This page intentionally left blank

Efficient Normalized Reduction and Generation of Equivalent Multivariate Binary Polynomials

Arnau Gàmez-Montolio (City, University of London; Activision Research)
Enric Florit (Universitat de Barcelona)
Martin Brain (City, University of London)
Jacob M. Howe (City, University of London)

BAR 2024 (NDSS) @ San Diego, CA - March 1, 2024

Motivation

Use of binary (permutation) polynomials as *obfuscation primitives*.

Data encodings

Prevent data values used for arbitrary computations from being revealed during program execution.

```
uint32_t a, b, r;
...
a = key1;
b = key2;
r = foo(a*b);
```

```
uint32_t a, b, r;
...
a = key1;
b = key2;
r = foo(a*b);
```

$$P(x) = 1789355803x + 1391591831$$
$$Q(x) = 3537017619x + 624260299$$
$$P(Q(x)) = x$$

```
uint32_t a, b, r;
...
a = key1;
b = key2;
r = foo(a*b);
```

$$P(x) = 1789355803x + 1391591831$$

$$Q(x) = 3537017619x + 624260299$$

$$P(Q(x)) = x$$

$$a = P(key_1)$$

$$b = P(key_2)$$

$$r = foo(Q(a)Q(b))$$

```
uint32_t a, b, r;
...
a = key1;
b = key2;
r = foo(a*b);
```

$$P(x) = 1789355803x + 1391591831$$
$$Q(x) = 3537017619x + 624260299$$
$$P(Q(x)) = x$$

$$a = P(key_1)$$
$$b = P(key_2)$$
$$r = foo(Q(a)Q(b))$$

```
uint32_t a, b, r;
...
a = 1789355803*key1 + 1391591831;
b = 1789355803*key2 + 1391591831;
r = foo(4112253801*a*b + 1966380049*a + 1966380049*b + 1062639865);
```

Insertion of identities

Increase the syntactic complexity of an expression by *wrapping* it with the identity function generated by the composition of two inverse functions (e.g., binary permutation polynomials).

```
uint8_t x, y, z;
x = ...;
y = ...;
z = x + y;
```

```c
uint8_t x, y, z;
x = ...;
y = ...;
z = x + y;
```

$$P(x) = 8x^2 + 151x + 111$$
$$Q(x) = 200x^2 + 183x + 223$$
$$P(Q(x)) = x$$

```c
uint8_t x, y, z;
x = ...;
y = ...;
z = x + y;
```

$$P(x) = 8x^2 + 151x + 111$$
$$Q(x) = 200x^2 + 183x + 223$$
$$P(Q(x)) = x$$

$$z = P(Q(x + y))$$

```
uint8_t x, y, z;
x = ...;
y = ...;
z = x + y;
```

$$P(x) = 8x^2 + 151x + 111$$
$$Q(x) = 200x^2 + 183x + 223$$
$$P(Q(x)) = x$$

$$z = P(Q(x + y))$$

```
uint8_t x, y, z;
x = ...;
y = ...;
z = 8*(200*(x + y)*(x + y) + 183*(x + y) + 223)*(200*(x + y)*(x + y) + 183*(x + y) +
23) + 151*(200*(x + y)*(x + y) + 183*(x + y) + 223) + 111;
```

Opaque constants

Conceal (the use of) sensitive constants by replacing them with an expression on an arbitrary number of variables. This expression will always evaluate to the *opaqued* constant during runtime computation, regardless of the concrete values its variables are assigned to.

```
uint8_t k = 123;
foo(k);
```

```c
uint8_t k = 123;
foo(k);
```

$$E(x, y) = x - y + 2(\neg x \wedge y) - (x \oplus y)$$
$$E(x, y) \equiv 0$$

$$P(x) = 248x^2 + 97x$$
$$Q(x) = 136x^2 + 161x$$
$$P(Q(x)) = x$$

```c
uint8_t k = 123;
foo(k);
```

