Public-key Cryptography
Theory and Practice

Abhijit Das

Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur

Chapter 4: The Intractable Mathematical Problems

The Intractable Problems

The Intractable Problems

- Public-key cryptography is based on **trapdoor one-way functions**. It should be easy to encrypt a message or verify a signature, but inverting the transform (decryption or signature generation) should be difficult, unless some secret information (the trapdoor) is known.

The Intractable Problems

- Public-key cryptography is based on **trapdoor one-way functions**. It should be easy to encrypt a message or verify a signature, but inverting the transform (decryption or signature generation) should be difficult, unless some secret information (the trapdoor) is known.
- Some difficult computational problems

The Intractable Problems

- Public-key cryptography is based on **trapdoor one-way functions**. It should be easy to encrypt a message or verify a signature, but inverting the transform (decryption or signature generation) should be difficult, unless some secret information (the trapdoor) is known.
- Some difficult computational problems
  - Factoring composite integers

The Intractable Problems

- Public-key cryptography is based on **trapdoor one-way functions**. It should be easy to encrypt a message or verify a signature, but inverting the transform (decryption or signature generation) should be difficult, unless some secret information (the trapdoor) is known.
- Some difficult computational problems
  - Factoring composite integers
  - Computing square roots modulo a composite integer

The Intractable Problems

- Public-key cryptography is based on **trapdoor one-way functions**. It should be easy to encrypt a message or verify a signature, but inverting the transform (decryption or signature generation) should be difficult, unless some secret information (the trapdoor) is known.
- Some difficult computational problems
  - Factoring composite integers
  - Computing square roots modulo a composite integer
  - Computing discrete logarithms in certain groups (finite fields, elliptic hyperelliptic curves, class groups of number fields, and so on)

The Intractable Problems

- Public-key cryptography is based on **trapdoor one-way functions**. It should be easy to encrypt a message or verify a signature, but inverting the transform (decryption or signature generation) should be difficult, unless some secret information (the trapdoor) is known.
- Some difficult computational problems
  - Factoring composite integers
  - Computing square roots modulo a composite integer
  - Computing discrete logarithms in certain groups (finite fields, elliptic hyperelliptic curves, class groups of number fields, and so on)
  - Finding shortest/closest vectors in a lattice

The Intractable Problems

- Public-key cryptography is based on **trapdoor one-way functions.** It should be easy to encrypt a message or verify a signature, but inverting the transform (decryption or signature generation) should be difficult, unless some secret information (the trapdoor) is known.
- Some difficult computational problems
  - Factoring composite integers
  - Computing square roots modulo a composite integer
  - Computing discrete logarithms in certain groups (finite fields, elliptic hyperelliptic curves, class groups of number fields, and so on)
  - Finding shortest/closest vectors in a lattice
  - Solving the subset sum problem

The Intractable Problems

- Public-key cryptography is based on **trapdoor one-way functions**. It should be easy to encrypt a message or verify a signature, but inverting the transform (decryption or signature generation) should be difficult, unless some secret information (the trapdoor) is known.
- Some difficult computational problems
  - Factoring composite integers
  - Computing square roots modulo a composite integer
  - Computing discrete logarithms in certain groups (finite fields, elliptic hyperelliptic curves, class groups of number fields, and so on)
  - Finding shortest/closest vectors in a lattice
  - Solving the subset sum problem
  - Finding roots of non-linear multivariate polynomials

The Intractable Problems

- Public-key cryptography is based on **trapdoor one-way functions**. It should be easy to encrypt a message or verify a signature, but inverting the transform (decryption or signature generation) should be difficult, unless some secret information (the trapdoor) is known.
- Some difficult computational problems
  - Factoring composite integers
  - Computing square roots modulo a composite integer
  - Computing discrete logarithms in certain groups (finite fields, elliptic hyperelliptic curves, class groups of number fields, and so on)
  - Finding shortest/closest vectors in a lattice
  - Solving the subset sum problem
  - Finding roots of non-linear multivariate polynomials
  - Solving the braid conjugacy problem

The Intractable Problems (contd)

The Intractable Problems (contd)

- Many sophisticated algorithms are proposed to break the trapdoor functions. Most of these are fully exponential. **Subexponential algorithms** are sometimes known.

The Intractable Problems (contd)

- Many sophisticated algorithms are proposed to break the trapdoor functions. Most of these are fully exponential. **Subexponential algorithms** are sometimes known.
- For suitably chosen domain parameters, these algorithms take infeasible time.

The Intractable Problems (contd)

- Many sophisticated algorithms are proposed to break the trapdoor functions. Most of these are fully exponential. **Subexponential algorithms** are sometimes known.
- For suitably chosen domain parameters, these algorithms take infeasible time.
- No non-trivial lower bounds on the complexity of these computational problems are known. Even existence of polynomial-time algorithms cannot be often ruled out.

The Intractable Problems (contd)

- Many sophisticated algorithms are proposed to break the trapdoor functions. Most of these are fully exponential. **Subexponential algorithms** are sometimes known.
- For suitably chosen domain parameters, these algorithms take infeasible time.
- No non-trivial lower bounds on the complexity of these computational problems are known. Even existence of polynomial-time algorithms cannot be often ruled out.
- Certain special cases have been discovered to be cryptographically weak. For practical designs, it is essential to avoid these special cases.

The Intractable Problems (contd)

- Many sophisticated algorithms are proposed to break the trapdoor functions. Most of these are fully exponential. **Subexponential algorithms** are sometimes known.
- For suitably chosen domain parameters, these algorithms take infeasible time.
- No non-trivial lower bounds on the complexity of these computational problems are known. Even existence of polynomial-time algorithms cannot be often ruled out.
- Certain special cases have been discovered to be cryptographically weak. For practical designs, it is essential to avoid these special cases.
- Polynomial-time quantum algorithms are known for factoring integers and computing discrete logarithms in finite fields.

Discrete Logarithms

Discrete Logarithms

- Let $\mathbb{F}_q$ be a finite field, g a generator of $\mathbb{F}_q^*$, and $a \in \mathbb{F}_q^*$. There exists a unique integer $x \in \{0, 1, 2, \ldots, q-1\}$ such that $a = g^x$. We call x the *index* or *discrete logarithm* of a to the base g. We denote this by $x = \operatorname{ind}_g a$.

Discrete Logarithms

- Let $\mathbb{F}_q$ be a finite field, g a generator of $\mathbb{F}_q^*$, and $a \in \mathbb{F}_q^*$. There exists a unique integer $x \in \{0, 1, 2, \ldots, q-1\}$ such that $a = g^x$. We call x the *index* or *discrete logarithm* of a to the base g. We denote this by $x = \operatorname{ind}_g a$.
- Indices follow arithmetic modulo $q - 1$.

$$\begin{aligned} \operatorname{ind}_g(ab) &\equiv \operatorname{ind}_g a + \operatorname{ind}_g b \pmod{q-1}, \\ \operatorname{ind}_g(a^e) &\equiv e \operatorname{ind}_g a \pmod{q-1}. \end{aligned}$$

Discrete Logarithms

- Let $\mathbb{F}_q$ be a finite field, g a generator of $\mathbb{F}_q^*$, and $a \in \mathbb{F}_q^*$. There exists a unique integer $x \in \{0, 1, 2, \ldots, q-1\}$ such that $a = g^x$. We call x the *index* or *discrete logarithm* of a to the base g. We denote this by $x = \operatorname{ind}_g a$.
- Indices follow arithmetic modulo $q - 1$.

$$\begin{aligned} \operatorname{ind}_g(ab) &\equiv \operatorname{ind}_g a + \operatorname{ind}_g b \pmod{q-1}, \\ \operatorname{ind}_g(a^e) &\equiv e \operatorname{ind}_g a \pmod{q-1}. \end{aligned}$$

- The concept of discrete logarithms can be extended to other finite groups (including the elliptic curve group).

Discrete Logarithm: Example

Discrete Logarithm: Example

- Take $p = 17$ and $g = 3$.

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\operatorname{ind}_3 a$	0	14	1	12	5	15	11	10	2	3	7	13	4	9	6	8

Discrete Logarithm: Example

- Take $p = 17$ and $g = 3$.

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\operatorname{ind}_3 a$	0	14	1	12	5	15	11	10	2	3	7	13	4	9	6	8

- $\operatorname{ind}_3 6 = 15$ and $\operatorname{ind}_3 11 = 7$. Since $6 \times 11 = 15 \pmod{17}$, we have $\operatorname{ind}_3 15 \equiv \operatorname{ind}_3 6 + \operatorname{ind}_3 11 \equiv 15 + 7 \equiv 6 \pmod{16}$.

The Most Common Intractable Problems

The Most Common Intractable Problems

Integer factorization problem (IFP): Given $n \in \mathbb{N}$, compute the complete prime factorization of n. Suppose there is an algorithm A that computes a non-trivial factor of n. We can use A repeatedly in order to compute the complete factorization of n. If $n = pq$ (with $p, q \in \mathbb{P}$), then computing p or q suffices.

The Most Common Intractable Problems

Integer factorization problem (IFP): Given $n \in \mathbb{N}$, compute the complete prime factorization of n. Suppose there is an algorithm A that computes a non-trivial factor of n. We can use A repeatedly in order to compute the complete factorization of n. If $n = pq$ (with $p, q \in \mathbb{P}$), then computing p or q suffices.

Example

Input: $n = 85067$.
Output: $85067 = 257 \times 331$.

The Most Common Intractable Problems

Integer factorization problem (IFP): Given $n \in \mathbb{N}$, compute the complete prime factorization of n. Suppose there is an algorithm A that computes a non-trivial factor of n. We can use A repeatedly in order to compute the complete factorization of n. If $n = pq$ (with $p, q \in \mathbb{P}$), then computing p or q suffices.

Example

Input: $n = 85067$.
Output: $85067 = 257 \times 331$.

Discrete logarithm problem (DLP): Let g be a generator of $\mathbb{F}_q^*$. Given $a \in \mathbb{F}_q^*$, compute $\mathrm{ind}_g\, a$.

The Most Common Intractable Problems

Integer factorization problem (IFP): Given $n \in \mathbb{N}$, compute the complete prime factorization of n. Suppose there is an algorithm A that computes a non-trivial factor of n. We can use A repeatedly in order to compute the complete factorization of n. If $n = pq$ (with $p, q \in \mathbb{P}$), then computing p or q suffices.

Example

Input: $n = 85067$.

Output: $85067 = 257 \times 331$.

Discrete logarithm problem (DLP): Let g be a generator of $\mathbb{F}_q^*$. Given $a \in \mathbb{F}_q^*$, compute $\mathrm{ind}_g\, a$.

Example

Input: $p = 17$, $g = 3$, $a = 11$.

Output: $\mathrm{ind}_g\, a = 7$.

The Most Common Intractable Problems (contd)

The Most Common Intractable Problems (contd)

- IFP and DLP are believed to be computationally difficult.

The Most Common Intractable Problems (contd)

- IFP and DLP are believed to be computationally difficult.
- The best known algorithms for IFP and DLP are subexponential.

The Most Common Intractable Problems (contd)

- IFP and DLP are believed to be computationally difficult.
- The best known algorithms for IFP and DLP are subexponential.
- IFP is the inverse of the integer multiplication problem.

The Most Common Intractable Problems (contd)

- IFP and DLP are believed to be computationally difficult.
- The best known algorithms for IFP and DLP are subexponential.
- IFP is the inverse of the integer multiplication problem.
- DLP is the inverse of the modular exponentiation problem.

The Most Common Intractable Problems (contd)

- IFP and DLP are believed to be computationally difficult.
- The best known algorithms for IFP and DLP are subexponential.
- IFP is the inverse of the integer multiplication problem.
- DLP is the inverse of the modular exponentiation problem.
- Integer multiplication and modular exponentiation are easy computational problems. They are believed to be one-way functions.

The Most Common Intractable Problems (contd)

- IFP and DLP are believed to be computationally difficult.
- The best known algorithms for IFP and DLP are subexponential.
- IFP is the inverse of the integer multiplication problem.
- DLP is the inverse of the modular exponentiation problem.
- Integer multiplication and modular exponentiation are easy computational problems. They are believed to be one-way functions.
- There is, however, no proof that IFP and DLP must be difficult.

The Most Common Intractable Problems (contd)

- IFP and DLP are believed to be computationally difficult.
- The best known algorithms for IFP and DLP are subexponential.
- IFP is the inverse of the integer multiplication problem.
- DLP is the inverse of the modular exponentiation problem.
- Integer multiplication and modular exponentiation are easy computational problems. They are believed to be one-way functions.
- There is, however, no proof that IFP and DLP must be difficult.
- Efficient quantum algorithms exist for solving IFP and DLP.

The Most Common Intractable Problems (contd)

- IFP and DLP are believed to be computationally difficult.
- The best known algorithms for IFP and DLP are subexponential.
- IFP is the inverse of the integer multiplication problem.
- DLP is the inverse of the modular exponentiation problem.
- Integer multiplication and modular exponentiation are easy computational problems. They are believed to be one-way functions.
- There is, however, no proof that IFP and DLP must be difficult.
- Efficient quantum algorithms exist for solving IFP and DLP.
- IFP and DLP are believed to be computationally equivalent.

Intractable Problems: Variants

Intractable Problems: Variants

- **Diffie-Hellman problem (DHP):** Let g be a generator of $\mathbb{F}_q^*$. Given the elements g^x and g^y of $\mathbb{F}_q^*$, compute g^{xy}.

Intractable Problems: Variants

- **Diffie-Hellman problem (DHP):** Let g be a generator of $\mathbb{F}_q^*$. Given the elements g^x and g^y of $\mathbb{F}_q^*$, compute g^{xy}.
- **Example**

Intractable Problems: Variants

- **Diffie-Hellman problem (DHP):** Let g be a generator of $\mathbb{F}_q^*$. Given the elements g^x and g^y of $\mathbb{F}_q^*$, compute g^{xy}.
- **Example**
  - Input: $p = 17$, $g = 3$, $g^x \equiv 11 \pmod{p}$ and $g^y \equiv 13 \pmod{p}$.

