** inputs but can ignore some.

Asymptotic density

Definition

Let I be all inputs, I_n – all inputs of size n. **Asymptotic density** of set $S \subseteq I$

$$\mu(S) = \lim_{n \to \infty} \frac{|S \cap I_n|}{|I_n|}.$$

Asymptotic density

Definition

Let I be all inputs, I_n – all inputs of size n. **Asymptotic density** of set $S \subseteq I$

$$\mu(S) = \lim_{n \to \infty} \frac{|S \cap I_n|}{|I_n|}.$$

Remark

$\frac{|S \cap I_n|}{|I_n|}$ is the probability to get an input from S during random and uniform generation of inputs of size n.

Generic algorithms

Definition

Algorithm $\mathcal{A} : I \to J \cup \{?\}$ is called **generic**, if

1. for every $x \in I$ $\mathcal{A}(x)$ halts,
2. $\mu(\{x : \mathcal{A}(x) = ?\}) = 0$.

Generic algorithms

Definition

Algorithm $\mathcal{A} : I \to J \cup \{?\}$ is called **generic**, if

1. for every $x \in I$ $\mathcal{A}(x)$ halts,
2. $\mu(\{x : \mathcal{A}(x) = ?\}) = 0$.

Definition

Generic algorithm $\mathcal{A}$ computes a function $f : I \to J$, if for every $x \in I$

$$\mathcal{A}(x) \neq ? \Rightarrow f(x) = \mathcal{A}(x).$$

Cryptographic problems

Cryptographic problems

1. Discrete logarithm problem.

Cryptographic problems

1. Discrete logarithm problem.
2. Problem of computing of square roots in groups of residue classes.

Cryptographic problems

1. Discrete logarithm problem.
2. Problem of computing of square roots in groups of residue classes.
3. Searching graph isomorphism problem.

Cryptographic problems

1. Discrete logarithm problem.
2. Problem of computing of square roots in groups of residue classes.
3. Searching graph isomorphism problem.

Cryptographic hypothesis

For problems 1-3 there are no polynomial (probabilistic) algorithms, deciding their for all inputs.

Generic amplification

Theorem

If there exists a polynomial generic algorithm for a cryptographic problem, then there exists a polynomial probabilistic algorithm deciding it for all inputs.

Generic amplification

Theorem

If there exists a polynomial generic algorithm for a cryptographic problem, then there exists a polynomial probabilistic algorithm deciding it for all inputs.

Corollary

If a cryptographic problem is hard in worst case, then it hard in generic case.

Discrete logarithm problem

Problem

$I = \{(a, g, p),\ p$ – prime, g – primitive $GF(p)$, $a \in GF(p)^*\}$,
Compute the function $dl : I \to \mathbb{N}$, defined in the following way:

$$dl(a, g, p) = x \Leftrightarrow g^x = a \text{ в } GF(p).$$

Discrete logarithm problem

Problem

$I = \{(a, g, p),\ p$ – prime, g – primitive $GF(p),\ a \in GF(p)^*\}$,
Compute the function $dl : I \to \mathbb{N}$, defined in the following way:

$$dl(a, g, p) = x \Leftrightarrow g^x = a \text{ в } GF(p).$$

Applications in cryptography

Diffie-Hellman protocol, El-Gamal scheme, Massey-Omura cryptosystem.

Stratification

Definition

Sequence

$$\pi = \{(p_m, g_m),\ \ m \in \mathbb{N}, \text{where } p_m - \text{prime},$$

$$g_m - \text{primitive in } \ GF(p_m)\}$$

is called **exponential**, if $2^m < p_m < 2^{m+1}$ for all m.

Stratification

Definition

Sequence

$$\pi = \{(p_m, g_m),\ \ m \in \mathbb{N}, \text{where } p_m - \text{prime},$$

$$g_m - \text{primitive in } GF(p_m)\}$$

is called **exponential**, if $2^m < p_m < 2^{m+1}$ for all m.

Stratificated Discrete logarithm problem

$I_\pi = \{(a, g, p),\ (p, g) \in \pi,\ a \in GF(p)^*\}$,
Compute the function $dl_\pi = dl|_{I_\pi}$.

Stratification

Definition

Sequence

$$\pi = \{(p_m, g_m),\ m \in \mathbb{N}, \text{where } p_m - \text{prime},$$

$$g_m - \text{primitive in } GF(p_m)\}$$

is called **exponential**, if $2^m < p_m < 2^{m+1}$ for all m.