$$E(x, y) = x - y + 2(\neg x \wedge y) - (x \oplus y)$$
$$E(x, y) \equiv 0$$

$$P(x) = 248x^2 + 97x$$
$$Q(x) = 136x^2 + 161x$$
$$P(Q(x)) = x$$

$$k = P(E(x, y) + Q(k))$$

```
uint8_t k = 123;
foo(k);
```

$$E(x, y) = x - y + 2(\neg x \wedge y) - (x \oplus y)$$
$$E(x, y) \equiv 0$$

$$P(x) = 248x^2 + 97x$$
$$Q(x) = 136x^2 + 161x$$
$$P(Q(x)) = x$$

$$k = P(E(x, y) + Q(k))$$

```
uint8_t x, y;
x = ...;
y = ...;
foo(195 + 97*x + 159*y + 194*~(x | ~y) + 159*(x ^ y) + (163 + x + 255*y + 2*~(x | ~y)
+ 255*(x ^ y)) * (232 + 248*x + 8*y + 240*~(x | ~y) + 8*(x ^ y)));
```

Goal

Obfuscation

Generation of equivalent binary polynomials.

- Increase diversity: different versions of the code with equivalent semantics.
- Introduce arbitrary algebraic complexity: thwart SMT solving capabilities, (de)compiler optimizations, etc.

Analysis

Reduce any binary polynomial to a unique (normalized) equivalent representative with guaranteed lowest degree.

- Deobfuscation efforts.
- Algebraic manipulation.

Null polynomials modulo 2^w

Idea

Idea

If you have an arbitrary polynomial $P_1(x)$, and a polynomial $Z(x)$ that is semantically the zero constant function, then

$$P_2(x) = P_1(x) + Z(x)$$

is a polynomial that is equivalent (as a function) to $P_1(x)$.

$$P_1(x) = 97x + 248x^2 \ (mod\ 2^8)$$
$$Z(x) = 128x + 128x^2 \ (mod\ 2^8)$$

$$P_1(x) = 97x + 248x^2 \ (mod\ 2^8)$$
$$Z(x) = 128x + 128x^2 \ (mod\ 2^8)$$

$$\begin{aligned} P_2(x) &= P_1(x) + Z(x) \\ &= 97x + 248x^2 + 128x + 128x^2 \ (mod\ 2^8) \\ &= 120x^2 + 225x \ (mod\ 2^8) \end{aligned}$$

$$P_1(x) = 97x + 248x^2 \ (mod\ 2^8)$$
$$Z(x) = 128x + 128x^2 \ (mod\ 2^8)$$

$$\begin{aligned} P_2(x) &= P_1(x) + Z(x) \\ &= 97x + 248x^2 + 128x + 128x^2 \ (mod\ 2^8) \\ &= 120x^2 + 225x \ (mod\ 2^8) \end{aligned}$$

$$P_1(x) \equiv P_2(x)$$

In other words, given two equivalent binary polynomials, its difference is a polynomial semantically equivalent to the zero constant function.

In other words, given two equivalent binary polynomials, its difference is a polynomial semantically equivalent to the zero constant function.

$$P_1(x) \equiv P_2(x)$$
$$Z(x) := P_1(x) - P_2(x)$$
$$Z(x) \equiv 0$$

In other words, given two equivalent binary polynomials, its difference is a polynomial semantically equivalent to the zero constant function.

$$
\begin{aligned}
P_1(x) &\equiv P_2(x) \\
Z(x) &:= P_1(x) - P_2(x) \\
Z(x) &\equiv 0
\end{aligned}
$$

We call these $Z(x)$ **null polynomials**.