Intractable Problems: Variants

- **Diffie-Hellman problem (DHP):** Let g be a generator of $\mathbb{F}_q^*$. Given the elements g^x and g^y of $\mathbb{F}_q^*$, compute g^{xy}.
- **Example**
  - Input: $p = 17$, $g = 3$, $g^x \equiv 11 \pmod{p}$ and $g^y \equiv 13 \pmod{p}$.
  - Output: $g^{xy} \equiv 4 \pmod{p}$.

Intractable Problems: Variants

- **Diffie-Hellman problem (DHP):** Let g be a generator of $\mathbb{F}_q^*$. Given the elements g^x and g^y of $\mathbb{F}_q^*$, compute g^{xy}.
- **Example**
  - Input: $p = 17$, $g = 3$, $g^x \equiv 11 \pmod{p}$ and $g^y \equiv 13 \pmod{p}$.
  - Output: $g^{xy} \equiv 4 \pmod{p}$.
  - ($x = 7$, $y = 4$, that is, $xy \equiv 28 \equiv 12 \pmod{p-1}$, that is, $g^{xy} \equiv 3^{12} \equiv 4 \pmod{p}$.)

Intractable Problems: Variants

- **Diffie-Hellman problem (DHP):** Let g be a generator of $\mathbb{F}_q^*$. Given the elements g^x and g^y of $\mathbb{F}_q^*$, compute g^{xy}.
- **Example**
  - Input: $p = 17$, $g = 3$, $g^x \equiv 11 \pmod{p}$ and $g^y \equiv 13 \pmod{p}$.
  - Output: $g^{xy} \equiv 4 \pmod{p}$.
  - ($x = 7$, $y = 4$, that is, $xy \equiv 28 \equiv 12 \pmod{p-1}$, that is, $g^{xy} \equiv 3^{12} \equiv 4 \pmod{p}$.)
- DHP is another believably difficult computational problem.

Intractable Problems: Variants

- **Diffie-Hellman problem (DHP):** Let g be a generator of $\mathbb{F}_q^*$. Given the elements g^x and g^y of $\mathbb{F}_q^*$, compute g^{xy}.
- **Example**
  - Input: $p = 17$, $g = 3$, $g^x \equiv 11 \pmod{p}$ and $g^y \equiv 13 \pmod{p}$.
  - Output: $g^{xy} \equiv 4 \pmod{p}$.
  - ($x = 7$, $y = 4$, that is, $xy \equiv 28 \equiv 12 \pmod{p-1}$, that is, $g^{xy} \equiv 3^{12} \equiv 4 \pmod{p}$.)
- DHP is another believably difficult computational problem.
- If DLP can be solved, then DHP can be solved ($g^{xy} = (g^x)^y$).

Intractable Problems: Variants

- **Diffie-Hellman problem (DHP):** Let g be a generator of $\mathbb{F}_q^*$. Given the elements g^x and g^y of $\mathbb{F}_q^*$, compute g^{xy}.
- **Example**
  - Input: $p = 17$, $g = 3$, $g^x \equiv 11 \pmod{p}$ and $g^y \equiv 13 \pmod{p}$.
  - Output: $g^{xy} \equiv 4 \pmod{p}$.
  - ($x = 7$, $y = 4$, that is, $xy \equiv 28 \equiv 12 \pmod{p-1}$, that is, $g^{xy} \equiv 3^{12} \equiv 4 \pmod{p}$.)
- DHP is another believably difficult computational problem.
- If DLP can be solved, then DHP can be solved ($g^{xy} = (g^x)^y$).
- The converse is only believed to be true.

Intractable Problems: Variants

Intractable Problems: Variants

- **Elliptic Curve Discrete Logarithm Problem (ECDLP)**
  Let E be an elliptic curve defined over the finite field $\mathbb{F}_q$.
  Given points P and xP in $E(\mathbb{F}_q)$, compute x.

Intractable Problems: Variants

- **Elliptic Curve Discrete Logarithm Problem (ECDLP)**
  Let E be an elliptic curve defined over the finite field $\mathbb{F}_q$.
  Given points P and xP in $E(\mathbb{F}_q)$, compute x.

- **Elliptic Curve Diffie-Hellman Problem (ECDHP)**
  Let E be an elliptic curve defined over the finite field $\mathbb{F}_q$.
  Given points xP and yP in $E(\mathbb{F}_q)$, compute the point xyP.

Intractable Problems: Variants

- **Elliptic Curve Discrete Logarithm Problem (ECDLP)**
  Let E be an elliptic curve defined over the finite field $\mathbb{F}_q$.
  Given points P and xP in $E(\mathbb{F}_q)$, compute x.
- **Elliptic Curve Diffie-Hellman Problem (ECDHP)**
  Let E be an elliptic curve defined over the finite field $\mathbb{F}_q$.
  Given points xP and yP in $E(\mathbb{F}_q)$, compute the point xyP.
- **Example**

Intractable Problems: Variants

- **Elliptic Curve Discrete Logarithm Problem (ECDLP)**
  Let E be an elliptic curve defined over the finite field $\mathbb{F}_q$. Given points P and xP in $E(\mathbb{F}_q)$, compute x.

- **Elliptic Curve Diffie-Hellman Problem (ECDHP)**
  Let E be an elliptic curve defined over the finite field $\mathbb{F}_q$. Given points xP and yP in $E(\mathbb{F}_q)$, compute the point xyP.

- **Example**
  - Consider the curve $E : y^2 = x^3 + x + 3$ defined over $\mathbb{F}_7$.

Intractable Problems: Variants

- **Elliptic Curve Discrete Logarithm Problem (ECDLP)**
  Let E be an elliptic curve defined over the finite field $\mathbb{F}_q$.
  Given points P and xP in $E(\mathbb{F}_q)$, compute x.

- **Elliptic Curve Diffie-Hellman Problem (ECDHP)**
  Let E be an elliptic curve defined over the finite field $\mathbb{F}_q$.
  Given points xP and yP in $E(\mathbb{F}_q)$, compute the point xyP.

- **Example**
  - Consider the curve $E : y^2 = x^3 + x + 3$ defined over $\mathbb{F}_7$.
  - $E(\mathbb{F}_7)$ Is cyclic of order 6.

Intractable Problems: Variants

- **Elliptic Curve Discrete Logarithm Problem (ECDLP)**
  Let E be an elliptic curve defined over the finite field $\mathbb{F}_q$.
  Given points P and xP in $E(\mathbb{F}_q)$, compute x.
- **Elliptic Curve Diffie-Hellman Problem (ECDHP)**
  Let E be an elliptic curve defined over the finite field $\mathbb{F}_q$.
  Given points xP and yP in $E(\mathbb{F}_q)$, compute the point xyP.
- **Example**
  - Consider the curve $E : y^2 = x^3 + x + 3$ defined over $\mathbb{F}_7$.
  - $E(\mathbb{F}_7)$ Is cyclic of order 6.
  - $P = (4, 1)$ is a generator of $E(\mathbb{F}_7)$.

Intractable Problems: Variants

- **Elliptic Curve Discrete Logarithm Problem (ECDLP)**
  Let E be an elliptic curve defined over the finite field $\mathbb{F}_q$. Given points P and xP in $E(\mathbb{F}_q)$, compute x.
- **Elliptic Curve Diffie-Hellman Problem (ECDHP)**
  Let E be an elliptic curve defined over the finite field $\mathbb{F}_q$. Given points xP and yP in $E(\mathbb{F}_q)$, compute the point xyP.
- **Example**
  - Consider the curve $E : y^2 = x^3 + x + 3$ defined over $\mathbb{F}_7$.
  - $E(\mathbb{F}_7)$ Is cyclic of order 6.
  - $P = (4, 1)$ is a generator of $E(\mathbb{F}_7)$.
  - The index of $Q = (5, 0)$ to the base P is 3, that is, $Q = 3P$.

Intractable Problems: Variants

- **Elliptic Curve Discrete Logarithm Problem (ECDLP)**
  Let E be an elliptic curve defined over the finite field $\mathbb{F}_q$. Given points P and xP in $E(\mathbb{F}_q)$, compute x.
- **Elliptic Curve Diffie-Hellman Problem (ECDHP)**
  Let E be an elliptic curve defined over the finite field $\mathbb{F}_q$. Given points xP and yP in $E(\mathbb{F}_q)$, compute the point xyP.
- **Example**
  - Consider the curve $E : y^2 = x^3 + x + 3$ defined over $\mathbb{F}_7$.
  - $E(\mathbb{F}_7)$ Is cyclic of order 6.
  - $P = (4, 1)$ is a generator of $E(\mathbb{F}_7)$.
  - The index of $Q = (5, 0)$ to the base P is 3, that is, $Q = 3P$.
  - Let $Q_1 = 3P = (5, 0)$ and $Q_2 = 4P = (6, 1)$. Then, $(3 \times 4)P = 12P = 0P = \mathcal{O}$.

Intractable Problems: Variants

Intractable Problems: Variants

- All finite (Abelian) groups are not cyclic.

Intractable Problems: Variants

- All finite (Abelian) groups are not cyclic.
- Although all $\mathbb{F}_q^*$ are cyclic, all elliptic curve groups $E(\mathbb{F}_q)$ are not cyclic.

Intractable Problems: Variants

- All finite (Abelian) groups are not cyclic.
- Although all $\mathbb{F}_q^*$ are cyclic, all elliptic curve groups $E(\mathbb{F}_q)$ are not cyclic.
- Even if G is a cyclic group, a generator of G may be unknown.

Intractable Problems: Variants

- All finite (Abelian) groups are not cyclic.
- Although all $\mathbb{F}_q^*$ are cyclic, all elliptic curve groups $E(\mathbb{F}_q)$ are not cyclic.
- Even if G is a cyclic group, a generator of G may be unknown.
- Computing a generator of $\mathbb{F}_q^*$ requires the complete factorization of $q - 1$.

Intractable Problems: Variants

- All finite (Abelian) groups are not cyclic.
- Although all $\mathbb{F}_q^*$ are cyclic, all elliptic curve groups $E(\mathbb{F}_q)$ are not cyclic.
- Even if G is a cyclic group, a generator of G may be unknown.
- Computing a generator of $\mathbb{F}_q^*$ requires the complete factorization of $q - 1$.
- **Generalized Discrete Logarithm Problem (GDLP)**
  Let G be a (multiplicative) Abelian group of size n and let g be an element of G of order m (we have $m \mid n$). Let H be the subgroup of G generated by g. Given $a \in G$, determine whether $a \in H$, and if so, determine the unique integer $x \in \{0, 1, 2, \ldots, m - 1\}$ such that $a = g^x$.

GDLP: Example

GDLP: Example

- **Example**

GDLP: Example

- **Example**
  - Take $p = 661$ and $g = 29$. We have $\mathrm{ord}_p(g) = 66$.

GDLP: Example

- **Example**
  - Take $p = 661$ and $g = 29$. We have $\operatorname{ord}_p(g) = 66$.
  - We have $g^{15} \equiv 49 \pmod{p}$, that is, $\operatorname{ind}_g(49) = 15$.

GDLP: Example

- **Example**
  - Take $p = 661$ and $g = 29$. We have $\operatorname{ord}_p(g) = 66$.
  - We have $g^{15} \equiv 49 \pmod{p}$, that is, $\operatorname{ind}_g(49) = 15$.
  - $\operatorname{ind}_g(94)$ does not exist.

GDLP: Example

- **Example**
  - Take $p = 661$ and $g = 29$. We have $\operatorname{ord}_p(g) = 66$.
  - We have $g^{15} \equiv 49 \pmod{p}$, that is, $\operatorname{ind}_g(49) = 15$.
  - $\operatorname{ind}_g(94)$ does not exist.
- If G is cyclic, then $a \in H$ if and only if $a^m = e$.

GDLP: Example

- **Example**
  - Take $p = 661$ and $g = 29$. We have $\operatorname{ord}_p(g) = 66$.
  - We have $g^{15} \equiv 49 \pmod{p}$, that is, $\operatorname{ind}_g(49) = 15$.
  - $\operatorname{ind}_g(94)$ does not exist.
- If G is cyclic, then $a \in H$ if and only if $a^m = e$.
- **Example**

GDLP: Example

- **Example**
  - Take $p = 661$ and $g = 29$. We have $\mathrm{ord}_p(g) = 66$.
  - We have $g^{15} \equiv 49 \pmod{p}$, that is, $\mathrm{ind}_g(49) = 15$.
  - $\mathrm{ind}_g(94)$ does not exist.
- If G is cyclic, then $a \in H$ if and only if $a^m = e$.
- **Example**
  - $49^{66} \equiv 1 \pmod{661}$.

GDLP: Example

- **Example**
  - Take $p = 661$ and $g = 29$. We have $\operatorname{ord}_p(g) = 66$.
  - We have $g^{15} \equiv 49 \pmod{p}$, that is, $\operatorname{ind}_g(49) = 15$.
  - $\operatorname{ind}_g(94)$ does not exist.
- If G is cyclic, then $a \in H$ if and only if $a^m = e$.
- **Example**
  - $49^{66} \equiv 1 \pmod{661}$.
  - $94^{66} \equiv -1 \pmod{661}$.

The Integer Factorization Problem (IFP)

The Integer Factorization Problem (IFP)

Let n be the integer to be factored.

The Integer Factorization Problem (IFP)

Let n be the integer to be factored.

Older algorithms

- Trial division (efficient if all prime divisors of n are small)
- Pollard's rho method
- Pollard's $p - 1$ method (efficient if $p - 1$ has only small prime factors for some prime divisor p of n)
- Williams' $p + 1$ method (efficient if $p + 1$ has only small prime factors for some prime divisor p of n)

The Integer Factorization Problem (IFP)

Let n be the integer to be factored.

