# Combinatorics in Algebraic and Logical Cryptanalysis

**Monika Trimoska**

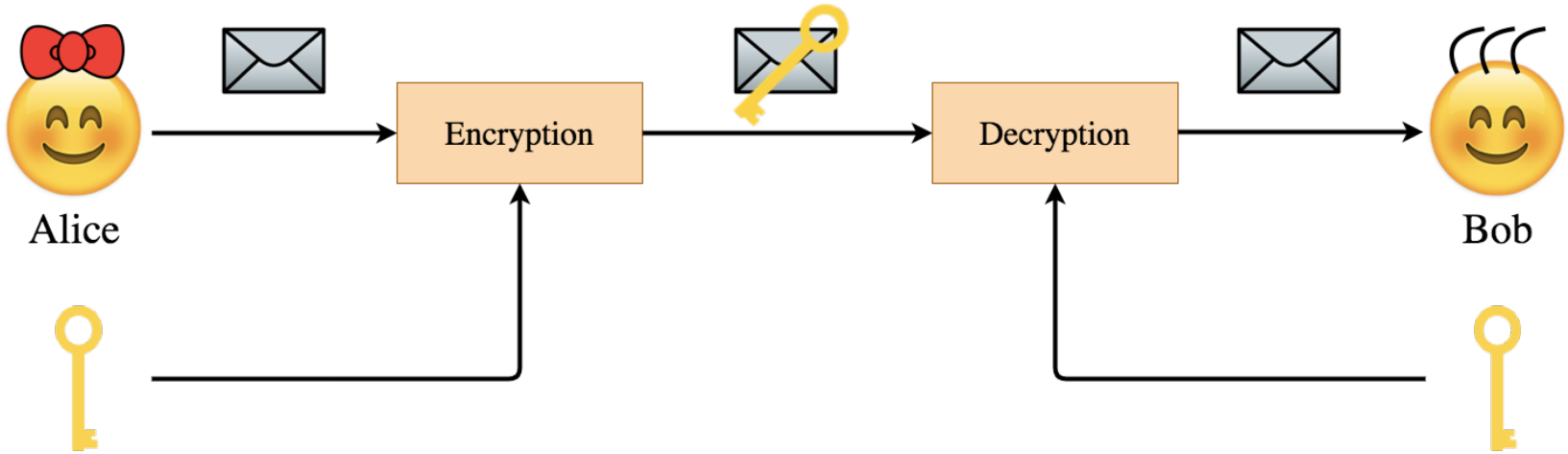
MIS, University of Picardie Jules Verne

PhD defense
14 January 2021





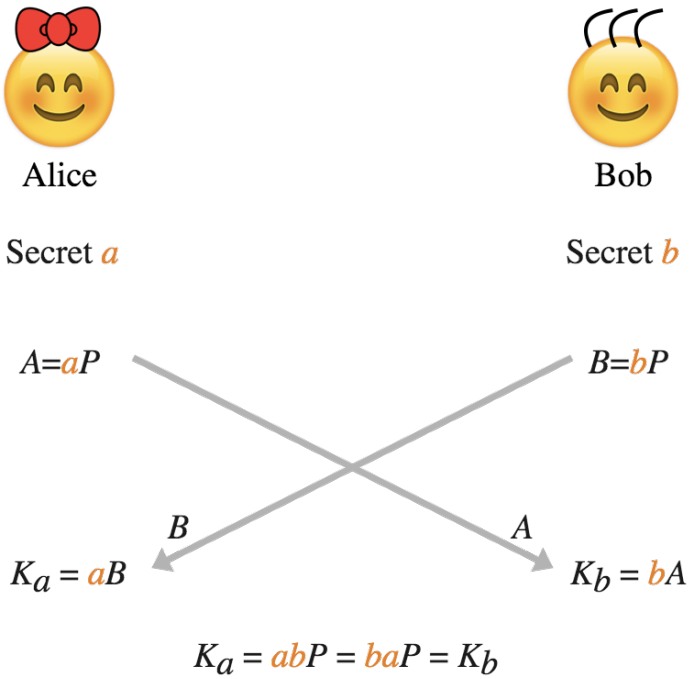
# Secret-key cryptography




The communicating parties share a secret key.

# Public-key cryptography

Protocols for performing secret-key exchange.




Use of one-way functions: easy to compute, hard to invert.

# Hard problems

There is no known polynomial-time algorithm for finding a solution or deciding if a solution exists. But, verifying if a solution is correct can be done in polynomial time.

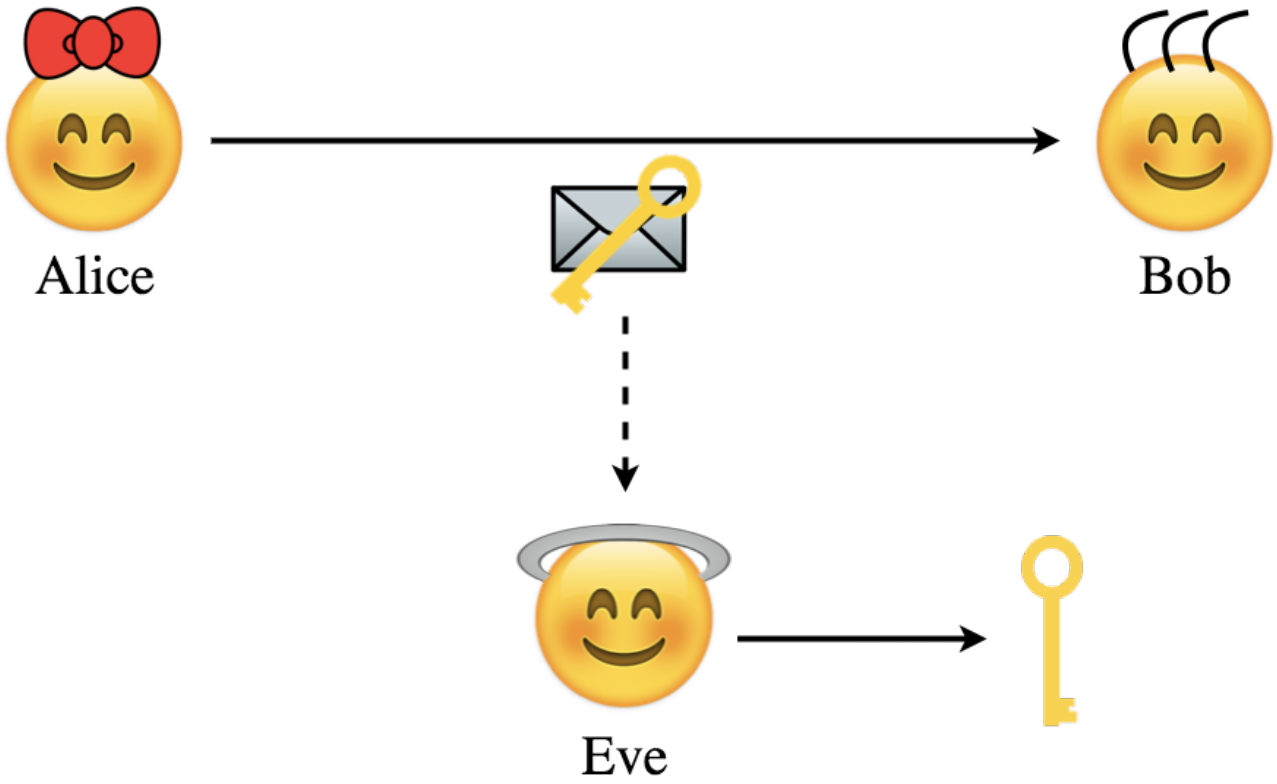
## NP Problems

- Knapsack problem
- Vertex cover problem
- Graph coloring problem

## Boolean satisfiability problem (SAT)

SAT is the problem of determining whether there exists an assignment of Boolean variables that satisfies a given propositional formula.