Generate null polynomials

Given a positive integer m and a bitsize w, we define the polynomial

$$G_m(x) = 2^{\max\{0, w - \nu_2(m!)\}} \prod_{i=0}^{m-1} (x - i)$$

Generate null polynomials

Given a positive integer m and a bitsize w, we define the polynomial

$$G_m(x) = 2^{\max\{0, w - \nu_2(m!)\}} \prod_{i=0}^{m-1} (x - i)$$

$$\begin{aligned}
G_2 &= 2^{\max\{0, 8 - \nu_2(2!)\}} \prod_{i=0}^{2-1} (x - i) \\
&= 2^{\max\{0,7\}} x(x - 1) \\
&= 2^7 x(x - 1) \\
&= 128x + 128x^2
\end{aligned}$$

Generate null polynomials

Given a positive integer m and a bitsize w, we define the polynomial

$$G_m(x) = 2^{\max\{0, w - v_2(m!)\}} \prod_{i=0}^{m-1} (x - i)$$

$$\begin{aligned} G_2 &= 2^{\max\{0, 8 - v_2(2!)\}} \prod_{i=0}^{2-1} (x - i) \\ &= 2^{\max\{0,7\}} x(x-1) \\ &= 2^7 x(x-1) \\ &= 128x + 128x^2 \end{aligned}$$

$$\begin{aligned} G_4 &= 2^{\max\{0, 8 - v_2(4!)\}} \prod_{i=0}^{4-1} (x - i) \\ &= 2^{\max\{0,5\}} x(x-1)(x-2)(x-3) \\ &= 2^5 x(x-1)(x-2)(x-3) \\ &= 64x + 96x^2 + 64x^3 + 32x^4 \end{aligned}$$

$$\forall j, G_j(x) \equiv 0$$

The set of polynomials

$$\{G_2\,,\, G_4\,,\, \ldots\,,\, G_{d_w}\,\}$$

generates the whole set of *null polynomials*, and is minimal.

The set of polynomials

$$\{G_2, G_4, \dots, G_{d_w}\}$$

generates the whole set of *null polynomials*, and is minimal.

Here, d_w is the least positive integer k such that 2^w divides $k!$, in other words:

$$d_w \text{ is the minimum positive integer such that } \nu_2(d_w!) \geq w$$

The set of polynomials

$$\{G_2, G_4, \ldots, G_{d_w}\}$$

generates the whole set of *null polynomials*, and is minimal.

Here, d_w is the least positive integer k such that 2^w divides $k!$, in other words:

$$d_w \text{ is the minimum positive integer such that } \nu_2(d_w!) \geq w$$

$$\nu_2(8!) = \nu_2(2^7 \cdot 3^2 \cdot 5 \cdot 7) = 7 \ldots$$
$$\nu_2(10!) = \nu_2(2^8 \cdot 3^4 \cdot 5^2 \cdot 7) = 8 \checkmark$$

Generates: Any null polynomial can be obtained by multiplying and adding the elements of $\{G_2, G_4, \ldots, G_{d_w}\}$ with any other polynomials (including constants).

Generates: Any null polynomial can be obtained by multiplying and adding the elements of $\{G_2, G_4, \ldots, G_{d_w}\}$ with any other polynomials (including constants).

Minimal: If we remove any of the $G_j \in \{G_2, G_4, \ldots, G_{d_w}\}$, we could not generate the whole set.

Factorial basis

Think of the polynomials as elements of a vector space generated by the *canonical basis*:

$$\{1, x, x^2, x^3, \ldots\}$$

Make a change of basis from the *canonical basis* into the *factorial basis*:

$$\{1, x, x^{(2)}, x^{(3)}, x^{(4)}, \ldots\}$$

Where $x^{(j)}$ is defined as:

$$\begin{aligned}
x^{(0)} &= 1 \\
x^{(1)} &= x \\
x^{(2)} &= x(x-1) \\
x^{(3)} &= x(x-1)(x-2) \\
\ldots \\
x^{(j)} &= x(x-1)(x-2)\ldots(x-j+1)
\end{aligned}$$

For instance, in 8 bits we have:

$G_2 = (0, 128, 128)$ in canonical basis, i.e. $G_2 = (0 \cdot 1) + (128 \cdot x) + (128 \cdot x^2)$

$G_2 = (0, 0, 128)$ in factorial basis, i.e. $G_2 = (0 \cdot 1) + (0 \cdot x) + (128 \cdot x(x-1))$

For instance, in 8 bits we have:

$G_2 = (0, 128, 128)$ in canonical basis, i.e. $G_2 = (0 \cdot 1) + (128 \cdot x) + (128 \cdot x^2)$

$G_2 = (0, 0, 128)$ in factorial basis, i.e. $G_2 = (0 \cdot 1) + (0 \cdot x) + (128 \cdot x(x-1))$

$G_4 = (0, 64, 96, 64, 32)$ in canonical basis.