Older algorithms

- Trial division (efficient if all prime divisors of n are small)
- Pollard's rho method
- Pollard's $p - 1$ method (efficient if $p - 1$ has only small prime factors for some prime divisor p of n)
- Williams' $p + 1$ method (efficient if $p + 1$ has only small prime factors for some prime divisor p of n)

In the worst case, these algorithms take exponential (in $\log n$) running time.

Modern Factoring Algorithms

Modern Factoring Algorithms

Subexponential running time:

$$L(n, \omega, c) = \exp\left[(c + \mathrm{o}(1))(\ln n)^{\omega}(\ln \ln n)^{1-\omega}\right]$$

Modern Factoring Algorithms

Subexponential running time:

$$L(n, \omega, c) = \exp\left[(c + \mathrm{o}(1))(\ln n)^{\omega}(\ln \ln n)^{1-\omega}\right]$$

$\omega = 0$: $L(n, \omega, c)$ is polynomial in $\ln n$.

Modern Factoring Algorithms

Subexponential running time:

$$L(n, \omega, c) = \exp\left[(c + \mathrm{o}(1))(\ln n)^{\omega}(\ln \ln n)^{1-\omega}\right]$$

$\omega = 0$: $L(n, \omega, c)$ is polynomial in $\ln n$.

$\omega = 1$: $L(n, \omega, c)$ is exponential in $\ln n$.

Modern Factoring Algorithms

Subexponential running time:

$L(n, \omega, c) = \exp\left[(c + \mathrm{o}(1))(\ln n)^{\omega}(\ln \ln n)^{1-\omega}\right]$

$\omega = 0$: $L(n, \omega, c)$ is polynomial in $\ln n$.
$\omega = 1$: $L(n, \omega, c)$ is exponential in $\ln n$.
$0 < \omega < 1$: $L(n, \omega, c)$ is between polynomial and exponential

Modern Factoring Algorithms

Subexponential running time:

$$L(n, \omega, c) = \exp\left[(c + \mathrm{o}(1))(\ln n)^{\omega}(\ln\ln n)^{1-\omega}\right]$$

$\omega = 0$: $L(n, \omega, c)$ is polynomial in $\ln n$.

$\omega = 1$: $L(n, \omega, c)$ is exponential in $\ln n$.

$0 < \omega < 1$: $L(n, \omega, c)$ is between polynomial and exponential

Algorithm	Inventor(s)	Running time
Continued fraction method (CFRAC)	Morrison & Brillhart (1975)	$L(n, 1/2, c)$
Quadratic sieve method (QSM)	Pomerance (1984)	$L(n, 1/2, 1)$
Cubic sieve method (CSM)	Reyneri	$L(n, 1/2, 0.816)$
Elliptic curve method (ECM)	H. W. Lenstra (1987)	$L(n, 1/2, c)$
Number field sieve method (NFSM)	A. K. Lenstra, H. W. Lenstra, Manasse & Pollard (1990)	$L(n, 1/3, 1.923)$

Trial Division

Trial Division

- Divide n by $2, 3, 4, 5, \ldots, \lfloor \sqrt{n} \rfloor$.

Trial Division

- Divide n by $2, 3, 4, 5, \ldots, \lfloor \sqrt{n} \rfloor$.
- It suffices to divide only by primes in the above range.

Trial Division

- Divide n by $2, 3, 4, 5, \ldots, \lfloor \sqrt{n} \rfloor$.
- It suffices to divide only by primes in the above range.
- A list of primes may be unavailable.

Trial Division

- Divide n by $2, 3, 4, 5, \ldots, \lfloor \sqrt{n} \rfloor$.
- It suffices to divide only by primes in the above range.
- A list of primes may be unavailable.
- Checking trial divisors for primality is time-consuming.

Trial Division

- Divide n by $2, 3, 4, 5, \ldots, \lfloor \sqrt{n} \rfloor$.
- It suffices to divide only by primes in the above range.
- A list of primes may be unavailable.
- Checking trial divisors for primality is time-consuming.
- Divide n by $d \geqslant 30$ if and only if $d \equiv 1, 7, 11, 13, 17, 19, 23, 29 \pmod{30}$.

Trial Division

- Divide n by $2, 3, 4, 5, \ldots, \lfloor \sqrt{n} \rfloor$.
- It suffices to divide only by primes in the above range.
- A list of primes may be unavailable.
- Checking trial divisors for primality is time-consuming.
- Divide n by $d \geqslant 30$ if and only if $d \equiv 1, 7, 11, 13, 17, 19, 23, 29 \pmod{30}$.
- **Example**

Trial Division

- Divide n by $2, 3, 4, 5, \ldots, \lfloor \sqrt{n} \rfloor$.
- It suffices to divide only by primes in the above range.
- A list of primes may be unavailable.
- Checking trial divisors for primality is time-consuming.
- Divide n by $d \geqslant 30$ if and only if $d \equiv 1, 7, 11, 13, 17, 19, 23, 29 \pmod{30}$.
- **Example**
  - Take $n = 1716617$.

Trial Division

- Divide n by $2, 3, 4, 5, \ldots, \lfloor \sqrt{n} \rfloor$.
- It suffices to divide only by primes in the above range.
- A list of primes may be unavailable.
- Checking trial divisors for primality is time-consuming.
- Divide n by $d \geqslant 30$ if and only if $d \equiv 1, 7, 11, 13, 17, 19, 23, 29 \pmod{30}$.
- **Example**
  - Take $n = 1716617$.
  - Make trial divisions by primes $\leqslant 30$. Factor found: 7^2.

Trial Division

- Divide n by $2, 3, 4, 5, \ldots, \lfloor \sqrt{n} \rfloor$.
- It suffices to divide only by primes in the above range.
- A list of primes may be unavailable.
- Checking trial divisors for primality is time-consuming.
- Divide n by $d \geqslant 30$ if and only if $d \equiv 1, 7, 11, 13, 17, 19, 23, 29 \pmod{30}$.
- **Example**
  - Take $n = 1716617$.
  - Make trial divisions by primes $\leqslant 30$. Factor found: 7^2.
  - Make trial divisions by $31, 37, 41, 43, 47, 49$. No factor found.

Trial Division

- Divide n by $2, 3, 4, 5, \ldots, \lfloor\sqrt{n}\rfloor$.
- It suffices to divide only by primes in the above range.
- A list of primes may be unavailable.
- Checking trial divisors for primality is time-consuming.
- Divide n by $d \geqslant 30$ if and only if $d \equiv 1, 7, 11, 13, 17, 19, 23, 29 \pmod{30}$.
- **Example**
  - Take $n = 1716617$.
  - Make trial divisions by primes $\leqslant 30$. Factor found: 7^2.
  - Make trial divisions by $31, 37, 41, 43, 47, 49$. No factor found.
  - Trial division by 53 yields a factor.

Trial Division

- Divide n by $2, 3, 4, 5, \ldots, \lfloor \sqrt{n} \rfloor$.
- It suffices to divide only by primes in the above range.
- A list of primes may be unavailable.
- Checking trial divisors for primality is time-consuming.
- Divide n by $d \geqslant 30$ if and only if $d \equiv 1, 7, 11, 13, 17, 19, 23, 29 \pmod{30}$.
- **Example**
  - Take $n = 1716617$.
  - Make trial divisions by primes $\leqslant 30$. Factor found: 7^2.
  - Make trial divisions by $31, 37, 41, 43, 47, 49$. No factor found.
  - Trial division by 53 yields a factor.
  - The remaining part 661 is a prime.

Trial Division

- Divide n by $2, 3, 4, 5, \ldots, \lfloor \sqrt{n} \rfloor$.
- It suffices to divide only by primes in the above range.
- A list of primes may be unavailable.
- Checking trial divisors for primality is time-consuming.
- Divide n by $d \geqslant 30$ if and only if $d \equiv 1, 7, 11, 13, 17, 19, 23, 29 \pmod{30}$.
- **Example**
  - Take $n = 1716617$.
  - Make trial divisions by primes $\leqslant 30$. Factor found: 7^2.
  - Make trial divisions by $31, 37, 41, 43, 47, 49$. No factor found.
  - Trial division by 53 yields a factor.
  - The remaining part 661 is a prime.
  - Thus, we have $n = 7^2 \times 53 \times 661$.

Trial Division

- Divide n by $2, 3, 4, 5, \ldots, \lfloor \sqrt{n} \rfloor$.
- It suffices to divide only by primes in the above range.
- A list of primes may be unavailable.
- Checking trial divisors for primality is time-consuming.
- Divide n by $d \geqslant 30$ if and only if $d \equiv 1, 7, 11, 13, 17, 19, 23, 29 \pmod{30}$.
- **Example**
  - Take $n = 1716617$.
  - Make trial divisions by primes $\leqslant 30$. Factor found: 7^2.
  - Make trial divisions by $31, 37, 41, 43, 47, 49$. No factor found.
  - Trial division by 53 yields a factor.
  - The remaining part 661 is a prime.
  - Thus, we have $n = 7^2 \times 53 \times 661$.
- Trial division is efficient if n has only small prime factors (except possibly one).

Pollard's Rho Method

Let $p \leqslant \sqrt{n}$ be an unknown prime divisor of n.
Let $f : \mathbb{Z}_n \to \mathbb{Z}_n$ be a pseudorandom function.

Pollard's Rho Method

Let $p \leqslant \sqrt{n}$ be an unknown prime divisor of n.
Let $f : \mathbb{Z}_n \to \mathbb{Z}_n$ be a pseudorandom function.

- Choose a random $x_0 \in \mathbb{Z}_n$.

Pollard's Rho Method

Let $p \leqslant \sqrt{n}$ be an unknown prime divisor of n.
Let $f : \mathbb{Z}_n \rightarrow \mathbb{Z}_n$ be a pseudorandom function.

- Choose a random $x_0 \in \mathbb{Z}_n$.
- Generate a sequence $x_0, x_1, x_2, \ldots$ as $x_i = f(x_{i-1})$.

Pollard's Rho Method

Let $p \leqslant \sqrt{n}$ be an unknown prime divisor of n.
Let $f : \mathbb{Z}_n \to \mathbb{Z}_n$ be a pseudorandom function.

- Choose a random $x_0 \in \mathbb{Z}_n$.
- Generate a sequence $x_0, x_1, x_2, \ldots$ as $x_i = f(x_{i-1})$.
- Let $y_i \equiv x_i \pmod{p}$.

Pollard's Rho Method

Let $p \leqslant \sqrt{n}$ be an unknown prime divisor of n.
Let $f : \mathbb{Z}_n \to \mathbb{Z}_n$ be a pseudorandom function.

- Choose a random $x_0 \in \mathbb{Z}_n$.
- Generate a sequence $x_0, x_1, x_2, \ldots$ as $x_i = f(x_{i-1})$.
- Let $y_i \equiv x_i \pmod{p}$.
- The sequence $y_0, y_1, y_2, \ldots$ plays from behind the curtain.

Pollard's Rho Method

Let $p \leqslant \sqrt{n}$ be an unknown prime divisor of n.
Let $f : \mathbb{Z}_n \to \mathbb{Z}_n$ be a pseudorandom function.

- Choose a random $x_0 \in \mathbb{Z}_n$.
- Generate a sequence $x_0, x_1, x_2, \ldots$ as $x_i = f(x_{i-1})$.
- Let $y_i \equiv x_i \pmod{p}$.
- The sequence $y_0, y_1, y_2, \ldots$ plays from behind the curtain.
- The sequence $y_0, y_1, y_2, \ldots$ must be (eventually) periodic.

Pollard's Rho Method

Let $p \leqslant \sqrt{n}$ be an unknown prime divisor of n.
Let $f : \mathbb{Z}_n \to \mathbb{Z}_n$ be a pseudorandom function.

- Choose a random $x_0 \in \mathbb{Z}_n$.
- Generate a sequence $x_0, x_1, x_2, \ldots$ as $x_i = f(x_{i-1})$.
- Let $y_i \equiv x_i \pmod{p}$.
- The sequence $y_0, y_1, y_2, \ldots$ plays from behind the curtain.
- The sequence $y_0, y_1, y_2, \ldots$ must be (eventually) periodic.
- Suppose $y_i \equiv y_j \pmod{p}$ for $i < j$.

Pollard's Rho Method

Let $p \leqslant \sqrt{n}$ be an unknown prime divisor of n.
Let $f : \mathbb{Z}_n \to \mathbb{Z}_n$ be a pseudorandom function.

- Choose a random $x_0 \in \mathbb{Z}_n$.
- Generate a sequence $x_0, x_1, x_2, \ldots$ as $x_i = f(x_{i-1})$.
- Let $y_i \equiv x_i \pmod{p}$.
- The sequence $y_0, y_1, y_2, \ldots$ plays from behind the curtain.
- The sequence $y_0, y_1, y_2, \ldots$ must be (eventually) periodic.
- Suppose $y_i \equiv y_j \pmod{p}$ for $i < j$.
- If $x_i \not\equiv x_j \pmod{n}$, then $\gcd(x_i - x_j, n)$ is a non-trivial factor of n.

Pollard's Rho Method

Let $p \leqslant \sqrt{n}$ be an unknown prime divisor of n.
Let $f : \mathbb{Z}_n \to \mathbb{Z}_n$ be a pseudorandom function.

- Choose a random $x_0 \in \mathbb{Z}_n$.
- Generate a sequence $x_0, x_1, x_2, \ldots$ as $x_i = f(x_{i-1})$.
- Let $y_i \equiv x_i \pmod{p}$.
- The sequence $y_0, y_1, y_2, \ldots$ plays from behind the curtain.
- The sequence $y_0, y_1, y_2, \ldots$ must be (eventually) periodic.
- Suppose $y_i \equiv y_j \pmod{p}$ for $i < j$.
- If $x_i \not\equiv x_j \pmod{n}$, then $\gcd(x_i - x_j, n)$ is a non-trivial factor of n.
- By the **Birthday Paradox**, the expected running time of Pollard's rho method is $\tilde{O}(\sqrt[4]{n})$.

