

Autoguess

A Tool for Finding Guess-and-Determine Attacks and Key Bridges

Hosein Hadipour and Maria Eichlseder

ACNS 2022 - Rome, Italy

Outline

- 1 Guess-and-Determine (GD)
- 2 Constraint Programming Model for GD
- 3 Autoguess
- 4 Key-Bridging (KB)
- 5 Conclusion

Guess-and-Determine



Guess-and-Determine (GD)

Guess-and-Determine

Given a set of variables and a set of relations between them, find the smallest subset of variables guessing the value of which uniquely determines the value of the remaining variables.

Guess-and-Determine (GD)

Guess-and-Determine

Given a set of variables and a set of relations between them, find the smallest subset of variables guessing the value of which uniquely determines the value of the remaining variables.

Example

✓ $u, \dots, z \in \mathbb{F}_2^{32}$

✓ F, G, H : bijective functions

✓ c_1, \dots, c_5 : constants

$$\left\{ \begin{array}{lcl} F(u + v) \oplus G(x) \oplus y \oplus (z \lll 7) & = & c_1 \\ G(u \oplus w) + (y \lll 3) + z & = & c_2 \\ F(w \oplus x) + y \oplus z & = & c_3 \\ F(u) \oplus G(w + z) & = & c_4 \\ (F(u) \times G(w \lll 7)) + H(z \oplus v) & = & c_5 \end{array} \right.$$

Guess-and-Determine (GD)

Guess-and-Determine

Given a set of variables and a set of relations between them, find the smallest subset of variables guessing the value of which uniquely determines the value of the remaining variables.

Example

✓ Guess w, z

✓ Determine $u(4), y(2)$

✓ Determine $x(3), v(5)$

$$\left\{ \begin{array}{lcl} F(u + v) \oplus G(x) \oplus y \oplus (z \lll 7) & = & c_1 \\ G(u \oplus w) + (y \lll 3) + z & = & c_2 \\ F(w \oplus x) + y \oplus z & = & c_3 \\ F(u) \oplus G(w + z) & = & c_4 \\ (F(u) \times G(w \lll 7)) + H(z \oplus v) & = & c_5 \end{array} \right.$$

Symmetric and Implication Relations

Assumption: Relations are symmetric or implication

✓ Implication relations:

$$x_1, \dots, x_n \Rightarrow y$$

✓ Symmetric relations:

$$[x_1, \dots, x_n]$$

Example

Assume that $x, y, z, k \in \mathbb{F}_2^{32}$, and $F : \mathbb{F}_2^{32} \rightarrow \mathbb{F}_2^{32}$ is bijective:

$$z = x \times y$$

$$x, y \Rightarrow z$$

$$z = F(x + k) \oplus y$$

$$[x, y, z, k]$$

Symmetric and Implication Relations

Assumption: Relations are symmetric or implication

✓ **Implication relations:**

$$x_1, \dots, x_n \Rightarrow y$$

✓ **Symmetric relations:**

$$[x_1, \dots, x_n]$$

Example

Assume that $x, y, z, k \in \mathbb{F}_2^{32}$, and $F : \mathbb{F}_2^{32} \rightarrow \mathbb{F}_2^{32}$ is bijective:

$$z = x \times y$$

$$x, y \Rightarrow z$$

$$z = F(x + k) \oplus y$$

$$[x, y, z, k]$$

Symmetric and Implication Relations

Assumption: Relations are symmetric or implication

✓ **Implication relations:**

$$x_1, \dots, x_n \Rightarrow y$$

✓ **Symmetric relations:**

$$[x_1, \dots, x_n]$$

Example

Assume that $x, y, z, k \in \mathbb{F}_2^{32}$, and $F : \mathbb{F}_2^{32} \rightarrow \mathbb{F}_2^{32}$ is bijective:

$$z = x \times y$$

$$x, y \Rightarrow z$$

$$z = F(x + k) \oplus y$$

$$[x, y, z, k]$$

Symmetric and Implication Relations

Assumption: Relations are symmetric or implication

✓ **Implication relations:**

$$x_1, \dots, x_n \Rightarrow y$$

✓ **Symmetric relations:**

$$[x_1, \dots, x_n]$$

Example

Assume that $x, y, z, k \in \mathbb{F}_2^{32}$, and $F : \mathbb{F}_2^{32} \rightarrow \mathbb{F}_2^{32}$ is bijective:

$$z = x \times y$$

$$x, y \Rightarrow z$$

$$z = F(x + k) \oplus y$$

$$[x, y, z, k]$$

Symmetric and Implication Relations

Assumption: Relations are symmetric or implication

✓ **Implication relations:**

$$x_1, \dots, x_n \Rightarrow y$$

✓ **Symmetric relations:**

$$[x_1, \dots, x_n]$$

Example

Assume that $x, y, z, k \in \mathbb{F}_2^{32}$, and $F : \mathbb{F}_2^{32} \rightarrow \mathbb{F}_2^{32}$ is bijective:

$$z = x \times y$$

$$x, y \Rightarrow z$$

$$z = F(x + k) \oplus y$$

$$[x, y, z, k]$$

Symmetric and Implication Relations

Assumption: Relations are symmetric or implication

✓ **Implication relations:**

$$x_1, \dots, x_n \Rightarrow y$$

✓ **Symmetric relations:**

$$[x_1, \dots, x_n]$$

Example

Assume that $x, y, z, k \in \mathbb{F}_2^{32}$, and $F : \mathbb{F}_2^{32} \rightarrow \mathbb{F}_2^{32}$ is bijective:

$$z = x \times y$$

$$x, y \Rightarrow z$$

$$z = F(x + k) \oplus y$$

$$[x, y, z, k]$$

System of Equations

$$E : \begin{cases} e_1 : F(u + v) \oplus G(x) \oplus y \oplus (z \lll 7) & = c_1 \\ e_2 : G(u \oplus w) + (y \lll 3) + z & = c_2 \\ e_3 : F(w \oplus x) + y \oplus z & = c_3 \\ e_4 : F(u) \oplus G(w + z) & = c_4 \\ e_5 : (F(u) \times G(w \lll 7)) + H(z \oplus v) & = c_5 \end{cases}$$
$$X = \{u, v, w, x, y, z\}, E = \{e_1, \dots, e_5\}$$

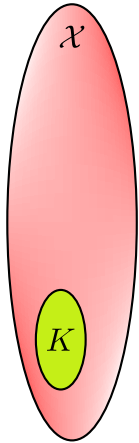
System of Equations \Rightarrow System of Relations

$$E : \begin{cases} e_1 : F(u + v) \oplus G(x) \oplus y \oplus (z \lll 7) & = c_1 \\ e_2 : G(u \oplus w) + (y \lll 3) + z & = c_2 \\ e_3 : F(w \oplus x) + y \oplus z & = c_3 \\ e_4 : F(u) \oplus G(w + z) & = c_4 \\ e_5 : (F(u) \times G(w \lll 7)) + H(z \oplus v) & = c_5 \end{cases}$$
$$X = \{u, v, w, x, y, z\}, E = \{e_1, \dots, e_5\}$$

$$\mathcal{R} : \begin{cases} r_1 : [u, v, x, y, z], & r_2 : [u, w, y, z] \\ r_3 : [w, x, y, z], & r_4 : [u, w, z] \\ r_5 : u, w \Rightarrow t, & r_6 : [t, z, v] \end{cases}$$
$$\mathcal{X} = \{u, v, w, x, y, z, t\}, \mathcal{R} = \{r_1, \dots, r_6\}$$