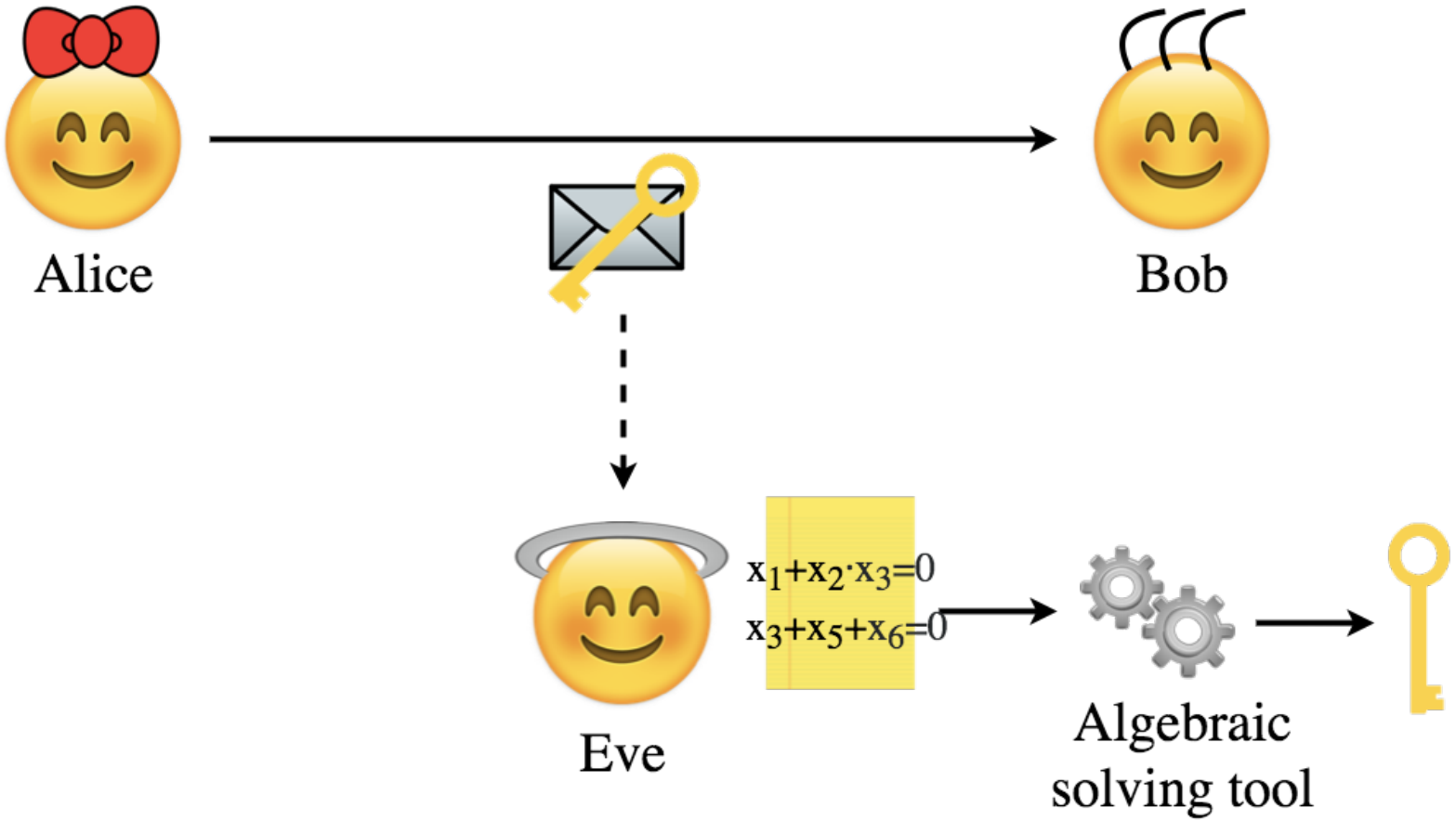
# Cryptanalysis



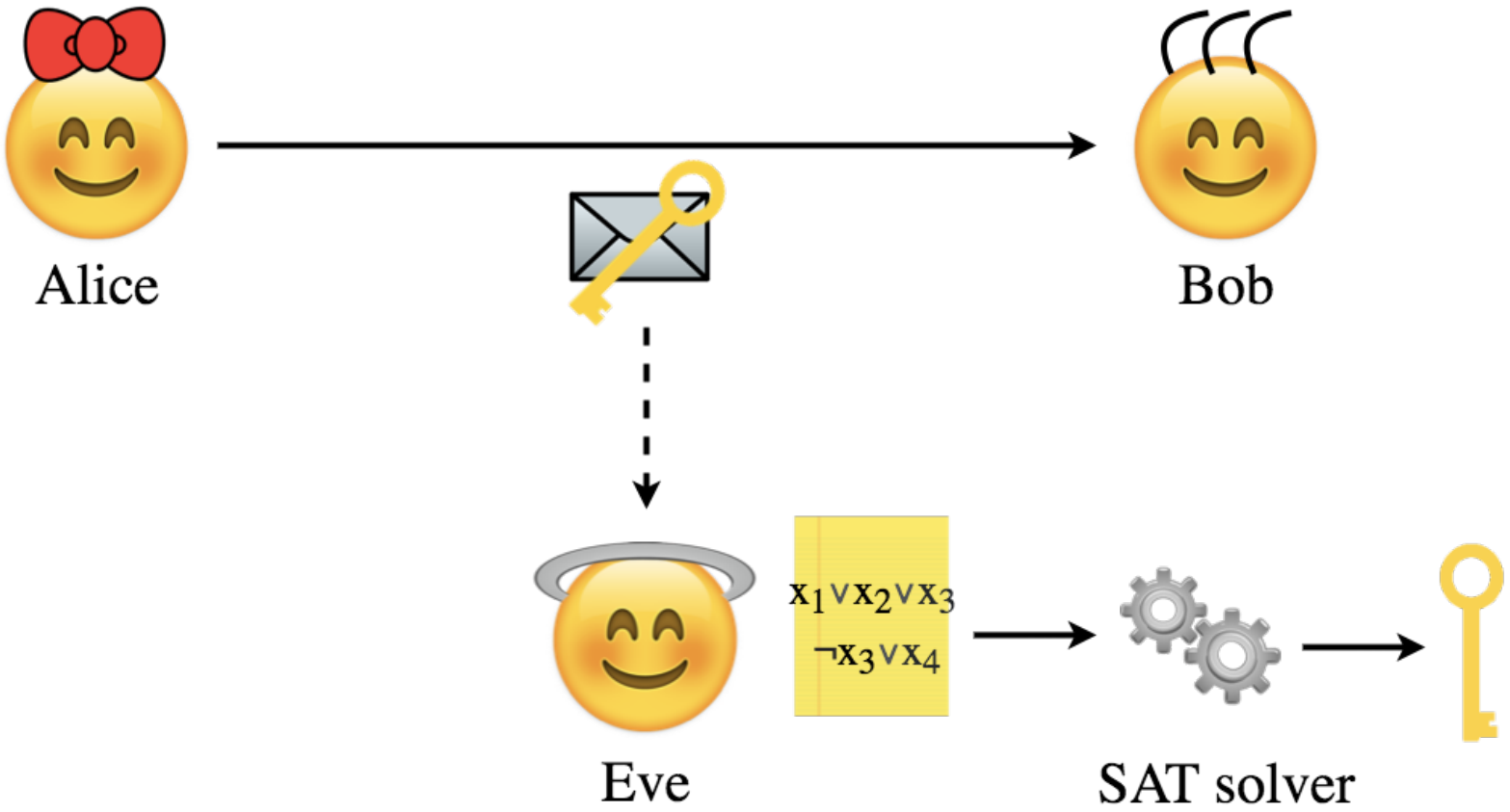

**Goal**

Determine minimum key length requirements.

# Algebraic cryptanalysis



# Logical cryptanalysis



# Discrete log problem

## Defining discrete log problem

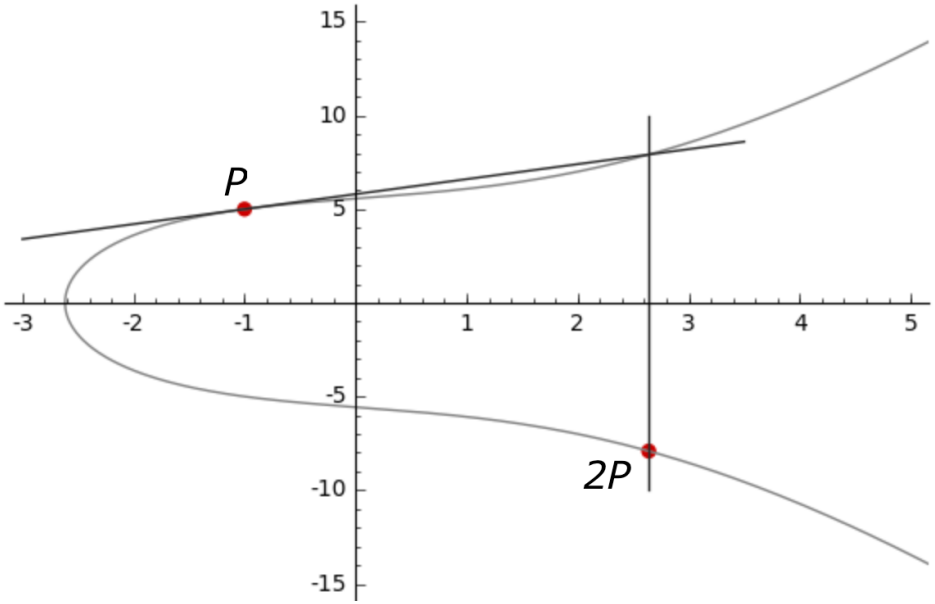
Given a finite cyclic group $(G, +)$ of order $N$ and two elements $g$, $h \in G$, find $x \in \mathbb{Z}$ such that

$$h = x \cdot g.$$