Pollard's Rho Method: Example

Let $n = 83947$.
Take $f(x) = x^2 - 1 \pmod{n}$.

Pollard's Rho Method: Example

Let $n = 83947$.

Take $f(x) = x^2 - 1 \pmod{n}$.

- Start with $x_0 = 123$.

Pollard's Rho Method: Example

Let $n = 83947$.
Take $f(x) = x^2 - 1 \pmod{n}$.

- Start with $x_0 = 123$.
- The first 20 terms in the x sequence are:
  $123, 15128, 16861, 48778,$ $67409, 6117, 61273, 18847,$
  $29651, 4869, 34106, 49603,$ $54985, 82966, 38943, 54693,$
  $40797, 61986, 10005, 35200$.

Pollard's Rho Method: Example

Let $n = 83947$.
Take $f(x) = x^2 - 1 \pmod{n}$.

- Start with $x_0 = 123$.
- The first 20 terms in the x sequence are:
  $123, 15128, 16861, 48778,\quad 67409, 6117, 61273, 18847,$
  $29651, 4869, 34106, 49603,\quad 54985, 82966, 38943, 54693,$
  $40797, 61986, 10005, 35200.$
- The corresponding terms in the y sequence are:
  $123, 15, 97, 10,\quad 99, 21, 59, 51,\quad 60, 43, 70, 73,$
  $121, 35, 81, 83,\quad 30, 10, 99, 21.$

Pollard's Rho Method: Example

Let $n = 83947$.
Take $f(x) = x^2 - 1 \pmod{n}$.

- Start with $x_0 = 123$.
- The first 20 terms in the x sequence are:
  $123, 15128, 16861, 48778,$ $67409, 6117, 61273, 18847,$
  $29651, 4869, 34106, 49603,$ $54985, 82966, 38943, 54693,$
  $40797, 61986, 10005, 35200.$
- The corresponding terms in the y sequence are:
  $123, 15, 97, 10,$ $99, 21, 59, 51,$ $60, 43, 70, 73,$
  $121, 35, 81, 83,$ $30, 10, 99, 21.$
- The periodic part in the y sequence is
  $10, 99, 21, 59, 51, 60, 43, 70, 73, 121, 35, 81, 83, 30.$

Pollard's Rho Method: Example

Let $n = 83947$.
Take $f(x) = x^2 - 1 \pmod{n}$.

- Start with $x_0 = 123$.
- The first 20 terms in the x sequence are:
  $123, 15128, 16861, 48778, \quad 67409, 6117, 61273, 18847,$
  $29651, 4869, 34106, 49603, \quad 54985, 82966, 38943, 54693,$
  $40797, 61986, 10005, 35200.$
- The corresponding terms in the y sequence are:
  $123, 15, 97, 10, \quad 99, 21, 59, 51, \quad 60, 43, 70, 73,$
  $121, 35, 81, 83, \quad 30, 10, 99, 21.$
- The periodic part in the y sequence is
  $10, 99, 21, 59, 51, 60, 43, 70, 73, 121, 35, 81, 83, 30.$
- The x sequence does not show the same periodicity.

Pollard's Rho Method: Example

Let $n = 83947$.
Take $f(x) = x^2 - 1 \pmod{n}$.

- Start with $x_0 = 123$.
- The first 20 terms in the x sequence are:
  $123, 15128, 16861, 48778, \quad 67409, 6117, 61273, 18847,$
  $29651, 4869, 34106, 49603, \quad 54985, 82966, 38943, 54693,$
  $40797, 61986, 10005, 35200.$
- The corresponding terms in the y sequence are:
  $123, 15, 97, 10, \quad 99, 21, 59, 51, \quad 60, 43, 70, 73,$
  $121, 35, 81, 83, \quad 30, 10, 99, 21.$
- The periodic part in the y sequence is
  $10, 99, 21, 59, 51, 60, 43, 70, 73, 121, 35, 81, 83, 30.$
- The x sequence does not show the same periodicity.
- $\gcd(61986 - 48778, n) = 127.$

Fermat's Factoring Method

Fermat's Factoring Method

Examples

Fermat's Factoring Method

Examples

- Take $n = 899$.
  $n = 900 - 1 = 30^2 - 1^2 = (30 - 1) \times (30 + 1) = 29 \times 31.$

Fermat's Factoring Method

Examples

- Take $n = 899$.
  $n = 900 - 1 = 30^2 - 1^2 = (30 - 1) \times (30 + 1) = 29 \times 31$.
- Take $n = 833$.
  $3 \times 833 = 2500 - 1 = 50^2 - 1^2 = (50 - 1) \times (50 + 1) = 49 \times 51$.
  $\gcd(50 - 1, 833) = 49$ is a non-trivial factor of 833.

Fermat's Factoring Method

Examples

- Take $n = 899$.
  $n = 900 - 1 = 30^2 - 1^2 = (30 - 1) \times (30 + 1) = 29 \times 31$.
- Take $n = 833$.
  $3 \times 833 = 2500 - 1 = 50^2 - 1^2 = (50 - 1) \times (50 + 1) = 49 \times 51$.
  $\gcd(50 - 1, 833) = 49$ is a non-trivial factor of 833.

Objective

To find integers $x, y \in \mathbb{Z}_n$ such that $x^2 \equiv y^2 \pmod{n}$. Unless $x \equiv \pm y \pmod{n}$, $\gcd(x - y, n)$ is a non-trivial divisor of n.

Fermat's Factoring Method

Examples

- Take $n = 899$.
  $n = 900 - 1 = 30^2 - 1^2 = (30 - 1) \times (30 + 1) = 29 \times 31$.
- Take $n = 833$.
  $3 \times 833 = 2500 - 1 = 50^2 - 1^2 = (50 - 1) \times (50 + 1) = 49 \times 51$.
  $\gcd(50 - 1, 833) = 49$ is a non-trivial factor of 833.

Objective

To find integers $x, y \in \mathbb{Z}_n$ such that $x^2 \equiv y^2 \pmod{n}$. Unless $x \equiv \pm y \pmod{n}$, $\gcd(x - y, n)$ is a non-trivial divisor of n.

If n is composite (but not a prime power), then for a randomly chosen pair (x, y) with $x^2 \equiv y^2 \pmod{n}$, the probability that $x \not\equiv \pm y \pmod{n}$ is at least $1/2$.

The Quadratic Sieve Method (QSM)

Let n be an odd integer with no small prime factors.
$H = \lceil \sqrt{n} \rceil$ and $J = H^2 - n$.

The Quadratic Sieve Method (QSM)

Let n be an odd integer with no small prime factors.
$H = \lceil \sqrt{n} \rceil$ and $J = H^2 - n$.

$(H + c)^2 \equiv J + 2Hc + c^2 \pmod{n}$ for small integers c.
Call $T(c) = J + 2Hc + c^2$.

The Quadratic Sieve Method (QSM)

Let n be an odd integer with no small prime factors.
$H = \lceil \sqrt{n} \rceil$ and $J = H^2 - n$.

$(H + c)^2 \equiv J + 2Hc + c^2 \pmod{n}$ for small integers c.
Call $T(c) = J + 2Hc + c^2$.

Suppose $T(c)$ factors over small primes $p_1, p_2, \ldots, p_t$:

$$(H + c)^2 \equiv p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_t^{\alpha_t} \pmod{n}.$$

This is called a **relation**.

The Quadratic Sieve Method (QSM)

Let n be an odd integer with no small prime factors.
$H = \lceil \sqrt{n} \rceil$ and $J = H^2 - n$.

$(H + c)^2 \equiv J + 2Hc + c^2 \pmod{n}$ for small integers c.
Call $T(c) = J + 2Hc + c^2$.

Suppose $T(c)$ factors over small primes $p_1, p_2, \ldots, p_t$:

$$(H + c)^2 \equiv p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_t^{\alpha_t} \pmod{n}.$$

This is called a **relation**.

The left side is already a square.
The right side is also a square if each α_i is even.
But this is very rare.

QSM (contd)

Collect many relations:

$$\left.\begin{array}{lrcl} \text{Relation 1:} & (H+c_1)^2 & = & p_1^{\alpha_{11}} p_2^{\alpha_{12}} \cdots p_t^{\alpha_{1t}} \\ \text{Relation 2:} & (H+c_2)^2 & = & p_1^{\alpha_{21}} p_2^{\alpha_{22}} \cdots p_t^{\alpha_{2t}} \\ \quad\cdots & & & \\ \text{Relation } r\text{:} & (H+c_r)^2 & = & p_1^{\alpha_{r1}} p_2^{\alpha_{r2}} \cdots p_t^{\alpha_{rt}} \end{array}\right\} \pmod{n}.$$

QSM (contd)

Collect many relations:

$$\left.\begin{array}{llcl}
\text{Relation 1:} & (H+c_1)^2 & = & p_1^{\alpha_{11}} p_2^{\alpha_{12}} \cdots p_t^{\alpha_{1t}} \\
\text{Relation 2:} & (H+c_2)^2 & = & p_1^{\alpha_{21}} p_2^{\alpha_{22}} \cdots p_t^{\alpha_{2t}} \\
\quad\cdots & & & \\
\text{Relation } r\text{:} & (H+c_r)^2 & = & p_1^{\alpha_{r1}} p_2^{\alpha_{r2}} \cdots p_t^{\alpha_{rt}}
\end{array}\right\} \pmod{n}.$$

Let $\beta_1, \beta_2, \ldots, \beta_r \in \{0, 1\}$.

$$\left[(H+c_1)^{\beta_1}(H+c_2)^{\beta_2}\cdots(H+c_r)^{\beta_r}\right]^2 \equiv p_1^{\gamma_1} p_2^{\gamma_2} \cdots p_t^{\gamma_t} \pmod{n}.$$

QSM (contd)

Collect many relations:

$$\left.\begin{array}{lrcl} \text{Relation 1:} & (H+c_1)^2 & = & p_1^{\alpha_{11}} p_2^{\alpha_{12}} \cdots p_t^{\alpha_{1t}} \\ \text{Relation 2:} & (H+c_2)^2 & = & p_1^{\alpha_{21}} p_2^{\alpha_{22}} \cdots p_t^{\alpha_{2t}} \\ \quad\cdots & & & \\ \text{Relation } r\text{:} & (H+c_r)^2 & = & p_1^{\alpha_{r1}} p_2^{\alpha_{r2}} \cdots p_t^{\alpha_{rt}} \end{array}\right\} \pmod{n}.$$

Let $\beta_1, \beta_2, \ldots, \beta_r \in \{0, 1\}$.

$$\left[(H+c_1)^{\beta_1}(H+c_2)^{\beta_2}\cdots(H+c_r)^{\beta_r}\right]^2 \equiv p_1^{\gamma_1} p_2^{\gamma_2} \cdots p_t^{\gamma_t} \pmod{n}.$$

The left side is already a square.
Tune $\beta_1, \beta_2, \ldots, \beta_r$ to make each γ_i even.

QSM (contd)

$$\begin{aligned}
\alpha_{11}\beta_1 + \alpha_{21}\beta_2 + \cdots + \alpha_{r1}\beta_r &= \gamma_1, \\
\alpha_{12}\beta_1 + \alpha_{22}\beta_2 + \cdots + \alpha_{r2}\beta_r &= \gamma_2, \\
&\cdots \\
\alpha_{1t}\beta_1 + \alpha_{2t}\beta_2 + \cdots + \alpha_{rt}\beta_r &= \gamma_t.
\end{aligned}$$

QSM (contd)

$$\begin{aligned}
\alpha_{11}\beta_1 + \alpha_{21}\beta_2 + \cdots + \alpha_{r1}\beta_r &= \gamma_1, \\
\alpha_{12}\beta_1 + \alpha_{22}\beta_2 + \cdots + \alpha_{r2}\beta_r &= \gamma_2, \\
&\cdots \\
\alpha_{1t}\beta_1 + \alpha_{2t}\beta_2 + \cdots + \alpha_{rt}\beta_r &= \gamma_t.
\end{aligned}$$

Linear system with t equations and r variables $\beta_1, \beta_2, \ldots, \beta_r$:

$$\begin{pmatrix} \alpha_{11} & \alpha_{21} & \cdots & \alpha_{r1} \\ \alpha_{12} & \alpha_{22} & \cdots & \alpha_{r2} \\ \vdots & \vdots & \cdots & \vdots \\ \alpha_{1t} & \alpha_{2t} & \cdots & \alpha_{rt} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_t \end{pmatrix} \equiv \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \pmod{2}.$$

QSM (contd)

For $r \geqslant t$, there are non-zero solutions for $\beta_1, \beta_2, \ldots, \beta_r$. Take

$$\begin{aligned} x &\equiv (H+c_1)^{\beta_1}(H+c_2)^{\beta_2}\cdots(H+c_r)^{\beta_r} \pmod n, \\ y &\equiv p_1^{\gamma_1/2} p_2^{\gamma_2/2} \cdots p_t^{\gamma_t/2} \pmod n. \end{aligned}$$

If $x \not\equiv \pm y \pmod n$, then $\gcd(x-y, n)$ is a non-trivial factor of n.

QSM (contd)

For $r \geqslant t$, there are non-zero solutions for $\beta_1, \beta_2, \ldots, \beta_r$. Take

$$\begin{aligned} x &\equiv (H+c_1)^{\beta_1}(H+c_2)^{\beta_2}\cdots(H+c_r)^{\beta_r} \pmod{n}, \\ y &\equiv p_1^{\gamma_1/2}p_2^{\gamma_2/2}\cdots p_t^{\gamma_t/2} \pmod{n}. \end{aligned}$$

If $x \not\equiv \pm y \pmod{n}$, then $\gcd(x-y, n)$ is a non-trivial factor of n.