$G_4 = (0, 0, 0, 0, 32)$ in factorial basis.

For instance, in 8 bits we have:

$G_2 = (0, 128, 128)$ in canonical basis, i.e. $G_2 = (0 \cdot 1) + (128 \cdot x) + (128 \cdot x^2)$

$G_2 = (0, 0, 128)$ in factorial basis, i.e. $G_2 = (0 \cdot 1) + (0 \cdot x) + (128 \cdot x(x-1))$

$G_4 = (0, 64, 96, 64, 32)$ in canonical basis.

$G_4 = (0, 0, 0, 0, 32)$ in factorial basis.

$\dots$

By construction, G_j in the factorial basis is the single coefficient $c_j = 2^{\max\{0, w - \nu_2(j!)\}}$, which *collapses* to $2^0 = 1$ for $j \geq d_w$. Thus:

By construction, G_j in the factorial basis is the single coefficient $c_j = 2^{\max\{0, w - \nu_2(j!)\}}$, which *collapses* to $2^0 = 1$ for $j \geq d_w$. Thus:

- "Adding an arbitrary null polynomial" translates to adding an arbitrary multiple of each c_j to each j^{th} coefficient of the initial polynomial expressed in the factorial basis.

Algorithms and examples

Algorithm

Generate equivalent binary polynomials

1. Take a binary polynomial $P(x)$
2. Represent its coefficients in factorial basis
3. Add an arbitrary multiple of c_j to each j^{th} coefficient in this form
4. Represent the new polynomial $P_{eq}(x)$ back in canonical basis (if needed)

In 8 bits, start with the polynomial $P(x) = 97x + 248x^2$, which corresponds to $(0, 97, 248)$ in canonical basis.

In factorial basis, $P(x)$ is represented as $(0, 89, 248)$.

In 8 bits, start with the polynomial $P(x) = 97x + 248x^2$, which corresponds to $(0, 97, 248)$ in canonical basis.

In factorial basis, $P(x)$ is represented as $(0, 89, 248)$.

Add arbitrary multiples of c_j to the j^{th} coefficient in this form.

$$(0, 89, 248 + c_2, c_3, 3c_4, 0, 0, 2c_7, 0, 0, 0, 0, 0, 0, 0, 0, 27c_{16})$$

In 8 bits, start with the polynomial $P(x) = 97x + 248x^2$, which corresponds to $(0, 97, 248)$ in canonical basis.

In factorial basis, $P(x)$ is represented as $(0, 89, 248)$.

Add arbitrary multiples of c_j to the j^{th} coefficient in this form.

$$(0, 89, 248 + c_2, c_3, 3c_4, 0, 0, 2c_7, 0, 0, 0, 0, 0, 0, 0, 0, 27c_{16})$$

$$(0, 89, 248 + 128, 128, 3 \cdot 32, 0, 0, 2 \cdot 16, 0, 0, 0, 0, 0, 0, 0, 0, 27 \cdot 1)$$

In 8 bits, start with the polynomial $P(x) = 97x + 248x^2$, which corresponds to $(0, 97, 248)$ in canonical basis.

In factorial basis, $P(x)$ is represented as $(0, 89, 248)$.