# Elliptic curve discrete log problem (ECDLP)

Let $K$ be a finite field, $a_1, a_2, a_3, a_4, a_6 \in K$ and let $E$ be an elliptic curve defined by

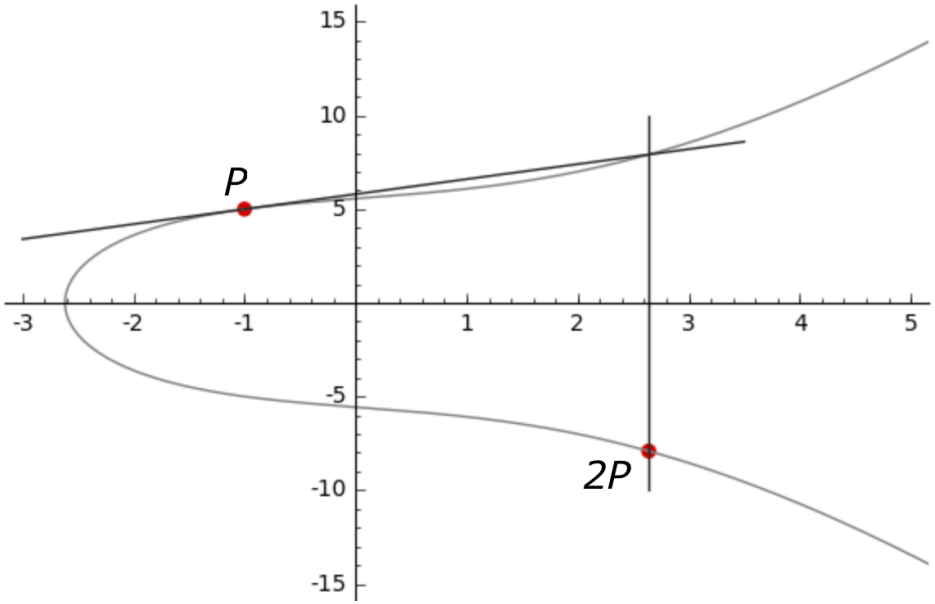
$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$



# Elliptic curve discrete log problem (ECDLP)

Let $K$ be a finite field, $a_1, a_2, a_3, a_4, a_6 \in K$ and let $E$ be an elliptic curve defined by

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

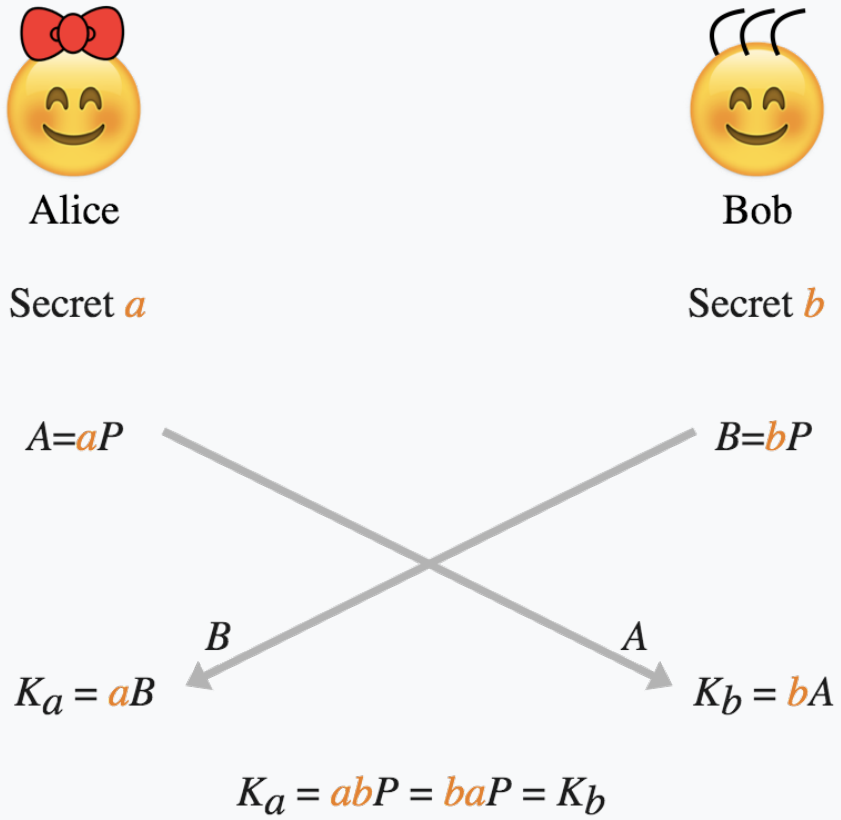



**Defining ECDLP:**

Find $x \in \mathbb{Z}$, such that $xP = Q$, where $P, Q \in E(K)$.