Let $p = p_i$ be a small prime.

$p \mid T(c)$ implies $(H+c)^2 \equiv n \pmod{p}$.

If n is not a quadratic residue modulo p, then $p \nmid T(c)$ for any c.

Consider only the small primes p modulo which n is a quadratic residue.

Example of QSM: Parameters

$n = 7116491$.

$H = \lceil \sqrt{n} \rceil = 2668$.

Take all primes < 100 modulo which n is a square:

$$B = \{2, 5, 7, 17, 29, 31, 41, 59, 61, 67, 71, 79, 97\}.$$

$t = 13$.

Take $r = 13$. (In practice, one takes $r \approx 2t$.)

Example of QSM: Relations

$$
\left.\begin{array}{rrcl}
\text{Relation 1:} & (H+3)^2 & \equiv & 2 \times 5^3 \times 71 \\
\text{Relation 2:} & (H+8)^2 & \equiv & 5 \times 7 \times 31 \times 41 \\
\text{Relation 3:} & (H+49)^2 & \equiv & 2 \times 41^2 \times 79 \\
\text{Relation 4:} & (H+64)^2 & \equiv & 7 \times 29^2 \times 59 \\
\text{Relation 5:} & (H+81)^2 & \equiv & 2 \times 5 \times 7^2 \times 29 \times 31 \\
\text{Relation 6:} & (H+109)^2 & \equiv & 2 \times 7 \times 17 \times 41 \times 61 \\
\text{Relation 7:} & (H+128)^2 & \equiv & 5^3 \times 71 \times 79 \\
\text{Relation 8:} & (H+145)^2 & \equiv & 2 \times 71^2 \times 79 \\
\text{Relation 9:} & (H+182)^2 & \equiv & 17^2 \times 59^2 \\
\text{Relation 10:} & (H+228)^2 & \equiv & 5^2 \times 7^2 \times 17 \times 61 \\
\text{Relation 11:} & (H+267)^2 & \equiv & 2 \times 7^2 \times 17 \times 29 \times 31 \\
\text{Relation 12:} & (H+382)^2 & \equiv & 7 \times 59 \times 67 \times 79 \\
\text{Relation 13:} & (H+411)^2 & \equiv & 2 \times 5^4 \times 31 \times 61
\end{array}\right\} \pmod{n}.
$$

Example of QSM: Linear System

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 3 & 1 & 0 & 0 & 1 & 0 & 3 & 0 & 0 & 2 & 0 & 0 & 4 \\ 0 & 1 & 0 & 1 & 2 & 1 & 0 & 0 & 0 & 2 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \\ \beta_6 \\ \beta_7 \\ \beta_8 \\ \beta_9 \\ \beta_{10} \\ \beta_{11} \\ \beta_{12} \\ \beta_{13} \end{pmatrix} \equiv \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \pmod{2}.$$

Example of QSM: Solution of Relations

$(\beta_1, \beta_2, \beta_3, \ldots, \beta_{13})$	x	y	$\gcd(x-y, n)$
$(0,0,0,0,0,0,0,0,0,0,0,0,0)$	1	1	7116491
$(1,0,1,0,0,0,1,0,0,0,0,0,0)$	1755331	560322	1847
$(0,0,1,0,0,0,0,1,0,0,0,0,0)$	526430	459938	1847
$(1,0,0,0,0,0,1,1,0,0,0,0,0)$	7045367	7045367	7116491
$(0,0,0,0,0,0,0,0,1,0,0,0,0)$	2850	1003	1847
$(1,0,1,0,0,0,1,0,1,0,0,0,0)$	6916668	6916668	7116491
$(0,0,1,0,0,0,0,1,1,0,0,0,0)$	5862390	5862390	7116491
$(1,0,0,0,0,0,1,1,1,0,0,0,0)$	3674839	6944029	1847
$(0,1,0,0,1,1,0,0,0,0,1,0,1)$	1079130	3965027	3853
$(1,1,1,0,1,1,1,0,0,0,1,0,1)$	5466596	1649895	1
$(0,1,1,0,1,1,0,1,0,0,1,0,1)$	5395334	1721157	1
$(1,1,0,0,1,1,1,1,0,0,1,0,1)$	6429806	3725000	3853
$(0,1,0,0,1,1,0,0,1,0,1,0,1)$	1196388	5920103	1
$(1,1,1,0,1,1,1,0,1,0,1,0,1)$	1799801	3818773	3853
$(0,1,1,0,1,1,0,1,1,0,1,0,1)$	5081340	4129649	3853
$(1,1,0,0,1,1,1,1,1,0,1,0,1)$	7099266	17225	1

Sieving

Sieving

- To identify which values of $T(c) = J + 2Hc + c^2$ are smooth with respect to the factor base B.

Sieving

- To identify which values of $T(c) = J + 2Hc + c^2$ are smooth with respect to the factor base B.
- Trial division is too expensive.

Sieving

- To identify which values of $T(c) = J + 2Hc + c^2$ are smooth with respect to the factor base B.
- Trial division is too expensive.
- Sieving replaces trial division by single-precision subtractions.

Sieving

- To identify which values of $T(c) = J + 2Hc + c^2$ are smooth with respect to the factor base B.
- Trial division is too expensive.
- Sieving replaces trial division by single-precision subtractions.
- Use an array $\mathcal{A}$ indexed by c in the range $-M \leqslant c \leqslant M$.

Sieving

- To identify which values of $T(c) = J + 2Hc + c^2$ are smooth with respect to the factor base B.
- Trial division is too expensive.
- Sieving replaces trial division by single-precision subtractions.
- Use an array $\mathcal{A}$ indexed by c in the range $-M \leqslant c \leqslant M$.
- Initialize: $\mathcal{A}_c = \log |T(c)|$.

Sieving

- To identify which values of $T(c) = J + 2Hc + c^2$ are smooth with respect to the factor base B.
- Trial division is too expensive.
- Sieving replaces trial division by single-precision subtractions.
- Use an array $\mathcal{A}$ indexed by c in the range $-M \leqslant c \leqslant M$.
- Initialize: $\mathcal{A}_c = \log |T(c)|$.
- Let $q \in B$, and h a small positive integer.

Sieving

- To identify which values of $T(c) = J + 2Hc + c^2$ are smooth with respect to the factor base B.
- Trial division is too expensive.
- Sieving replaces trial division by single-precision subtractions.
- Use an array $\mathcal{A}$ indexed by c in the range $-M \leqslant c \leqslant M$.
- Initialize: $\mathcal{A}_c = \log |T(c)|$.
- Let $q \in B$, and h a small positive integer.
- Solve $T(c) \equiv 0 \pmod{q^h}$.

Sieving

- To identify which values of $T(c) = J + 2Hc + c^2$ are smooth with respect to the factor base B.
- Trial division is too expensive.
- Sieving replaces trial division by single-precision subtractions.
- Use an array $\mathcal{A}$ indexed by c in the range $-M \leqslant c \leqslant M$.
- Initialize: $\mathcal{A}_c = \log |T(c)|$.
- Let $q \in B$, and h a small positive integer.
- Solve $T(c) \equiv 0 \pmod{q^h}$.
- For each solution χ, subtract $\log q$ from $\mathcal{A}_c$ for all $c \equiv \chi \pmod{q^h}$.

Sieving

- To identify which values of $T(c) = J + 2Hc + c^2$ are smooth with respect to the factor base B.
- Trial division is too expensive.
- Sieving replaces trial division by single-precision subtractions.
- Use an array $\mathcal{A}$ indexed by c in the range $-M \leqslant c \leqslant M$.
- Initialize: $\mathcal{A}_c = \log|T(c)|$.
- Let $q \in B$, and h a small positive integer.
- Solve $T(c) \equiv 0 \pmod{q^h}$.
- For each solution χ, subtract $\log q$ from $\mathcal{A}_c$ for all $c \equiv \chi \pmod{q^h}$.
- $T(c)$ is B-smooth if and only if the remaining $\mathcal{A}_c \approx 0$.

Sieving

- To identify which values of $T(c) = J + 2Hc + c^2$ are smooth with respect to the factor base B.
- Trial division is too expensive.
- Sieving replaces trial division by single-precision subtractions.
- Use an array $\mathcal{A}$ indexed by c in the range $-M \leqslant c \leqslant M$.
- Initialize: $\mathcal{A}_c = \log |T(c)|$.
- Let $q \in B$, and h a small positive integer.
- Solve $T(c) \equiv 0 \pmod{q^h}$.
- For each solution χ, subtract $\log q$ from $\mathcal{A}_c$ for all $c \equiv \chi \pmod{q^h}$.
- $T(c)$ is B-smooth if and only if the remaining $\mathcal{A}_c \approx 0$.
- Each *smooth* $T(c)$ is factored by trial division.

The Discrete Logarithm Problem

The Discrete Logarithm Problem

To compute the discrete logarithm of a in $\mathbb{F}_q^*$ to the primitive base g.

The Discrete Logarithm Problem

To compute the discrete logarithm of a in $\mathbb{F}_q^*$ to the primitive base g.

Older algorithms

- Brute-force search
- Shanks' Baby-step-giant-step method
- Pollard's rho method
- Pollard's lambda method
- Pohlig-Hellman method (Efficient if $p - 1$ has only small prime divisors)

The Discrete Logarithm Problem

To compute the discrete logarithm of a in $\mathbb{F}_q^*$ to the primitive base g.

Older algorithms

- Brute-force search
- Shanks' Baby-step-giant-step method
- Pollard's rho method
- Pollard's lambda method
- Pohlig-Hellman method (Efficient if $p - 1$ has only small prime divisors)

Worst-case complexity: Exponential in $\log q$

Modern algorithms

Modern algorithms

Based on the index calculus method (ICM)

Modern algorithms

Based on the index calculus method (ICM)

Subexponential running time:
$L(q, \omega, c) = \exp\left[(c + \mathrm{o}(1))(\ln q)^{\omega}(\ln\ln q)^{1-\omega}\right].$

Modern algorithms

Based on the index calculus method (ICM)

Subexponential running time:
$L(q, \omega, c) = \exp\left[(c + \mathrm{o}(1))(\ln q)^{\omega}(\ln \ln q)^{1-\omega}\right]$.

Algorithm	Inventor(s)	Running time
Basic ICM	Western & Miller (1968)	$L(q, 1/2, c)$
Linear sieve method (LSM) Residue list sieve method Gaussian integer method	Coppersmith, Odlyzko & Schroeppel (1986)	$L(q, 1/2, 1)$
Cubic sieve method (CSM)	Reyneri	$L(q, 1/2, 0.816)$
Number field sieve method (NFSM) [for $\mathbb{F}_p$ only]	Gordon (1993)	$L(q, 1/3, 1.923)$
Coppersmith's method [for $\mathbb{F}_{2^n}$ only]	Coppersmith	$L(q, 1/3, 1.526)$

The Baby-Step-Giant-Step Method

Let G be a cyclic multiplicative group of size n.
Let g be a generator of G.
We plan to compute $\mathrm{ind}_a(g)$ for some $a \in G$.

The Baby-Step-Giant-Step Method

Let G be a cyclic multiplicative group of size n.
Let g be a generator of G.
We plan to compute $\mathrm{ind}_a(g)$ for some $a \in G$.

- Let $m = \left\lceil \sqrt{n} \right\rceil$.

The Baby-Step-Giant-Step Method

Let G be a cyclic multiplicative group of size n.
Let g be a generator of G.
We plan to compute $\operatorname{ind}_a(g)$ for some $a \in G$.

- Let $m = \lceil \sqrt{n} \rceil$.
- **Baby steps:** For $i \in \{0, 1, 2, \ldots, m-1\}$, compute g^i, and store (i, g^i) sorted with respect to the second element.

The Baby-Step-Giant-Step Method

Let G be a cyclic multiplicative group of size n.
Let g be a generator of G.
We plan to compute $\operatorname{ind}_a(g)$ for some $a \in G$.

- Let $m = \lceil \sqrt{n} \rceil$.
- **Baby steps:** For $i \in \{0, 1, 2, \ldots, m-1\}$, compute g^i, and store (i, g^i) sorted with respect to the second element.
- **Giant steps:** For $j = 0, 1, 2, \ldots, m-1$, compute ag^{-jm}, and try to locate ag^{-jm} in the table of baby steps.

The Baby-Step-Giant-Step Method

Let G be a cyclic multiplicative group of size n.
Let g be a generator of G.
We plan to compute $\operatorname{ind}_a(g)$ for some $a \in G$.

- Let $m = \lceil \sqrt{n} \rceil$.
- **Baby steps:** For $i \in \{0, 1, 2, \ldots, m-1\}$, compute g^i, and store (i, g^i) sorted with respect to the second element.
- **Giant steps:** For $j = 0, 1, 2, \ldots, m-1$, compute ag^{-jm}, and try to locate ag^{-jm} in the table of baby steps.
- If a search is successful, we have $ag^{-jm} = g^i$ for some i, j, that is, $a = g^{jm+i}$, that is, $\operatorname{ind}_g(a) = jm + i$.

The BSGS Method: Example

Take $G = \mathbb{F}_{43}^*$ with size $n = 42$, $m = \left\lceil \sqrt{42} \right\rceil = 7$, and $g = 19$.

The BSGS Method: Example

Take $G = \mathbb{F}_{43}^*$ with size $n = 42$, $m = \left\lceil \sqrt{42} \right\rceil = 7$, and $g = 19$.

Table of Baby Steps

i	0	6	2	1	3	5	4
g^i	1	11	17	19	22	30	31

The BSGS Method: Example

Take $G = \mathbb{F}_{43}^*$ with size $n = 42$, $m = \left\lceil \sqrt{42} \right\rceil = 7$, and $g = 19$.