Add arbitrary multiples of c_j to the j^{th} coefficient in this form.

$$(0, 89, 248 + c_2, c_3, 3c_4, 0, 0, 2c_7, 0, 0, 0, 0, 0, 0, 0, 0, 27c_{16})$$

$$(0, 89, 248 + 128, 128, 3 \cdot 32, 0, 0, 2 \cdot 16, 0, 0, 0, 0, 0, 0, 0, 0, 27 \cdot 1)$$

$$(0, 89, 120, 128, 96, 0, 0, 32, 0, 0, 0, 0, 0, 0, 0, 0, 27)$$

In factorial basis, we have the equivalent polynomial

$$(0, 89, 120, 128, 96, 0, 0, 32, 0, 0, 0, 0, 0, 0, 0, 0, 27)$$

which corresponds to the following one in canonical basis:

$$(0, 161, 152, 192, 48, 32, 248, 184, 11, 96, 148, 80, 2, 160, 252, 88, 27)$$

In factorial basis, we have the equivalent polynomial

$$(0, 89, 120, 128, 96, 0, 0, 32, 0, 0, 0, 0, 0, 0, 0, 0, 27)$$

which corresponds to the following one in canonical basis:

$$(0, 161, 152, 192, 48, 32, 248, 184, 11, 96, 148, 80, 2, 160, 252, 88, 27)$$

$$\begin{aligned} P_{eq}(x) &= 89x + 120x^{(2)} + 128x^{(3)} + 96x^{(4)} + 32x^{(7)} + 27x^{(16)} \\ &= 161x + 152x^2 + 192x^3 + 48x^4 + 32x^5 + 248x^6 + 184x^7 \\ &+ 11x^8 + 96x^9 + 148x^{10} + 80x^{11} + 2x^{12} + 160x^{13} \\ &+ 252x^{14} + 88x^{15} + 27x^{16} \end{aligned}$$

$$P(x) = 97x + 248x^2$$

$$\equiv$$

$$\begin{aligned} P_{eq}(x) &= 89x + 120x^{(2)} + 128x^{(3)} + 96x^{(4)} + 32x^{(7)} + 27x^{(16)} \\ &= 161x + 152x^2 + 192x^3 + 48x^4 + 32x^5 + 248x^6 + 184x^7 \\ &+ 11x^8 + 96x^9 + 148x^{10} + 80x^{11} + 2x^{12} + 160x^{13} \\ &+ 252x^{14} + 88x^{15} + 27x^{16} \end{aligned}$$

Algorithm

Reduce binary polynomials to a unique normalized equivalent representative

1. Take a binary polynomial $P(x)$
2. Represent its coefficients in factorial basis
3. Reduce the j^{th} coefficient modulo c_j in this form
4. Represent the new polynomial $P_{red}(x)$ back in canonical basis (if needed)

In 8 bits, we have the polynomial

$$\begin{aligned} P(x) = {} & 193x + 112x^2 + 67x^3 + 143x^4 + 160x^5 \\ & + 19x^6 + 132x^7 + 176x^8 + 174x^9 + 130x^{10} \\ & + 170x^{11} + 188x^{12} + 91x^{13} + 140x^{14} \end{aligned}$$

which corresponds in canonical basis to

$$(0, 193, 112, 67, 143, 160, 19, 132, 176, 174, 130, 170, 188, 91, 140)$$

In factorial basis, $P(x)$ is represented as

$$(0, 103, 30, 166, 162, 72, 51, 166, 60, 172)$$

In factorial basis, $P(x)$ is represented as

$$(0, 103, 30, 166, 162, 72, 51, 166, 60, 172)$$

We reduce the j^{th} coefficient modulo c_j.