# Application example

## Diffie-Hellman key exchange




$E$ and $P \in E(K)$ are public parameters.

# Cryptographic attacks

1. Generic attacks
   - Parallel Collision Search algorithm (PCS)

     TCHES, Volume 2021, Issue 2

2. Attacks on specific families
   - Index calculus attack on elliptic curves defined over $\mathbb{F}_{2^n}$, with $n$ prime.

     CP 2020

     AfricaCrypt 2020

# Parallel Collision Search

## Solving the ECDLP

Having two different linear combinations of a random point $R \in E(K)$

$$R = aP + bQ$$
$$R = a'P + b'Q,$$

we infer that

$$aP + bQ = a'P + b'Q$$
$$(a - a')P = (b' - b)xP,$$

and we compute

$$x = \frac{a - a'}{b' - b} \pmod{N}.$$

# Collision search

## Collision

Given a random map $f : S \to S$ on a finite set $S$, we call collision any pair $R, R'$ of elements in $S$ such that $f(R) = f(R')$.

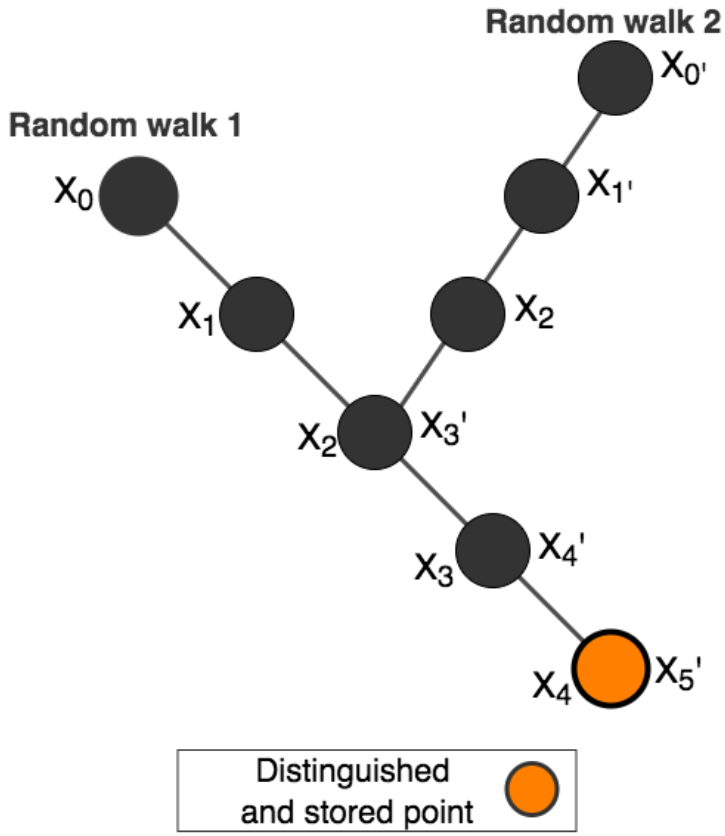
$$f(R) = \begin{cases} R + P & \text{if } R \in S_1 \\ 2R & \text{if } R \in S_2 \\ R + Q & \text{if } R \in S_3, \end{cases}$$

## Property of $f$

Input $(aP + bQ) \to$ Output $(a'P + b'Q)$.

# Parallel Collision Search

- Proposed by van Oorschot & Wiener (1996).
- Distinguished points : a set of points having an easily testable property.
  ex. The x-coordinate has 3 trailling zero bits: 10101101000.
- Only distinguished points are stored in memory.
- $\theta$ - the proportion of distinguished points in a set S.



# Contributions

- We provided a more refined analysis of the running time of a parallel collision search for finding one or multiple collisions.

**Theorem. [T., Ionica, Dequen (2020)]**

Let $S$ be a set with $N$ elements and $f : S \to S$ a random map. We denote by $\theta$ the proportion of distinguished points in $S$. The expected running time to find $m$ collisions for $f$ with a memory constraint of $w$ words is:

$$\frac{1}{L}\left(\frac{w}{\theta} + (m - \frac{w^2}{2\theta^2 N})\frac{\theta N}{w} + \frac{2m}{\theta}\right).$$

# Contributions

- We replaced the classical hash table by a simple structure with lower memory requirements, that is inspired by radix trees (Packed Radix-Tree-List - PRTL).

| Collisions | Memory limit | Runtime | | Stored points | |
|---|---|---|---|---|---|
| | | PRTL | Hash table | PRTL | Hash table |
| 4,000,000 | 1 GB | 34.64 h | 58.80 h | 46,820,082 | 12,912,177 |
| 16,000,000 | 2 GB | 88.18 h | 137.46 h | 93,640,161 | 25,824,345 |
| 50,000,000 | 4 GB | 203.24 h | 276.80 h | 168,325,978 | 51,648,716 |

Table: Runtime for multi-collision search for a 55-bit curve using PRTLs and hash tables.

# Index Calculus

# Index calculus on binary elliptic curves

Let $\mathbb{F}_{2^n}$ be a finite field and $E$ be an elliptic curve defined by

$$E : y^2 + xy = x^3 + ax^2 + b$$

with $a, b \in \mathbb{F}_{2^n}$ and $n$ prime.

# Index calculus on binary elliptic curves

Let $\mathbb{F}_{2^n}$ be a finite field and $E$ be an elliptic curve defined by

$$E : y^2 + xy = x^3 + ax^2 + b$$

with $a, b \in \mathbb{F}_{2^n}$ and $n$ prime.