Table of Baby Steps

i	0	6	2	1	3	5	4
g^i	1	11	17	19	22	30	31

Giant Steps: Take $a = 3$.

The BSGS Method: Example

Take $G = \mathbb{F}_{43}^*$ with size $n = 42$, $m = \left\lceil \sqrt{42} \right\rceil = 7$, and $g = 19$.

Table of Baby Steps

i	0	6	2	1	3	5	4
g^i	1	11	17	19	22	30	31

Giant Steps: Take $a = 3$.

- $j = 0$: $ag^{-0m} \equiv 3 \pmod{43}$ is not in the table.

The BSGS Method: Example

Take $G = \mathbb{F}_{43}^*$ with size $n = 42$, $m = \left\lceil \sqrt{42} \right\rceil = 7$, and $g = 19$.

Table of Baby Steps

i	0	6	2	1	3	5	4
g^i	1	11	17	19	22	30	31

Giant Steps: Take $a = 3$.

- $j = 0$: $ag^{-0m} \equiv 3 \pmod{43}$ is not in the table.
- $j = 1$: $ag^{-m} \equiv 21 \pmod{43}$ is not in the table.

The BSGS Method: Example

Take $G = \mathbb{F}_{43}^*$ with size $n = 42$, $m = \left\lceil \sqrt{42} \right\rceil = 7$, and $g = 19$.

Table of Baby Steps

i	0	6	2	1	3	5	4
g^i	1	11	17	19	22	30	31

Giant Steps: Take $a = 3$.

- $j = 0$: $ag^{-0m} \equiv 3 \pmod{43}$ is not in the table.
- $j = 1$: $ag^{-m} \equiv 21 \pmod{43}$ is not in the table.
- $j = 2$: $ag^{-2m} \equiv 18 \pmod{43}$ is not in the table.

The BSGS Method: Example

Take $G = \mathbb{F}_{43}^*$ with size $n = 42$, $m = \left\lceil \sqrt{42} \right\rceil = 7$, and $g = 19$.

Table of Baby Steps

i	0	6	2	1	3	5	4
g^i	1	11	17	19	22	30	31

Giant Steps: Take $a = 3$.

- $j = 0$: $ag^{-0m} \equiv 3 \pmod{43}$ is not in the table.
- $j = 1$: $ag^{-m} \equiv 21 \pmod{43}$ is not in the table.
- $j = 2$: $ag^{-2m} \equiv 18 \pmod{43}$ is not in the table.
- $j = 3$: $ag^{-3m} \equiv 40 \pmod{43}$ is not in the table.

The BSGS Method: Example

Take $G = \mathbb{F}_{43}^*$ with size $n = 42$, $m = \left\lceil \sqrt{42} \right\rceil = 7$, and $g = 19$.

Table of Baby Steps

i	0	6	2	1	3	5	4
g^i	1	11	17	19	22	30	31

Giant Steps: Take $a = 3$.

- $j = 0$: $ag^{-0m} \equiv 3 \pmod{43}$ is not in the table.
- $j = 1$: $ag^{-m} \equiv 21 \pmod{43}$ is not in the table.
- $j = 2$: $ag^{-2m} \equiv 18 \pmod{43}$ is not in the table.
- $j = 3$: $ag^{-3m} \equiv 40 \pmod{43}$ is not in the table.
- $j = 4$: $ag^{-4m} \equiv 22 \equiv g^3 \pmod{43}$, so $\operatorname{ind}_g(a) = 4 \times 7 + 3 = 31$.

The Basic ICM for Prime Fields: Precomputation

Goal: To compute $\operatorname{ind}_g(a)$ in $\mathbb{F}_p^*$ to a primitive root g modulo p.

The Basic ICM for Prime Fields: Precomputation

Goal: To compute $\mathrm{ind}_g(a)$ in $\mathbb{F}_p^*$ to a primitive root g modulo p.

Factor base: First t primes $B = \{p_1, p_2, \ldots, p_t\}$

The Basic ICM for Prime Fields: Precomputation

Goal: To compute $\operatorname{ind}_g(a)$ in $\mathbb{F}_p^*$ to a primitive root g modulo p.

Factor base: First t primes $B = \{p_1, p_2, \ldots, p_t\}$

To compute $d_i = \operatorname{ind}_g p_i$ for $i = 1, 2, \ldots, t$

The Basic ICM for Prime Fields: Precomputation

Goal: To compute $\operatorname{ind}_g(a)$ in $\mathbb{F}_p^*$ to a primitive root g modulo p.

Factor base: First t primes $B = \{p_1, p_2, \ldots, p_t\}$

To compute $d_i = \operatorname{ind}_g p_i$ for $i = 1, 2, \ldots, t$

For random $j \in \{1, 2, \ldots, p-2\}$, try to factor $g^j \pmod{p}$ over B.

The Basic ICM for Prime Fields: Precomputation

Goal: To compute $\operatorname{ind}_g(a)$ in $\mathbb{F}_p^*$ to a primitive root g modulo p.

Factor base: First t primes $B = \{p_1, p_2, \ldots, p_t\}$

To compute $d_i = \operatorname{ind}_g p_i$ for $i = 1, 2, \ldots, t$

For random $j \in \{1, 2, \ldots, p-2\}$, try to factor $g^j \pmod{p}$ over B.

Relation: $g^j \equiv p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_t^{\alpha_t} \pmod{p}$

The Basic ICM for Prime Fields: Precomputation

Goal: To compute $\operatorname{ind}_g(a)$ in $\mathbb{F}_p^*$ to a primitive root g modulo p.

Factor base: First t primes $B = \{p_1, p_2, \ldots, p_t\}$

To compute $d_i = \operatorname{ind}_g p_i$ for $i = 1, 2, \ldots, t$

For random $j \in \{1, 2, \ldots, p-2\}$, try to factor $g^j \pmod{p}$ over B.

Relation: $g^j \equiv p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_t^{\alpha_t} \pmod{p}$

Linear equation in t variables $d_1, d_2, \ldots, d_t$:

$$j \equiv \alpha_1 d_1 + \alpha_2 d_2 + \cdots + \alpha_t d_t \pmod{p-1}$$

The Basic ICM: Precomputation (contd)

Generate $r \geqslant t$ relations for different values of j:

$$\left.\begin{array}{lcl}
\text{Relation 1:} \quad j_1 & \equiv & \alpha_{11}d_1 + \alpha_{12}d_2 + \cdots + \alpha_{1t}d_t \\
\text{Relation 2:} \quad j_2 & \equiv & \alpha_{21}d_1 + \alpha_{22}d_2 + \cdots + \alpha_{2t}d_t \\
\quad \ldots & & \\
\text{Relation } r\text{:} \quad j_r & \equiv & \alpha_{r1}d_1 + \alpha_{r2}d_2 + \cdots + \alpha_{rt}d_t
\end{array}\right\} \pmod{p-1}.$$

The Basic ICM: Precomputation (contd)

Generate $r \geqslant t$ relations for different values of j:

$$\left.\begin{array}{lrcl} \text{Relation 1:} & j_1 & \equiv & \alpha_{11}d_1 + \alpha_{12}d_2 + \cdots + \alpha_{1t}d_t \\ \text{Relation 2:} & j_2 & \equiv & \alpha_{21}d_1 + \alpha_{22}d_2 + \cdots + \alpha_{2t}d_t \\ \quad\cdots & & & \\ \text{Relation } r\text{:} & j_r & \equiv & \alpha_{r1}d_1 + \alpha_{r2}d_2 + \cdots + \alpha_{rt}d_t \end{array}\right\} \pmod{p-1}.$$

Solve the system modulo $p-1$ to determine $d_1, d_2, \ldots, d_t$.

The Basic ICM: Second stage

The Basic ICM: Second stage

Choose random $j \in \{1, 2, \ldots, p-2\}$.
Try to factor $ag^j \pmod{p}$ over B.

The Basic ICM: Second stage

Choose random $j \in \{1, 2, \ldots, p-2\}$.
Try to factor $ag^j \pmod{p}$ over B.

A successful factorization gives:

$$ag^j \equiv p_1^{\beta_1} p_2^{\beta_2} \cdots p_t^{\beta_t} \pmod{p}.$$

The Basic ICM: Second stage

Choose random $j \in \{1, 2, \ldots, p-2\}$.
Try to factor $ag^j \pmod{p}$ over B.

A successful factorization gives:

$$ag^j \equiv p_1^{\beta_1} p_2^{\beta_2} \cdots p_t^{\beta_t} \pmod{p}.$$

Take discrete log:

$$\operatorname{ind}_g a \equiv -j + \beta_1 d_1 + \beta_2 d_2 + \cdots + \beta_t d_t \pmod{p-1}.$$

The Basic ICM: Second stage

Choose random $j \in \{1, 2, \ldots, p-2\}$.
Try to factor $ag^j \pmod{p}$ over B.

A successful factorization gives:

$$ag^j \equiv p_1^{\beta_1} p_2^{\beta_2} \cdots p_t^{\beta_t} \pmod{p}.$$

Take discrete log:

$$\operatorname{ind}_g a \equiv -j + \beta_1 d_1 + \beta_2 d_2 + \cdots + \beta_t d_t \pmod{p-1}.$$

Substitute the values of $d_1, d_2, \ldots, d_t$ to get $\operatorname{ind}_g a$.

The Basic ICM: Example (Precomputation)

Parameters: $p = 839$, $g = 31$, $B = \{2, 3, 5, 7, 11\}$, $t = 5$, $r = 10$.

The Basic ICM: Example (Precomputation)

Parameters: $p = 839$, $g = 31$, $B = \{2, 3, 5, 7, 11\}$, $t = 5$, $r = 10$.

Relations

$$\left.\begin{array}{lrcl}
\text{Relation 1:} & g^{118} & \equiv & 2^3 \times 5^2 \\
\text{Relation 2:} & g^{574} & \equiv & 2^7 \times 5 \\
\text{Relation 3:} & g^{318} & \equiv & 2^2 \times 3^3 \\
\text{Relation 4:} & g^{46} & \equiv & 2^7 \\
\text{Relation 5:} & g^{786} & \equiv & 2^2 \times 3^3 \times 7 \\
\text{Relation 6:} & g^{323} & \equiv & 2 \times 3 \times 11 \\
\text{Relation 7:} & g^{606} & \equiv & 3^4 \\
\text{Relation 8:} & g^{252} & \equiv & 2^3 \times 3^2 \times 7 \\
\text{Relation 9:} & g^{160} & \equiv & 3 \times 5^2 \\
\text{Relation 10:} & g^{600} & \equiv & 2 \times 3^3 \times 5
\end{array}\right\} \pmod{p}.$$

The Basic ICM: Example (Precomputation)

$$\begin{pmatrix} 3 & 0 & 2 & 0 & 0 \\ 7 & 0 & 1 & 0 & 0 \\ 2 & 3 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 \\ 2 & 3 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 4 & 0 & 0 & 0 \\ 3 & 2 & 0 & 1 & 0 \\ 0 & 1 & 2 & 0 & 0 \\ 1 & 3 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{pmatrix} \equiv \begin{pmatrix} 118 \\ 574 \\ 318 \\ 46 \\ 786 \\ 323 \\ 606 \\ 252 \\ 160 \\ 600 \end{pmatrix} \pmod{p-1}.$$

The Basic ICM: Example (Precomputation)

The coefficient matrix has full column rank (5) modulo $p - 1 = 838$.

The Basic ICM: Example (Precomputation)

The coefficient matrix has full column rank (5) modulo $p - 1 = 838$.

The solution is unique.

The Basic ICM: Example (Precomputation)

The coefficient matrix has full column rank (5) modulo $p - 1 = 838$.

The solution is unique.

$$\left.\begin{array}{rcrcr} d_1 & \equiv & \operatorname{ind}_{31} 2 & = & 246 \\ d_2 & \equiv & \operatorname{ind}_{31} 3 & = & 780 \\ d_3 & \equiv & \operatorname{ind}_{31} 5 & = & 528 \\ d_4 & \equiv & \operatorname{ind}_{31} 7 & = & 468 \\ d_5 & \equiv & \operatorname{ind}_{31} 11 & = & 135 \end{array}\right\} \pmod{p-1}.$$

The Basic ICM: Example (Second Stage)

The Basic ICM: Example (Second Stage)

Take $a = 561$.

$$ag^{312} \equiv 600 \equiv 2^3 \times 3 \times 5^2 \pmod{p}, \quad \text{that is,}$$

$$\operatorname{ind}_{31} 561 \equiv -312 + 3 \times 246 + 780 + 2 \times 528 \equiv 586 \pmod{p-1}.$$

The Basic ICM: Example (Second Stage)

Take $a = 561$.

$$\begin{aligned} ag^{312} &\equiv 600 \equiv 2^3 \times 3 \times 5^2 \pmod{p}, \quad \text{that is,} \\ \operatorname{ind}_{31} 561 &\equiv -312 + 3 \times 246 + 780 + 2 \times 528 \equiv 586 \pmod{p-1}. \end{aligned}$$

Take $a = 89$.

$$\begin{aligned} ag^{342} &\equiv 99 \equiv 3^2 \times 11 \pmod{p}, \quad \text{that is,} \\ \operatorname{ind}_{31} 89 &\equiv -342 + 2 \times 780 + 135 \equiv 515 \pmod{p-1}. \end{aligned}$$

The Basic ICM: Example (Second Stage)

Take $a = 561$.

$$\begin{aligned} ag^{312} &\equiv 600 \equiv 2^3 \times 3 \times 5^2 \pmod{p}, \quad \text{that is,} \\ \operatorname{ind}_{31} 561 &\equiv -312 + 3 \times 246 + 780 + 2 \times 528 \equiv 586 \pmod{p-1}. \end{aligned}$$

Take $a = 89$.

$$\begin{aligned} ag^{342} &\equiv 99 \equiv 3^2 \times 11 \pmod{p}, \quad \text{that is,} \\ \operatorname{ind}_{31} 89 &\equiv -342 + 2 \times 780 + 135 \equiv 515 \pmod{p-1}. \end{aligned}$$

Take $a = 625$.

$$\begin{aligned} ag^{806} &\equiv 70 \equiv 2 \times 5 \times 7 \pmod{p}, \quad \text{that is,} \\ \operatorname{ind}_{31} 625 &\equiv -806 + 246 + 528 + 468 \equiv 436 \pmod{p-1}. \end{aligned}$$

The Basic ICM for $\mathbb{F}_{2^n}$

Represent $\mathbb{F}_{2^n} = \mathbb{F}_2(\alpha)$, where $f(\alpha) = 0$. Let $g(\alpha)$ be a generator of $\mathbb{F}_{2^n}^*$. We plan to compute $\mathrm{ind}_{g(\alpha)}\, t(\alpha)$.