The associated vector of c_j values is

$$(256, 256, 128, 128, 32, 32, 16, 16, 2, 2)$$

Thus, doing the modulo element-wise we get the reduced coefficients in factorial basis

$$(0, 103, 30, 38, 2, 8, 3, 6, 0, 0)$$

Represent the polynomial back in canonical basis

$$(0, 193, 16, 159, 119, 245, 133, 6)$$

Represent the polynomial back in canonical basis

$$(0, 193, 16, 159, 119, 245, 133, 6)$$

$$P_{red}(x) = 193x + 16x^2 + 159x^3 + 119x^4 + 245x^5 + 133x^6 + 6x^7$$

$$
\begin{aligned}
P(x) = {} & 193x + 112x^2 + 67x^3 + 143x^4 + 160x^5 \\
& + 19x^6 + 132x^7 + 176x^8 + 174x^9 + 130x^{10} \\
& + 170x^{11} + 188x^{12} + 91x^{13} + 140x^{14}
\end{aligned}
$$

$$\equiv$$

$$
\begin{aligned}
P_{red}(x) = {} & 193x + 16x^2 + 159x^3 + 119x^4 \\
& + 245x^5 + 133x^6 + 6x^7
\end{aligned}
$$

Interesting facts

Each coefficient manipulation in factorial basis (either of generation of equivalent or reduction) is independent of the rest.

Each coefficient manipulation in factorial basis (either of generation of equivalent or reduction) is independent of the rest.

In other words, the manipulation of coefficients in factorial basis is **parallelizable**.

The degree of the reduced form of a polynomial in factorial basis is preserved in canonical basis.

The degree of the reduced form of a polynomial in factorial basis is preserved in canonical basis.

Thus, it makes sense to talk about *reduced* polynomials in a general sense.

If w is a power of 2 (i.e., $w \in \{8, 16, 32, 64, \dots\}$), then $d_w = w + 2$.

Thus, any binary polynomial in w-bits has an equivalent one of degree at most $w + 1$.

- By definition the leading coefficient of G_j is 1 for $j \geq d_w$.
- If we have a polynomial of degree $\geq d_w$ we can get rid of the coefficients of degree $\geq d_w$ by subtracting an arbitrary multiple of the associated G_j.

If w is a power of 2 (i.e., $w \in \{8, 16, 32, 64, \dots\}$), then $d_w = w + 2$.

Thus, any binary polynomial in w-bits has an equivalent one of degree at most $w + 1$.

- By definition the leading coefficient of G_j is 1 for $j \geq d_w$.
- If we have a polynomial of degree $\geq d_w$ we can get rid of the coefficients of degree $\geq d_w$ by subtracting an arbitrary multiple of the associated G_j.

This implies, from a reduction standpoint:

- We do not need to compute coefficients starting from $w + 2$ on the factorial form.

Multivariate

All the results and algorithms can be generalized to the multivariate case.

All the results and algorithms can be generalized to the multivariate case.

Notation gets a bit tricky, but it follows from a rather natural generalization of the concepts of null polynomials and factorial basis for the multivariate case.

Idea

Idea

$$\begin{aligned}
G_{(2,2)}(x, y) &= 2^{\max(8-(\nu_2(2!)+\nu_2(2!)),0)}\, x^{(2)} y^{(2)} \\
&= 2^6\, x(x-1)y(y-1) \\
&= 64x^2y^2 - 64xy^2 - 64x^2y + 64xy
\end{aligned}$$

Idea

$$
\begin{aligned}
G_{(2,2)}(x, y) &= 2^{\max(8-(\nu_2(2!)+\nu_2(2!)),0)}\, x^{(2)} y^{(2)} \\
&= 2^6\, x(x-1)y(y-1) \\
&= 64x^2y^2 - 64xy^2 - 64x^2y + 64xy
\end{aligned}
$$

Canonical basis: $\{1, x, y, x^2, xy, y^2, x^3, x^2y, xy^2, y^3, \dots\}$

Factorial basis: $\{1, x, y, x^{(2)}, xy, y^{(2)}, x^{(3)}, x^{(2)}y, xy^{(2)}, y^{(3)}, \dots\}$