Stratificated Discrete logarithm problem

$I_\pi = \{(a, g, p),\ (p, g) \in \pi,\ a \in GF(p)^*\}$,
Compute the function $dl_\pi = dl|_{I_\pi}$.

Size of input (a, g, p) is the length of binary expansion of p.

Definition

Sequence

$$\pi = \{(p_m, g_m),\ m \in \mathbb{N}, \text{where } p_m - \text{prime},$$

$$g_m - \text{primitive in } GF(p_m)\}$$

is called **exponential**, if $2^m < p_m < 2^{m+1}$ for all m.

Stratificated Discrete logarithm problem

$I_\pi = \{(a, g, p),\ (p, g) \in \pi,\ a \in GF(p)^*\}$,
Compute the function $dl_\pi = dl|_{I_\pi}$.

Size of input (a, g, p) is the length of binary expansion of p. Input set of size n: all triples of type (a, g, p), where g, p are fixed, $a \in GF(p)^*$.

Discrete logarithm problem is generically hard

Theorem

If there is no polynomial probabilistic algorithm for dl, then there exists an exponential sequence π such that there is no probabilistic algorithm for dl_π.

Discrete logarithm problem is generically hard

Theorem

If there is no polynomial probabilistic algorithm for dl, then there exists an exponential sequence π such that there is no probabilistic algorithm for dl_π.

Theorem

If there is a polynomial generic algorithm for dl_π, then there is polynomial probabilistic algorithm computing dl_π for all inputs.

Amplification of Discrete logarithm

$(a, g, p) \to (ag^k, g, p)$, where k is randomly and uniformly generated number from 0 to $p - 2$.

Amplification of Discrete logarithm

$(a, g, p) \to (ag^k, g, p)$, where k is randomly and uniformly generated number from 0 to $p - 2$.
$ag^k = g^x \Rightarrow a = g^{x-k}$.

Problem of computing of square roots in groups of residue classes

Problem

$I = \{(a, m) : \ m = pq, p, q - \text{ primes}, \ a \in \mathbb{Z}/(m)\}$,
Compute the function $sr : I \to \mathbb{N}$, defined in the following way:

$$sr(a, m) = \begin{cases} x, & \text{if } x^2 = a \text{ в } \mathbb{Z}/(m), \\ 0, & \text{otherwise.} \end{cases}$$

Problem of computing of square roots in groups of residue classes

Problem

$I = \{(a, m) : \ m = pq, p, q -$ primes, $a \in \mathbb{Z}/(m)\}$,
Compute the function $sr : I \to \mathbb{N}$, defined in the following way:

$$sr(a, m) = \begin{cases} x, & \text{if } x^2 = a \text{ в } \mathbb{Z}/(m), \\ 0, & \text{otherwise.} \end{cases}$$

Applications in cryptography

Rabin cryptosystem.

Stratification

Definition

Sequence

$$\mu = \{m_k,\ k \in \mathbb{N}, \text{где } m_k - \text{произведение двух простых}\}$$

is called **exponential**, if $2^k < m_k < 2^{k+1}$ for all k.

Definition

Sequence

$$\mu = \{m_k,\ \ k \in \mathbb{N}, \text{где } m_k - \text{произведение двух простых}\}$$

is called **exponential**, if $2^k < m_k < 2^{k+1}$ for all k.

Stratificated problem

$I_\mu = \{(a, m) :\ \ m \in \mu,\ \ a \in \mathbb{Z}/(m)\}$,
Compute the function $sr_\mu = sr|_{I_\mu}$.

Stratification

Definition

Sequence

$$\mu = \{m_k,\ \ k \in \mathbb{N}, \text{где } m_k - \text{произведение двух простых}\}$$

is called **exponential**, if $2^k < m_k < 2^{k+1}$ for all k.

Stratificated problem

$I_\mu = \{(a, m) :\ \ m \in \mu,\ \ a \in \mathbb{Z}/(m)\}$,
Compute the function $sr_\mu = sr|_{I_\mu}$.

Size of input (a, m) is the length of binary expansion of m.

Stratification

Definition

Sequence

$$\mu = \{m_k,\ k \in \mathbb{N}, \text{где } m_k - \text{произведение двух простых}\}$$

is called **exponential**, if $2^k < m_k < 2^{k+1}$ for all k.

Stratificated problem

$I_\mu = \{(a, m) :\ m \in \mu,\ a \in \mathbb{Z}/(m)\}$,
Compute the function $sr_\mu = sr|_{I_\mu}$.