The Basic ICM for $\mathbb{F}_{2^n}$

Represent $\mathbb{F}_{2^n} = \mathbb{F}_2(\alpha)$, where $f(\alpha) = 0$. Let $g(\alpha)$ be a generator of $\mathbb{F}_{2^n}^*$. We plan to compute $\operatorname{ind}_{g(\alpha)} t(\alpha)$.

- **Factor base:** $B = \{u(\alpha) \mid \deg g \leqslant m\}$.

The Basic ICM for $\mathbb{F}_{2^n}$

Represent $\mathbb{F}_{2^n} = \mathbb{F}_2(\alpha)$, where $f(\alpha) = 0$. Let $g(\alpha)$ be a generator of $\mathbb{F}_{2^n}^*$. We plan to compute $\operatorname{ind}_{g(\alpha)} t(\alpha)$.

- **Factor base:** $B = \{u(\alpha) \mid \deg g \leqslant m\}$.
- **Relation:** Choose $j \in \{0, 1, 2, \ldots, 2^n - 2\}$ randomly, and try to arrive at factorizations of the form:

$$g(\alpha)^j = \prod_{u(\alpha) \in B} u(\alpha)^{\gamma_{u(\alpha)}}.$$

The Basic ICM for $\mathbb{F}_{2^n}$

Represent $\mathbb{F}_{2^n} = \mathbb{F}_2(\alpha)$, where $f(\alpha) = 0$. Let $g(\alpha)$ be a generator of $\mathbb{F}_{2^n}^*$. We plan to compute $\mathrm{ind}_{g(\alpha)}\, t(\alpha)$.

- **Factor base:** $B = \{u(\alpha) \mid \deg g \leqslant m\}$.
- **Relation:** Choose $j \in \{0, 1, 2, \ldots, 2^n - 2\}$ randomly, and try to arrive at factorizations of the form:

$$g(\alpha)^j = \prod_{u(\alpha)\in B} u(\alpha)^{\gamma_{u(\alpha)}}.$$

- **Linear algebra:** Solve the resulting system of congruences modulo $2^n - 1$, and obtain the indices $\mathrm{ind}_{g(\alpha)}\, u(\alpha)$ for all $u(\alpha) \in B$.

The Basic ICM for $\mathbb{F}_{2^n}$

Represent $\mathbb{F}_{2^n} = \mathbb{F}_2(\alpha)$, where $f(\alpha) = 0$. Let $g(\alpha)$ be a generator of $\mathbb{F}_{2^n}^*$. We plan to compute $\operatorname{ind}_{g(\alpha)} t(\alpha)$.

- **Factor base:** $B = \{u(\alpha) \mid \deg g \leqslant m\}$.
- **Relation:** Choose $j \in \{0, 1, 2, \ldots, 2^n - 2\}$ randomly, and try to arrive at factorizations of the form:

$$g(\alpha)^j = \prod_{u(\alpha) \in B} u(\alpha)^{\gamma_{u(\alpha)}}.$$

- **Linear algebra:** Solve the resulting system of congruences modulo $2^n - 1$, and obtain the indices $\operatorname{ind}_{g(\alpha)} u(\alpha)$ for all $u(\alpha) \in B$.
- **Second stage:** Generate a single relation of the form:

$$t(\alpha) g(\alpha)^j = \prod_{u(\alpha) \in B} u(\alpha)^{\delta_{u(\alpha)}}.$$

The Basic ICM: Example

The Basic ICM: Example

- Represent $\mathbb{F}_{128} = \mathbb{F}_2(\alpha)$ with $\alpha^7 + \alpha + 1 = 0$.

The Basic ICM: Example

- Represent $\mathbb{F}_{128} = \mathbb{F}_2(\alpha)$ with $\alpha^7 + \alpha + 1 = 0$.
- $|\mathbb{F}_{128}^*| = 127$ is prime. Take $g(\alpha) = \alpha^5 + \alpha^2 + 1$.

The Basic ICM: Example

- Represent $\mathbb{F}_{128} = \mathbb{F}_2(\alpha)$ with $\alpha^7 + \alpha + 1 = 0$.
- $|\mathbb{F}_{128}^*| = 127$ is prime. Take $g(\alpha) = \alpha^5 + \alpha^2 + 1$.
- Take $m = 2$, that is, $B = \{\alpha, \alpha + 1, \alpha^2 + \alpha + 1\}$.

The Basic ICM: Example

- Represent $\mathbb{F}_{128} = \mathbb{F}_2(\alpha)$ with $\alpha^7 + \alpha + 1 = 0$.
- $|\mathbb{F}_{128}^*| = 127$ is prime. Take $g(\alpha) = \alpha^5 + \alpha^2 + 1$.
- Take $m = 2$, that is, $B = \{\alpha, \alpha + 1, \alpha^2 + \alpha + 1\}$.
- **Relations in the first stage**

$$
\begin{aligned}
g(\alpha)^7 &= \alpha^6 + \alpha^2 = \alpha^2(\alpha + 1)^4, \\
g(\alpha)^{101} &= \alpha^4 + \alpha^3 + \alpha + 1 = (\alpha + 1)^2(\alpha^2 + \alpha + 1), \\
g(\alpha)^{121} &= \alpha^5 + \alpha^2 = \alpha^2(\alpha + 1)(\alpha^2 + \alpha + 1).
\end{aligned}
$$

The Basic ICM: Example

- Represent $\mathbb{F}_{128} = \mathbb{F}_2(\alpha)$ with $\alpha^7 + \alpha + 1 = 0$.
- $|\mathbb{F}^*_{128}| = 127$ is prime. Take $g(\alpha) = \alpha^5 + \alpha^2 + 1$.
- Take $m = 2$, that is, $B = \{\alpha, \alpha + 1, \alpha^2 + \alpha + 1\}$.
- **Relations in the first stage**

$$\begin{aligned} g(\alpha)^7 &= \alpha^6 + \alpha^2 = \alpha^2(\alpha+1)^4, \\ g(\alpha)^{101} &= \alpha^4 + \alpha^3 + \alpha + 1 = (\alpha+1)^2(\alpha^2+\alpha+1), \\ g(\alpha)^{121} &= \alpha^5 + \alpha^2 = \alpha^2(\alpha+1)(\alpha^2+\alpha+1). \end{aligned}$$

- **Linear system of congruences**

$$\begin{pmatrix} 2 & 4 & 0 \\ 0 & 2 & 1 \\ 2 & 1 & 1 \end{pmatrix} \begin{pmatrix} d_\alpha \\ d_{\alpha+1} \\ d_{\alpha^2+\alpha+1} \end{pmatrix} \equiv \begin{pmatrix} 7 \\ 101 \\ 121 \end{pmatrix} \pmod{127},$$

where $d_\beta = \operatorname{ind}_{g(\alpha)}(\beta)$.

The Basic ICM: Example (contd)

The Basic ICM: Example (contd)

- **Indices of factor base elements**
  $d_\alpha = 123$, $d_{\alpha+1} = 99$, and $d_{\alpha^2+\alpha+1} = 30$.

The Basic ICM: Example (contd)

- **Indices of factor base elements**
  $d_\alpha = 123$, $d_{\alpha+1} = 99$, and $d_{\alpha^2+\alpha+1} = 30$.
- **Second stage**

The Basic ICM: Example (contd)

- **Indices of factor base elements**
  $d_\alpha = 123$, $d_{\alpha+1} = 99$, and $d_{\alpha^2+\alpha+1} = 30$.

- **Second stage**
  - $t(\alpha) = \alpha^3 + 1$.
    $t(\alpha)g(\alpha)^{57} = \alpha^5 + \alpha^3 = \alpha^3(\alpha+1)^2$.
    $\mathrm{ind}_{g(\alpha)}\, t(\alpha) \equiv -57 + 3d_\alpha + 2d_{\alpha+1} \equiv 2 \pmod{127}$.

The Basic ICM: Example (contd)

- **Indices of factor base elements**
  $d_\alpha = 123$, $d_{\alpha+1} = 99$, and $d_{\alpha^2+\alpha+1} = 30$.

- **Second stage**
  - $t(\alpha) = \alpha^3 + 1$.
    $t(\alpha)g(\alpha)^{57} = \alpha^5 + \alpha^3 = \alpha^3(\alpha+1)^2$.
    $\operatorname{ind}_{g(\alpha)} t(\alpha) \equiv -57 + 3d_\alpha + 2d_{\alpha+1} \equiv 2 \pmod{127}$.
  - $t(\alpha) = \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1$.
    $t(\alpha)g(\alpha)^{73} = \alpha^5 + \alpha^4\alpha^2 + \alpha = \alpha(\alpha+1)^2(\alpha^2+\alpha+1)$.
    $\operatorname{ind}_{g(\alpha)} t(\alpha) \equiv -73 + d_\alpha + 2d_{\alpha+1} + d_{\alpha^2+\alpha+1} \equiv 24 \pmod{127}$.

The Basic ICM: Example (contd)

- **Indices of factor base elements**
  $d_\alpha = 123$, $d_{\alpha+1} = 99$, and $d_{\alpha^2+\alpha+1} = 30$.

- **Second stage**
  - $t(\alpha) = \alpha^3 + 1$.
    $t(\alpha)g(\alpha)^{57} = \alpha^5 + \alpha^3 = \alpha^3(\alpha+1)^2$.
    $\mathrm{ind}_{g(\alpha)}\, t(\alpha) \equiv -57 + 3d_\alpha + 2d_{\alpha+1} \equiv 2 \pmod{127}$.
  - $t(\alpha) = \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1$.
    $t(\alpha)g(\alpha)^{73} = \alpha^5 + \alpha^4\alpha^2 + \alpha = \alpha(\alpha+1)^2(\alpha^2+\alpha+1)$.
    $\mathrm{ind}_{g(\alpha)}\, t(\alpha) \equiv -73 + d_\alpha + 2d_{\alpha+1} + d_{\alpha^2+\alpha+1} \equiv 24 \pmod{127}$.
  - $t(\alpha) = \alpha^5 + 1$.
    $t(\alpha)g(\alpha)^{18} = \alpha^5 + \alpha^3 + \alpha = \alpha(\alpha^2+\alpha+1)^2$.
    $\mathrm{ind}_{g(\alpha)}\, t(\alpha) \equiv -18 + d_\alpha + 2d_{\alpha^2+\alpha+1} \equiv 38 \pmod{127}$.

The Elliptic Curve Discrete Logarithm Problem

The Elliptic Curve Discrete Logarithm Problem

- For a general curve, only the exponential square-root methods apply.

The Elliptic Curve Discrete Logarithm Problem

- For a general curve, only the exponential square-root methods apply.
- Index calculus methods for elliptic curves are neither known nor likely to exist.

The Elliptic Curve Discrete Logarithm Problem

- For a general curve, only the exponential square-root methods apply.
- Index calculus methods for elliptic curves are neither known nor likely to exist.
- The subexponential **MOV attack** applies to supersingular curves.

The Elliptic Curve Discrete Logarithm Problem

- For a general curve, only the exponential square-root methods apply.
- Index calculus methods for elliptic curves are neither known nor likely to exist.
- The subexponential **MOV attack** applies to supersingular curves.
- The linear-time **anomalous attack** (also called the **SmartASS attack**) applies to anomalous curves.

The Elliptic Curve Discrete Logarithm Problem

- For a general curve, only the exponential square-root methods apply.
- Index calculus methods for elliptic curves are neither known nor likely to exist.
- The subexponential **MOV attack** applies to supersingular curves.
- The linear-time **anomalous attack** (also called the **SmartASS attack**) applies to anomalous curves.
- Supersingular and anomalous curves are not used in cryptography.

The Elliptic Curve Discrete Logarithm Problem

- For a general curve, only the exponential square-root methods apply.
- Index calculus methods for elliptic curves are neither known nor likely to exist.
- The subexponential **MOV attack** applies to supersingular curves.
- The linear-time **anomalous attack** (also called the **SmartASS attack**) applies to anomalous curves.
- Supersingular and anomalous curves are not used in cryptography.
- The **Xedni calculus method** applies to general curves, but is found to be impractical.

Solving Large Sparse Linear Systems

Solving Large Sparse Linear Systems

- To solve $A\mathbf{x} \equiv \mathbf{b} \pmod{M}$, where A a **sparse** $m \times n$ matrix.

Solving Large Sparse Linear Systems

- To solve $A\mathbf{x} \equiv \mathbf{b} \pmod{M}$, where A a **sparse** $m \times n$ matrix.
- If M is prime, we work in the finite field $\mathbb{F}_M$.

Solving Large Sparse Linear Systems

- To solve $A\mathbf{x} \equiv \mathbf{b} \pmod{M}$, where A a **sparse** $m \times n$ matrix.
- If M is prime, we work in the finite field $\mathbb{F}_M$.
- If M is composite with known factorization, we solve the system modulo prime power divisors of M.