In 8 bits, we have the polynomial

$$
\begin{aligned}
P(x, y) = {} & x^3y^8 + 228x^3y^7 + 253x^2y^8 + 66x^3y^6 + 84x^2y^7 \\
& + 2xy^8 + 4x^4y^4 + 88x^3y^5 + 58x^2y^6 + 200xy^7 \\
& + 232x^4y^3 + 89x^3y^4 + 248x^2y^5 + 132xy^6 + 44x^4y^2 \\
& + 68x^3y^3 + 217x^2y^4 + 176xy^5 + 232x^4y + 4x^3y^2 \\
& + 220x^2y^3 + 202xy^4 + 224x^3y + 193x^2y^2 + 248xy^3 \\
& + 8x^2y + 16xy^2 + 48xy
\end{aligned}
$$

In factorial basis, we have

$$P(x, y) = x^{(1)}y^{(1)} + x^{(2)}y^{(1)} + x^{(1)}y^{(2)} + x^{(2)}y^{(2)} + 4x^{(4)}y^{(4)} + x^{(3)}y^{(8)}$$

In factorial basis, we have

$$P(x,y) = x^{(1)}y^{(1)} + x^{(2)}y^{(1)} + x^{(1)}y^{(2)} + x^{(2)}y^{(2)} + 4x^{(4)}y^{(4)} + x^{(3)}y^{(8)}$$

To perform the reduction of coefficients, we find the coefficients

$$\begin{aligned}
c_{(1,1)} &= 2^{8-\nu_2(1!)-\nu_2(1!)} = 2^8 = 256 \\
c_{(2,1)} &= c_{(1,2)} = 2^{8-\nu_2(2!)-\nu_2(1!)} = 2^7 = 128 \\
c_{(2,2)} &= 2^{8-\nu_2(2!)-\nu_2(2!)} = 2^6 = 64 \\
c_{(4,4)} &= 2^{8-\nu_2(4!)-\nu_2(4!)} = 2^2 = 4 \\
c_{(3,8)} &= 2^{8-\nu_2(3!)-\nu_2(8!)} = 2^0 = 1.
\end{aligned}$$

After reduction, we get in factorial basis

$$\tilde{P}(x, y) = x^{(1)}y^{(1)} + x^{(2)}y^{(1)} + x^{(1)}y^{(2)} + x^{(2)}y^{(2)}$$

After reduction, we get in factorial basis

$$\tilde{P}(x, y) = x^{(1)}y^{(1)} + x^{(2)}y^{(1)} + x^{(1)}y^{(2)} + x^{(2)}y^{(2)}$$

Represent the polynomial back in canonical basis

$$\tilde{P}(x, y) = x^2y^2$$

$$
\begin{aligned}
P(x,y) = {} & x^3y^8 + 228x^3y^7 + 253x^2y^8 + 66x^3y^6 + 84x^2y^7 \\
& + 2xy^8 + 4x^4y^4 + 88x^3y^5 + 58x^2y^6 + 200xy^7 \\
& + 232x^4y^3 + 89x^3y^4 + 248x^2y^5 + 132xy^6 + 44x^4y^2 \\
& + 68x^3y^3 + 217x^2y^4 + 176xy^5 + 232x^4y + 4x^3y^2 \\
& + 220x^2y^3 + 202xy^4 + 224x^3y + 193x^2y^2 + 248xy^3 \\
& + 8x^2y + 16xy^2 + 48xy
\end{aligned}
$$

$$\equiv$$

$$\tilde{P}(x,y) = x^2y^2$$

Conclusion

Conclusion

- Thorough mathematical study of binary polynomials

Conclusion

- Thorough mathematical study of binary polynomials
- Characterization of equivalent binary polynomials

Conclusion

- Thorough mathematical study of binary polynomials
- Characterization of equivalent binary polynomials
- Explicit algorithms:
    - Generate arbitrary equivalent polynomials
    - Reduce binary polynomials to a normalized lowest-degree form

Conclusion

- Thorough mathematical study of binary polynomials
- Characterization of equivalent binary polynomials
- Explicit algorithms:
  - Generate arbitrary equivalent polynomials
  - Reduce binary polynomials to a normalized lowest-degree form

Arnau.Gamez-Montolio@city.ac.uk https://arnaugamez.com