1. Choice of an appropriate factor base $B$
2. Point decomposition phase

   Find $P_1, \ldots, P_{m-1} \in B$, such that, for $R \in E(\mathbb{F}_{2^n})$

   $$R = P_1 + \ldots + P_{m-1}$$

3. Linear algebra

# Point Decomposition Problem (PDP)

## Semaev's summation polynomials (2004)

$$S_2(X_1, X_2) = X_1 + X_2,$$
$$S_3(X_1, X_2, X_3) = X_1^2X_2^2 + X_1^2X_3^2 + X_1X_2X_3 + X_2^2X_3^2 + b,$$

For $m \geq 4$

$$S_m(X_1, \ldots, X_m) =$$
$$Res_X(S_{m-k}(X_1, \ldots, X_{m-k-1}, X), S_{k+2}(X_{m-k}, \ldots, X_m, X))$$

# Point Decomposition Problem (PDP)

## Semaev's summation polynomials (2004)

$$S_2(X_1, X_2) = X_1 + X_2,$$
$$S_3(X_1, X_2, X_3) = X_1^2X_2^2 + X_1^2X_3^2 + X_1X_2X_3 + X_2^2X_3^2 + b,$$

For $m \geq 4$

$$S_m(X_1, \ldots, X_m) =$$
$$Res_X(S_{m-k}(X_1, \ldots, X_{m-k-1}, X), S_{k+2}(X_{m-k}, \ldots, X_m, X))$$

## Reducing the PDP to the problem of finding the roots of $S_m$

For $R, P_1, \ldots, P_{m-1} \in E(\mathbb{F}_{2^n})$

$$R + P_1 + \ldots + P_{m-1} = \mathcal{O} \iff S_m(\mathbf{x}_R, \mathbf{x}_{P_1}, \ldots, \mathbf{x}_{P_{m-1}}) = 0$$

# Index calculus on elliptic curves over extension fields

Gaudry (2008), Diem (2009)

**Choice of an appropriate factor base**

When $E$ is an elliptic curve defined over $\mathbb{F}_{q^n}$, with $n$ small, the factor base is the set of points whose $x$-coordinate lies in $\mathbb{F}_q$.

**Weil descent**

Rewrite the equation $S_{n+1}(\mathbf{x}_R, X_1, \ldots, X_n) = 0$ as a system of $n$ equations over $\mathbb{F}_q$.

# PDP algebraic model

Yun-Ju *et al.* (2013)

**Factor base for elliptic curves defined over $\mathbb{F}_{2^n}$, with $n$ prime**

An $l$-dimensional vector subspace $V$ of $\mathbb{F}_{2^n}/\mathbb{F}_2$. When $l \sim \frac{n}{m}$ the system has a reasonable chance to have a solution.

**$X_i$-variables**

$$X_1 = a_{1,0} + \ldots + a_{1,l-1}t^{l-1}$$
$$X_2 = a_{2,0} + \ldots + a_{2,l-1}t^{l-1}$$
$$\ldots$$
$$X_m = a_{m,0} + \ldots + a_{m,l-1}t^{l-1}$$

# Logical cryptanalysis with WDSat

# From the algebraic model to the SAT-reasoning model

Using SAT solvers as a cryptanalytic tool requires expressing the cryptographic problem as a Boolean formula in conjunctive normal form (CNF) - a conjunction ($\wedge$) of OR-clauses.

*Example.*

$$(\neg x_1 \vee x_2) \wedge$$
$$(\neg x_2 \vee x_4 \vee \neg x_5)) \wedge$$
$$(x_5 \vee x_6)$$

# From the algebraic model to the SAT-reasoning model

XOR-enabled SAT solvers are adapted to read a formula in CNF-XOR form - a conjunction ($\wedge$) of OR-clauses and XOR-clauses.

*Example.*

$$(\neg x_1 \vee x_2) \wedge$$
$$(\neg x_2 \vee x_4 \vee \neg x_5)) \wedge$$
$$(x_1 \oplus x_5 \oplus x_6)$$

# From the algebraic model to the CNF-XOR model

Variables in $\mathbb{F}_2$:

$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6$.

Propositional variables:

$x_1, x_2, x_3, x_4, x_5, x_6$ with truth values in $\{\text{TRUE}, \text{FALSE}\}$

$$\mathbf{x}_1 + \mathbf{x}_2 \cdot \mathbf{x}_4 + \mathbf{x}_5 \cdot \mathbf{x}_6 + 1 = 0$$
$$\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_4 + \mathbf{x}_5 + 1 = 0$$
$$\mathbf{x}_3 + \mathbf{x}_4 + \mathbf{x}_2 \cdot \mathbf{x}_4 = 0$$
$$\mathbf{x}_2 + \mathbf{x}_5 + \mathbf{x}_2 \cdot \mathbf{x}_4 + \mathbf{x}_5 \cdot \mathbf{x}_6 + 1 = 0$$
$$\mathbf{x}_3 + \mathbf{x}_4 + \mathbf{x}_6 + 1 = 0$$

$$(x_1 \oplus (x_2 \wedge x_4) \oplus (x_5 \wedge x_6)) \wedge$$
$$(x_1 \oplus x_2 \oplus x_4 \oplus x_5) \wedge$$
$$(x_3 \oplus x_4 \oplus (x_2 \wedge x_4) \oplus \top) \wedge$$
$$(x_2 \oplus x_5 \oplus (x_2 \wedge x_4) \oplus (x_5 \wedge x_6)) \wedge$$
$$(x_3 \oplus x_4 \oplus x_6)$$

Multiplication in $\mathbb{F}_2$ $(\cdot)$ becomes the logical AND operation $(\wedge)$ and addition in $\mathbb{F}_2$ $(+)$ becomes the logical XOR $(\oplus)$.

# From the algebraic model to the CNF-XOR model

Add new variable $x_{2,4}$ to substitute the conjunction $x_2 \wedge x_4$.

Transform the constraint

$$x_{2,4} \Leftrightarrow (x_2 \wedge x_4)$$

into CNF.

# From the algebraic model to the CNF-XOR model

Propositional variables:

$x_1$, $x_2$, $x_3$, $x_4$, $x_5$, $x_6$, $x_{2,4}$, $x_{5,6}$ with truth values in $\{\text{TRUE}, \text{FALSE}\}$

$$(x_1 \oplus (x_2 \wedge x_4) \oplus (x_5 \wedge x_6)) \wedge$$
$$(x_1 \oplus x_2 \oplus x_4 \oplus x_5) \wedge$$
$$(x_3 \oplus x_4 \oplus (x_2 \wedge x_4) \oplus \top) \wedge$$
$$(x_2 \oplus x_5 \oplus (x_2 \wedge x_4) \oplus (x_5 \wedge x_6)) \wedge$$
$$(x_3 \oplus x_4 \oplus x_6)$$

$$(\neg x_{2,4} \vee x_2) \wedge$$
$$(\neg x_{2,4} \vee x_4) \wedge$$
$$(\neg x_2 \vee \neg x_4 \vee x_{2,4}) \wedge$$
$$(\neg x_{5,6} \vee x_5) \wedge$$
$$(\neg x_{5,6} \vee x_6) \wedge$$
$$(\neg x_5 \vee \neg x_6 \vee x_{5,6}) \wedge$$
$$(x_1 \oplus x_{2,4} \oplus x_{5,6}) \wedge$$
$$(x_1 \oplus x_2 \oplus x_4 \oplus x_5) \wedge$$
$$(x_3 \oplus x_4 \oplus x_{2,4} \oplus \top) \wedge$$
$$(x_2 \oplus x_5 \oplus x_{2,4} \oplus x_{5,6}) \wedge$$
$$(x_3 \oplus x_4 \oplus x_6)$$