Solving Large Sparse Linear Systems

- To solve $A\mathbf{x} \equiv \mathbf{b} \pmod{M}$, where A a **sparse** $m \times n$ matrix.
- If M is prime, we work in the finite field $\mathbb{F}_M$.
- If M is composite with known factorization, we solve the system modulo prime power divisors of M.
- If M is composite with unknown factorization, we pretend M as prime. If inversion modulo M fails, we discover non-trivial factors of M, and solve the system modulo each factor thus discovered.

Solving Large Sparse Linear Systems

- To solve $A\mathbf{x} \equiv \mathbf{b} \pmod{M}$, where A a **sparse** $m \times n$ matrix.
- If M is prime, we work in the finite field $\mathbb{F}_M$.
- If M is composite with known factorization, we solve the system modulo prime power divisors of M.
- If M is composite with unknown factorization, we pretend M as prime. If inversion modulo M fails, we discover non-trivial factors of M, and solve the system modulo each factor thus discovered.
- We have $m = \Theta(n)$.

Solving Large Sparse Linear Systems

- To solve $A\mathbf{x} \equiv \mathbf{b} \pmod{M}$, where A a **sparse** $m \times n$ matrix.
- If M is prime, we work in the finite field $\mathbb{F}_M$.
- If M is composite with known factorization, we solve the system modulo prime power divisors of M.
- If M is composite with unknown factorization, we pretend M as prime. If inversion modulo M fails, we discover non-trivial factors of M, and solve the system modulo each factor thus discovered.
- We have $m = \Theta(n)$.
- If A is dense, the system solving phase runs in $\tilde{O}(n^3)$ time.

Solving Large Sparse Linear Systems

- To solve $A\mathbf{x} \equiv \mathbf{b} \pmod{M}$, where A a **sparse** $m \times n$ matrix.
- If M is prime, we work in the finite field $\mathbb{F}_M$.
- If M is composite with known factorization, we solve the system modulo prime power divisors of M.
- If M is composite with unknown factorization, we pretend M as prime. If inversion modulo M fails, we discover non-trivial factors of M, and solve the system modulo each factor thus discovered.
- We have $m = \Theta(n)$.
- If A is dense, the system solving phase runs in $\tilde{O}(n^3)$ time.
- If A is sparse, there are $\tilde{O}(n^2)$-time algorithms.

Structured Gaussian Elimination

Structured Gaussian Elimination

- Used to reduce the size of A.

Structured Gaussian Elimination

- Used to reduce the size of A.
- The size reduction often becomes substantial.

Structured Gaussian Elimination

- Used to reduce the size of A.
- The size reduction often becomes substantial.
- The reduced matrix becomes much denser.

Structured Gaussian Elimination

- Used to reduce the size of A.
- The size reduction often becomes substantial.
- The reduced matrix becomes much denser.
- Some steps of structured Gaussian elimination:

Structured Gaussian Elimination

- Used to reduce the size of A.
- The size reduction often becomes substantial.
- The reduced matrix becomes much denser.
- Some steps of structured Gaussian elimination:
  - Delete zero columns.

Structured Gaussian Elimination

- Used to reduce the size of A.
- The size reduction often becomes substantial.
- The reduced matrix becomes much denser.
- Some steps of structured Gaussian elimination:
  - Delete zero columns.
  - Delete columns with single non-zero entries and the corresponding rows.

Structured Gaussian Elimination

- Used to reduce the size of A.
- The size reduction often becomes substantial.
- The reduced matrix becomes much denser.
- Some steps of structured Gaussian elimination:
  - Delete zero columns.
  - Delete columns with single non-zero entries and the corresponding rows.
  - Delete rows with single non-zero entries.

Structured Gaussian Elimination

- Used to reduce the size of A.
- The size reduction often becomes substantial.
- The reduced matrix becomes much denser.
- Some steps of structured Gaussian elimination:
  - Delete zero columns.
  - Delete columns with single non-zero entries and the corresponding rows.
  - Delete rows with single non-zero entries.
  - Throw excess rows with large numbers of non-zero entries.

Lanczos Method

Let A be a symmetric positive-definite matrix with real entries.
We plan to solve $A\mathbf{x} = \mathbf{b}$.
We generate a set of pairwise orthogonal directions $\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2, \ldots$ until we run out of new orthogonal directions.

Lanczos Method

Let A be a symmetric positive-definite matrix with real entries.
We plan to solve $A\mathbf{x} = \mathbf{b}$.
We generate a set of pairwise orthogonal directions $\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2, \ldots$ until we run out of new orthogonal directions.

- **Initialization**

  $\mathbf{d}_0 = \mathbf{b}$, $\mathbf{v}_1 = A\mathbf{d}_0$, $\mathbf{d}_1 = \mathbf{v}_1 - \mathbf{d}_0(\mathbf{v}_1^t A\mathbf{d}_0)/(\mathbf{d}_0^t A\mathbf{d}_0)$,
  $a_0 = (\mathbf{d}_0^t\mathbf{d}_0)/(\mathbf{d}_0^t A\mathbf{d}_0)$, and $\mathbf{x}_0 = a_0\mathbf{d}_0$.

Lanczos Method

Let A be a symmetric positive-definite matrix with real entries.
We plan to solve $A\mathbf{x} = \mathbf{b}$.
We generate a set of pairwise orthogonal directions $\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2, \ldots$ until we run out of new orthogonal directions.

- **Initialization**

  $\mathbf{d}_0 = \mathbf{b}$, $\mathbf{v}_1 = A\mathbf{d}_0$, $\mathbf{d}_1 = \mathbf{v}_1 - \mathbf{d}_0(\mathbf{v}_1^{\mathrm{t}} A\mathbf{d}_0)/(\mathbf{d}_0^{\mathrm{t}} A\mathbf{d}_0)$,
  $a_0 = (\mathbf{d}_0^{\mathrm{t}}\mathbf{d}_0)/(\mathbf{d}_0^{\mathrm{t}} A\mathbf{d}_0)$, and $\mathbf{x}_0 = a_0\mathbf{d}_0$.

- **Iteration:** For $i = 1, 2, 3, \ldots$, repeat:

  $\mathbf{v}_{i+1} = A\mathbf{d}_i$.
  $\mathbf{d}_{i+1} = \mathbf{v}_{i+1} - \mathbf{d}_i(\mathbf{v}_{i+1}^{\mathrm{t}} A\mathbf{d}_i)/(\mathbf{d}_i^{\mathrm{t}} A\mathbf{d}_i) - \mathbf{d}_{i-1}(\mathbf{v}_{i+1}^{\mathrm{t}} A\mathbf{d}_{i-1})/(\mathbf{d}_{i-1}^{\mathrm{t}} A\mathbf{d}_{i-1})$.
  $a_i = (\mathbf{d}_i^{\mathrm{t}}\mathbf{b})/(\mathbf{d}_i^{\mathrm{t}} A\mathbf{d}_i)$.
  $\mathbf{x}_i = \mathbf{x}_{i-1} + a_i\mathbf{d}_i$.

Adaptation of the Lanczos Method

Adaptation of the Lanczos Method

- In general, A is not a square matrix.

Adaptation of the Lanczos Method

- In general, A is not a square matrix.
- **Remedy**

Adaptation of the Lanczos Method

- In general, A is not a square matrix.
- **Remedy**
  - $(A^{\mathrm{t}}A)\mathbf{x} = A^{\mathrm{t}}\mathbf{b}$ is a square $(n \times n)$ system.

Adaptation of the Lanczos Method

- In general, A is not a square matrix.
- **Remedy**
  - $(A^{\mathrm{t}}A)\mathbf{x} = A^{\mathrm{t}}\mathbf{b}$ is a square $(n \times n)$ system.
  - Moreover, $A^{\mathrm{t}}A$ is symmetric.

Adaptation of the Lanczos Method

- In general, A is not a square matrix.
- **Remedy**
  - $(A^{\mathrm{t}}A)\mathbf{x} = A^{\mathrm{t}}\mathbf{b}$ is a square $(n \times n)$ system.
  - Moreover, $A^{\mathrm{t}}A$ is symmetric.
  - Instead of computing $A^{\mathrm{t}}A$, multiply separately by A and A^{t}.

Adaptation of the Lanczos Method

- In general, A is not a square matrix.
- **Remedy**
  - $(A^{\mathrm{t}}A)\mathbf{x} = A^{\mathrm{t}}\mathbf{b}$ is a square $(n \times n)$ system.
  - Moreover, $A^{\mathrm{t}}A$ is symmetric.
  - Instead of computing $A^{\mathrm{t}}A$, multiply separately by A and A^{t}.
- Positive-definiteness makes no sense in $\mathbb{Z}_M$. Problem arises when we encounter a non-zero vector $\mathbf{d}_i$ with $\mathbf{d}_i^{\mathrm{t}} A \mathbf{d}_i = 0$. The problem is likely to occur unless M is large.

Adaptation of the Lanczos Method

- In general, A is not a square matrix.
- **Remedy**
  - $(A^{\mathrm{t}}A)\mathbf{x} = A^{\mathrm{t}}\mathbf{b}$ is a square $(n \times n)$ system.
  - Moreover, $A^{\mathrm{t}}A$ is symmetric.
  - Instead of computing $A^{\mathrm{t}}A$, multiply separately by A and A^{t}.
- Positive-definiteness makes no sense in $\mathbb{Z}_M$. Problem arises when we encounter a non-zero vector $\mathbf{d}_i$ with $\mathbf{d}_i^{\mathrm{t}} A \mathbf{d}_i = 0$. The problem is likely to occur unless M is large.
- **Remedy**

Adaptation of the Lanczos Method

- In general, A is not a square matrix.
- **Remedy**
  - $(A^{\mathrm{t}}A)\mathbf{x} = A^{\mathrm{t}}\mathbf{b}$ is a square ($n \times n$) system.
  - Moreover, $A^{\mathrm{t}}A$ is symmetric.
  - Instead of computing $A^{\mathrm{t}}A$, multiply separately by A and A^{t}.
- Positive-definiteness makes no sense in $\mathbb{Z}_M$. Problem arises when we encounter a non-zero vector $\mathbf{d}_i$ with $\mathbf{d}_i^{\mathrm{t}}A\mathbf{d}_i = 0$. The problem is likely to occur unless M is large.
- **Remedy**
  - Work in extension fields ($\mathbb{F}_{M^s}$ in place of $\mathbb{F}_M$).

Adaptation of the Lanczos Method

- In general, A is not a square matrix.
- **Remedy**
  - $(A^{\mathrm{t}}A)\mathbf{x} = A^{\mathrm{t}}\mathbf{b}$ is a square ($n \times n$) system.
  - Moreover, $A^{\mathrm{t}}A$ is symmetric.
  - Instead of computing $A^{\mathrm{t}}A$, multiply separately by A and A^{t}.
- Positive-definiteness makes no sense in $\mathbb{Z}_M$. Problem arises when we encounter a non-zero vector $\mathbf{d}_i$ with $\mathbf{d}_i^{\mathrm{t}}A\mathbf{d}_i = 0$. The problem is likely to occur unless M is large.
- **Remedy**
  - Work in extension fields ($\mathbb{F}_{M^s}$ in place of $\mathbb{F}_M$).
  - Solve $D(A^{\mathrm{t}}A)\mathbf{x} = DA^{\mathrm{t}}\mathbf{b}$ for random non-singular diagonal matrices D.

Other Sparse System Solvers

Other Sparse System Solvers

- **Conjugate gradient method:** An iterative method similar to the Lanczos method.

Other Sparse System Solvers

- **Conjugate gradient method:** An iterative method similar to the Lanczos method.
- **Wiedemann method:** Computes the minimal polynomial of A in $\mathbb{F}_M[x]$.

Other Sparse System Solvers

- **Conjugate gradient method:** An iterative method similar to the Lanczos method.
- **Wiedemann method:** Computes the minimal polynomial of A in $\mathbb{F}_M[x]$.
- **Block Lanczos method:**

Other Sparse System Solvers

- **Conjugate gradient method:** An iterative method similar to the Lanczos method.
- **Wiedemann method:** Computes the minimal polynomial of A in $\mathbb{F}_M[x]$.
- **Block Lanczos method:**
  - Meant for systems modulo 2.

Other Sparse System Solvers

- **Conjugate gradient method:** An iterative method similar to the Lanczos method.
- **Wiedemann method:** Computes the minimal polynomial of A in $\mathbb{F}_M[x]$.
- **Block Lanczos method:**
  - Meant for systems modulo 2.
  - Bits are packed into words.

Other Sparse System Solvers

- **Conjugate gradient method:** An iterative method similar to the Lanczos method.
- **Wiedemann method:** Computes the minimal polynomial of A in $\mathbb{F}_M[x]$.
- **Block Lanczos method:**
  - Meant for systems modulo 2.
  - Bits are packed into words.
  - Multiple direction vectors are computed per iteration.

Other Sparse System Solvers

- **Conjugate gradient method:** An iterative method similar to the Lanczos method.
- **Wiedemann method:** Computes the minimal polynomial of A in $\mathbb{F}_M[x]$.
- **Block Lanczos method:**
  - Meant for systems modulo 2.
  - Bits are packed into words.
  - Multiple direction vectors are computed per iteration.
  - The problem of self-orthogonality of non-zero vectors is less acute.

Other Sparse System Solvers

- **Conjugate gradient method:** An iterative method similar to the Lanczos method.
- **Wiedemann method:** Computes the minimal polynomial of A in $\mathbb{F}_M[x]$.
- **Block Lanczos method:**
  - Meant for systems modulo 2.
  - Bits are packed into words.
  - Multiple direction vectors are computed per iteration.
  - The problem of self-orthogonality of non-zero vectors is less acute.
- **Block Wiedemann method:** The block implementation of the Wiedemann method for systems modulo 2.