Size of input (a, m) is the length of binary expansion of m. Set of inputs of size n: all pairs of type (a, m), where m is fixed, $a \in \mathbb{Z}/(m)$.

Square root problem is generically hard

Theorem

If there is no polynomial probabilistic algorithm for sr, then there exists an exponential sequence μ such that there is no probabilistic algorithm for sr_μ.

Square root problem is generically hard

Theorem

If there is no polynomial probabilistic algorithm for sr, then there exists an exponential sequence μ such that there is no probabilistic algorithm for sr_{μ}.

Theorem

If there is a polynomial generic algorithm for sr_{μ}, then there is polynomial probabilistic algorithm computing sr_{μ} for all inputs.

Amplification of the square root problem

$(a, m) \to (ab^2, m)$, where b is randomly and uniformly generated element from $\mathbb{Z}/(m)$.

Amplification of the square root problem

$(a, m) \to (ab^2, m)$, where b is randomly and uniformly generated element from $\mathbb{Z}/(m)$.
$ab^2 = x^2 \Rightarrow a = (xb^{-1})^2$.

Searching graph isomorphism problem

Problem

$I = \{(G_1, G_2),\ G_1, G_2\ -\ \text{isomorphic graphs}\}$. Compute the function $sgi : I \to S_1 \cup S_2 \cup \ldots S_n \cup \ldots$, defined in the following way:

$$sgi(G_1, G_2) = \text{permutation of vertices – isomorphism } G_1 \text{ and } G_2.$$

Searching graph isomorphism problem

Problem

$I = \{(G_1, G_2),\ G_1, G_2\ -\ \text{isomorphic graphs}\}$. Compute the function $sgi : I \rightarrow S_1 \cup S_2 \cup \ldots S_n \cup \ldots$, defined in the following way:

$$sgi(G_1, G_2) = \text{permutation of vertices – isomorphism } G_1 \text{ and } G_2.$$

Applications in cryptography

Zero-knowledge proof (Shafi Goldwasser, Silvio Micali, and Charles Rackoff).

Denotation

Sequence of graphs

$$\gamma = \{G_n,\ n \in \mathbb{N}, \text{where } G_n - \text{graph with } n \text{ vertices}\}.$$

Denotation

Sequence of graphs

$$\gamma = \{G_n,\ n \in \mathbb{N}, \text{where } G_n - \text{graph with } n \text{ vertices}\}.$$

Problem

$I_\gamma = \{(G_1, G_2) :\ G_2 \in \gamma\}$. Compute the function $sgi_\gamma = sgi|_{I_\gamma}$.

Denotation

Sequence of graphs

$$\gamma = \{G_n,\ n \in \mathbb{N}, \text{where } G_n - \text{graph with } n \text{ vertices}\}.$$

Problem

$I_\gamma = \{(G_1, G_2) :\ G_2 \in \gamma\}$. Compute the function $sgi_\gamma = sgi|_{I_\gamma}$.

Size of input (G_1, G_2) is number of vertices of G_2.

Denotation

Sequence of graphs

$$\gamma = \{G_n,\ n \in \mathbb{N}, \text{where } G_n - \text{graph with } n \text{ vertices}\}.$$

Problem

$I_\gamma = \{(G_1, G_2) :\ G_2 \in \gamma\}$. Compute the function $sgi_\gamma = sgi|_{I_\gamma}$.

Size of input (G_1, G_2) is number of vertices of G_2. Set of inputs of size n: all pairs of type (G_1, G_2), where G_2 is fixed, G_1 is graph isomorphic to G_2.

Searching graph isomorphism problem is generically hard

Theorem

If there is no polynomial probabilistic algorithm for *sgi*, then there exists a graph sequence γ such that there is no probabilistic algorithm for sgi_{γ}.

Theorem

If there is no polynomial probabilistic algorithm for *sgi*, then there exists a graph sequence γ such that there is no probabilistic algorithm for sgi_γ.

Theorem

If there is a polynomial generic algorithm for sgi_γ, then there is polynomial probabilistic algorithm computing sgi_γ for all inputs.

Amplification of the Searching graph isomorphism problem

$(G_1, G_2) \to (\pi(G_1), G_2)$, where π is randomly and uniformly generated permutation from S_n.

$(G_1, G_2) \to (\pi(G_1), G_2)$, where π is randomly and uniformly generated permutation from S_n.
$\tau(\pi(G_1)) = G_2 \Rightarrow \chi = \tau\pi$.