# WDSat algorithm

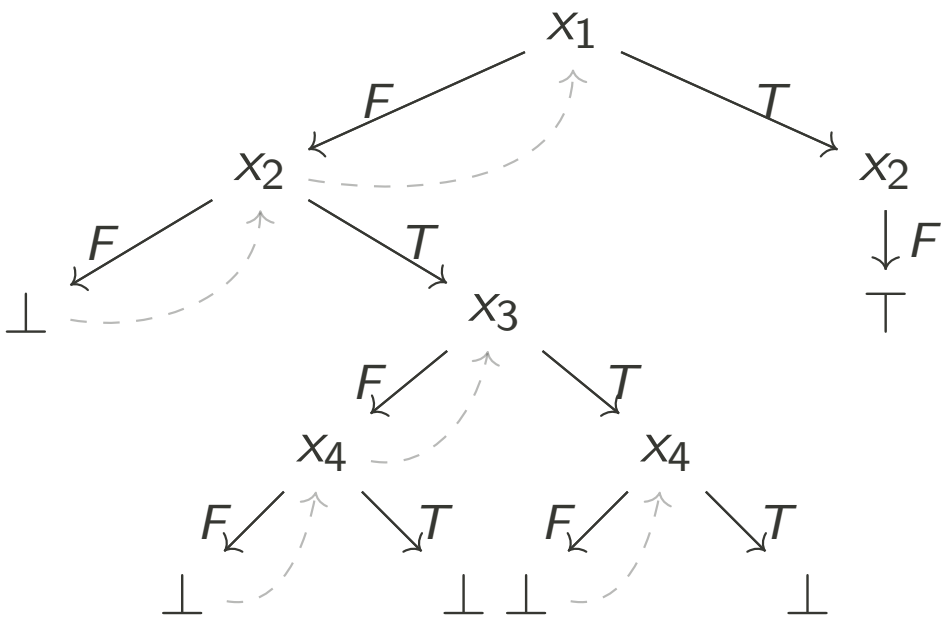
Based on the Davis-Putnam-Logemann-Loveland (DPLL) algorithm.

Building a binary search-tree of height equivalent (at worst) to the number of variables.

# WDSat algorithm

Based on the Davis-Putnam-Logemann-Loveland (DPLL) algorithm.

Building a binary search-tree of height equivalent (at worst) to the number of variables.



# WDSat - Three reasoning modules

## CNF module

Performs unit propagation on CNF-clauses.

## XORSET module

Performs unit propagation on the parity constraints. When all except one literal in a XOR clause is assigned, we infer the truth value of the last literal according to parity reasoning.

## XORGAUSS module

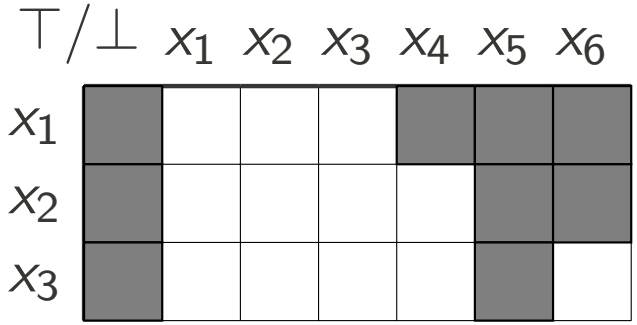
Performs Gaussian elimination on the XOR system.

# WDSat - XORGAUSS module

- All variables in an XOR-clause belong to the same equivalence class.
- We choose one literal from the equivalence class to be the representative.
- Property: a representative of an equivalence class will never be present in another equivalence class.

| XOR-clauses | Equivalence classes |
|---|---|
| $x_1 \oplus x_4 \oplus x_5 \oplus x_6$ | $x_1 \Leftrightarrow x_4 \oplus x_5 \oplus x_6 \oplus \top$ |
| $x_1 \oplus x_2 \oplus x_4 \oplus \top$ | $x_2 \Leftrightarrow x_5 \oplus x_6 \oplus \top$ |
| $x_2 \oplus x_3 \oplus x_6 \oplus \top$ | $x_3 \Leftrightarrow x_5 \oplus \top$ |

- Implementation: A compact *EC* structure.

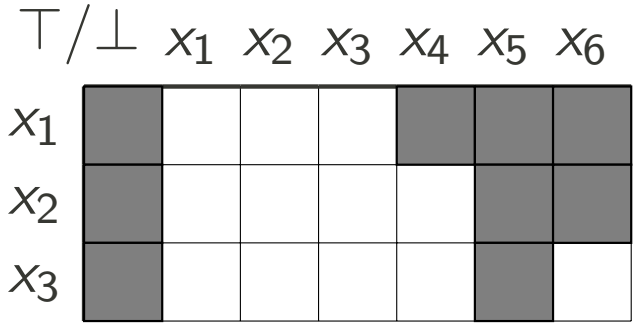


# WDSat - XORGAUSS module

- All variables in an XOR-clause belong to the same equivalence class.
- We choose one literal from the equivalence class to be the representative.
- Property: a representative of an equivalence class will never be present in another equivalence class.

| | XOR-clauses | Equivalence classes |
|---|---|---|
| | $x_1 \oplus x_4 \oplus x_5 \oplus x_6$ | $x_1 \Leftrightarrow x_4 \oplus x_5 \oplus x_6 \oplus \top$ |
| $x_2 \oplus x_5 \oplus x_6$ | ~~$x_1 \oplus x_2 \oplus x_4 \oplus \top$~~ | $x_2 \Leftrightarrow x_5 \oplus x_6 \oplus \top$ |
| | $x_2 \oplus x_3 \oplus x_6 \oplus \top$ | $x_3 \Leftrightarrow x_5 \oplus \top$ |

- Implementation: A compact *EC* structure.

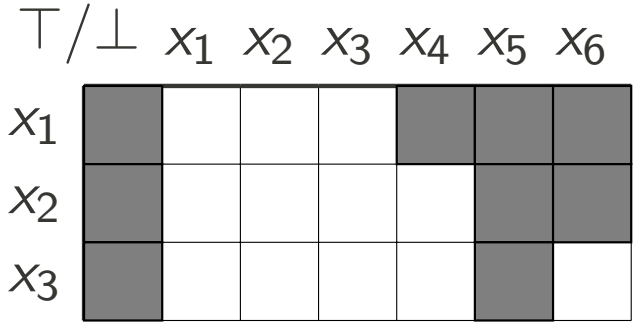


# WDSat - XORGAUSS module

- All variables in an XOR-clause belong to the same equivalence class.
- We choose one literal from the equivalence class to be the representative.
- Property: a representative of an equivalence class will never be present in another equivalence class.

| | XOR-clauses | Equivalence classes |
|---|---|---|
| | $x_1 \oplus x_4 \oplus x_5 \oplus x_6$ | $x_1 \Leftrightarrow x_4 \oplus x_5 \oplus x_6 \oplus \top$ |
| $x_2 \oplus x_5 \oplus x_6$ | ~~$x_1 \oplus x_2 \oplus x_4 \oplus \top$~~ | $x_2 \Leftrightarrow x_5 \oplus x_6 \oplus \top$ |
| $x_3 \oplus x_5$ | ~~$x_2 \oplus x_3 \oplus x_6 \oplus \top$~~ | $x_3 \Leftrightarrow x_5 \oplus \top$ |

- Implementation: A compact *EC* structure.



**Motivation**

The case where a possible cancellation of terms is overseen due to the CNF-XOR form.

Boolean polynomial system

$$\mathbf{x}_1 + \mathbf{x}_2\mathbf{x}_3 + \mathbf{x}_5 + \mathbf{x}_6 + 1 = 0$$
$$\mathbf{x}_3 + \mathbf{x}_5 + \mathbf{x}_6 = 0.$$

CNF-XOR formula

$$\neg x_{2,3} \vee x_2$$
$$\neg x_{2,3} \vee x_3$$
$$\neg x_2 \vee \neg x_3 \vee x_{2,3}$$
$$x_1 \oplus x_{2,3} \oplus x_5 \oplus x_6$$
$$x_3 \oplus x_5 \oplus x_6 \oplus \top.$$

**Motivation**

The case where a possible cancellation of terms is overseen due to the CNF-XOR form.

Boolean polynomial system

$$\mathbf{x}_1 + \mathbf{x}_2\mathbf{x}_3 + \mathbf{x}_5 + \mathbf{x}_6 + 1 = 0$$
$$\mathbf{x}_3 + \mathbf{x}_5 + \mathbf{x}_6 = 0.$$

CNF-XOR formula

$$\neg x_{2,3} \vee x_2$$
$$\neg x_{2,3} \vee x_3$$
$$\neg x_2 \vee \neg x_3 \vee x_{2,3}$$
$$x_1 \oplus x_{2,3} \oplus x_5 \oplus x_6$$
$$x_3 \oplus x_5 \oplus x_6 \oplus \top.$$

Set $\mathbf{x}_2$ to 1 / Set $x_2$ to $\top$

## Motivation

The case where a possible cancellation of terms is overseen due to the CNF-XOR form.

Boolean polynomial system

$$\mathbf{x}_1 + \mathbf{x}_2\mathbf{x}_3 + \mathbf{x}_5 + \mathbf{x}_6 + 1 = 0$$
$$\mathbf{x}_3 + \mathbf{x}_5 + \mathbf{x}_6 = 0.$$

CNF-XOR formula

$$\neg x_{2,3} \vee x_2$$
$$\neg x_{2,3} \vee x_3$$
$$\neg x_2 \vee \neg x_3 \vee x_{2,3}$$
$$x_1 \oplus x_{2,3} \oplus x_5 \oplus x_6$$
$$x_3 \oplus x_5 \oplus x_6 \oplus \top.$$

Set $\mathbf{x}_2$ to 1 / Set $x_2$ to $\top$

**Motivation**

The case where a possible cancellation of terms is overseen due to the CNF-XOR form.

Boolean polynomial system

$$\mathbf{x}_1 + \mathbf{x}_3 + \mathbf{x}_5 + \mathbf{x}_6 + 1 = 0$$
$$\mathbf{x}_3 + \mathbf{x}_5 + \mathbf{x}_6 = 0.$$

CNF-XOR formula

$$\neg x_{2,3} \vee x_3$$
$$\neg x_3 \vee x_{2,3}$$
$$x_1 \oplus x_{2,3} \oplus x_5 \oplus x_6$$
$$x_3 \oplus x_5 \oplus x_6 \oplus \top.$$

**Motivation**

The case where a possible cancellation of terms is overseen due to the CNF-XOR form.

Boolean polynomial system

$$\mathbf{x}_1 + \mathbf{x}_3 + \mathbf{x}_5 + \mathbf{x}_6 + 1 = 0$$
$$\mathbf{x}_3 + \mathbf{x}_5 + \mathbf{x}_6 = 0.$$

$$\mathbf{x}_1 = 1$$

CNF-XOR formula

$$\neg x_{2,3} \vee x_3$$
$$\neg x_3 \vee x_{2,3}$$
$$x_1 \oplus x_{2,3} \oplus x_5 \oplus x_6$$
$$x_3 \oplus x_5 \oplus x_6 \oplus \top.$$

**Define the following rule:**

$$\frac{x_1 \quad x_{1,2} \Leftrightarrow (x_1 \wedge x_2)}{x_{1,2} \Leftrightarrow x_2}.$$

If $x_1$ is set to TRUE, replace all occurrences of $x_{1,2}$ by $x_2$.

# Branching variables

Our DPLL-based algorithm only makes assignments on variables that are present in the initial Boolean polynomial system. Substitution variables are propagated as a consequence.

# Branching variables

Our DPLL-based algorithm only makes assignments on variables that are present in the initial Boolean polynomial system. Substitution variables are propagated as a consequence.

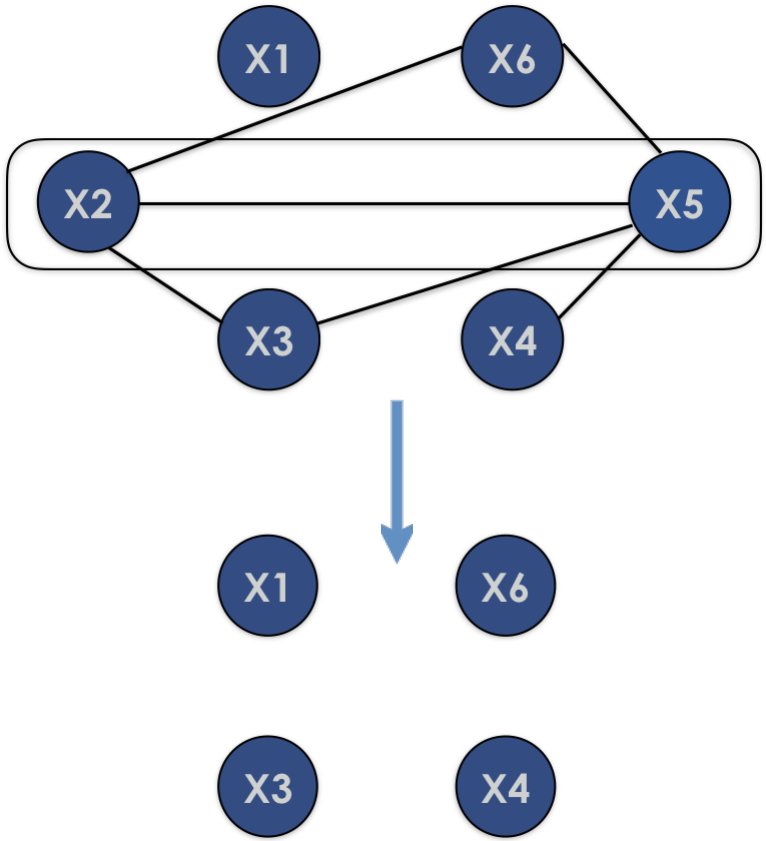
Order of branching variables?

# Branching variables

MVC preprocessing technique

$$x_1 + x_2x_3 + x_4 + x_4x_5 = 0$$
$$x_1 + x_2x_3 = 0$$
$$x_1 + x_3x_5 + x_6 = 0$$
$$x_1 + x_2x_5x_6 + x_6 = 0$$

$$x_1 + x_3 = 0$$
$$x_1 + x_3 + x_6 = 0$$
$$x_1 = 0.$$



Gaudry (2008)

## Symmetrization

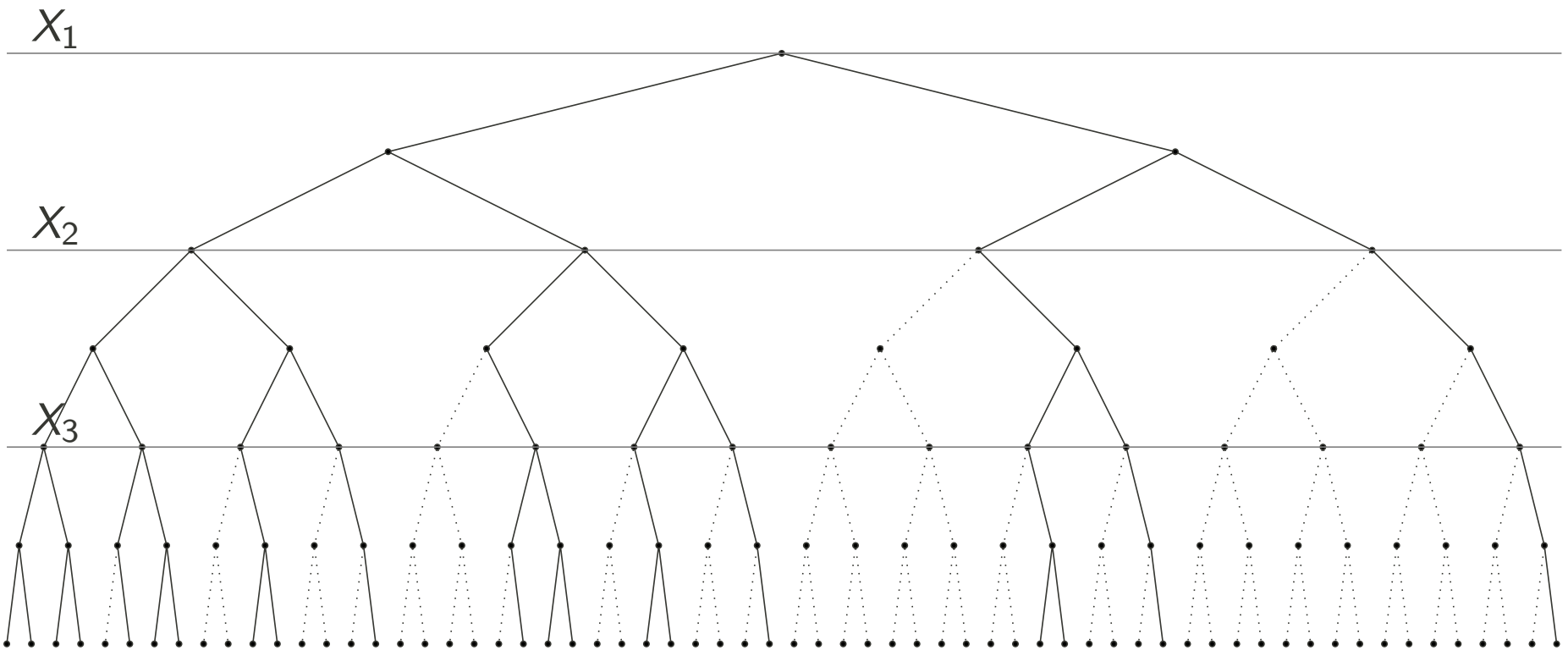
Rewrite $S_m$ in terms of the elementary symmetric polynomials

$$\begin{aligned} \mathbf{e}_1 &= \sum_{1 \le i_1 \le m} X_{i_1}, \\ \mathbf{e}_2 &= \sum_{1 \le i_1, i_2 \le m} X_{i_1} X_{i_2}, \\ &\dots \\ \mathbf{e}_m &= \prod_{1 \le i \le m} X_i. \end{aligned}$$

# WDSat - breaking symmetry

- Exploit the symmetry of Semaev's summation polynomials: when $X_1, ..., X_m$ is a solution, all permutations of this set are a solution as well.
- Establish the following constraint $X_1 \leq X_2 \leq \ldots \leq X_m$.
- Implement constraint in the solver using a tree-pruning-like technique.
- Optimize the complexity by a factor of $m!$.

# WDSat - breaking symmetry



# Experimental results

Third summation polynomial
$n = 42,\ l = 20$

| Solving approach | SAT | | UNSAT | |
|---|---|---|---|---|
| | Runtime (s) | #Conflicts | Runtime (s) | #Conflicts |
| Gröbner | 16.8 | $N/A$ | 18.7 | $N/A$ |
| MiniSat | $> 600$ | | $> 600$ | |
| Glucose | $> 600$ | | $> 600$ | |
| MapleLCMDistChronoBT | $> 600$ | | $> 600$ | |
| CaDiCaL | $> 600$ | | $> 600$ | |
| CryptoMiniSat | 29.0 | 226668 | 84.3 | 627539 |
| WDSat+XG-EXT+MVC | **4.2** | **27684** | **13.5** | **86152** |

Table: Comparing Gröbner basis and SAT-based approaches for solving the PDP. Running times are in seconds.

# Experimental results

Fourth summation polynomial
$n = 19,\ l = 6$

| Solving approach | SAT | | UNSAT | |
|---|---|---|---|---|
| | Runtime (s) | #Conflicts | Runtime (s) | #Conflicts |
| Gröbner | 229.3 | $N/A$ | 229.4 | $N/A$ |
| MiniSat | 239.7 | 1840190 | 517.0 | 3433304 |
| Glucose | 189.2 | 1527158 | 274.8 | 2056575 |
| MapleLCMDistChronoBT | 655.1 | 4035131 | 918.7 | 5378945 |
| CaDiCaL | 43.6 | 254194 | 141.3 | 629869 |
| CryptoMiniSat | 331.8 | 1791188 | 707.9 | 3416526 |
| WDSat+br-sym | **0.24** | **19166** | **0.63** | **44034** |

Table: Comparing Gröbner basis and SAT-based approaches for solving the PDP. Running times are in seconds.

# Conclusion

- When solving the PDP for prime degree extension fields of $\mathbb{F}_2$, Gröbner basis methods should be replaced with a SAT-based approach.
- Our CNF-XOR model with the dedicated SAT-solver, WDSAT, yields significantly faster running times than all other algebraic and SAT-based approaches.
- Extending the WDSAT solver with our symmetry breaking technique optimizes the resolution of the PDP by a factor of m!.

# Perspectives

- Understand the complexity of CNF-XOR instance solving.
- Combine WDSAT with CDCL techniques.
- Use WDSAT for attacks on other cryptosystems.
- Adapt our PCS implementation for meet-in-the-middle attacks on isogeny-based cryptosystems.

# Open source code

1. **Parity (XOR) Reasoning for the Index Calculus Attack**

   https://github.com/mtrimoska/WDSat

2. **A SAT-Based Approach for Index Calculus on Binary Elliptic Curves**

   https://github.com/mtrimoska/EC-Index-Calculus-Benchmarks

3. **Time-Memory Trade-offs for Parallel Collision Search Algorithms**

   https://github.com/mtrimoska/PCS