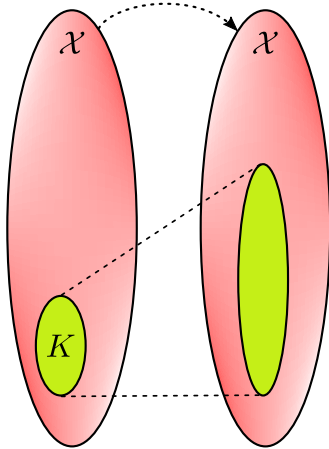
Knowledge Propagation

$(\mathcal{X}, \mathcal{K})$: System of relations, $K \subseteq \mathcal{X}$



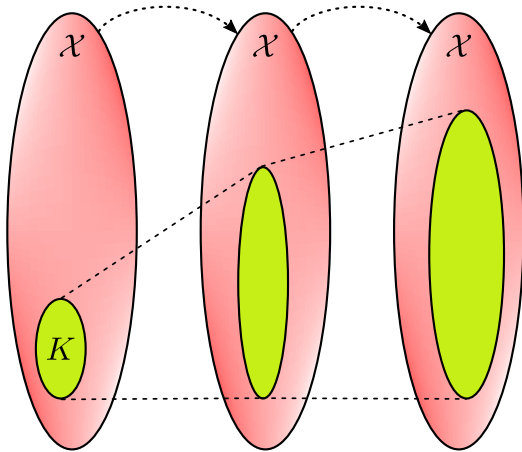
Knowledge Propagation

$(\mathcal{X}, \mathcal{R})$: System of relations, $K \subseteq \mathcal{X}$



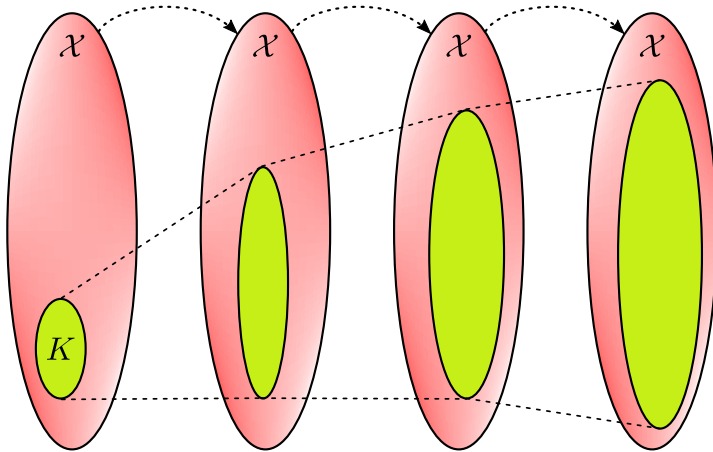
Knowledge Propagation

$(\mathcal{X}, \mathcal{R})$: System of relations, $K \subseteq \mathcal{X}$



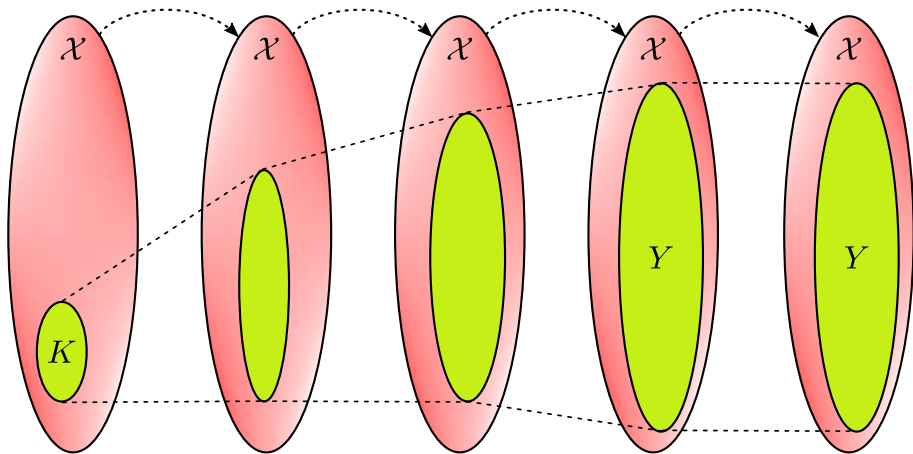
Knowledge Propagation

$(\mathcal{X}, \mathcal{R})$: System of relations, $K \subseteq \mathcal{X}$



Knowledge Propagation

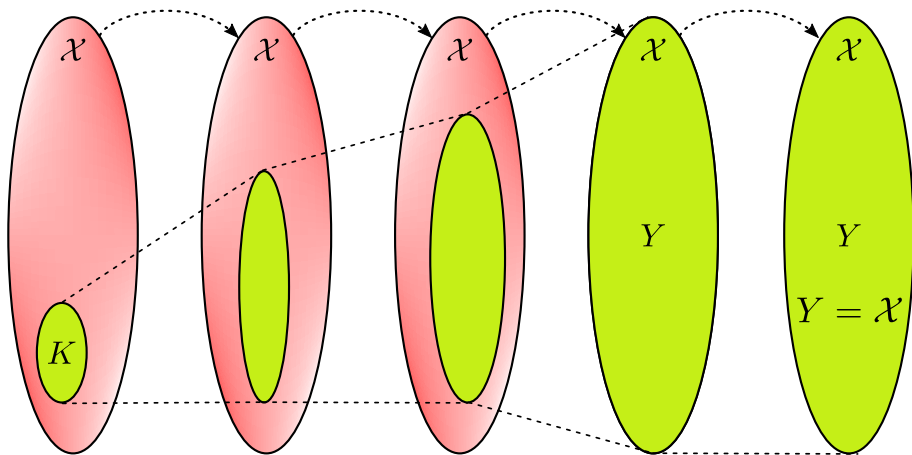
$(\mathcal{X}, \mathcal{R})$: System of relations, $K \subseteq \mathcal{X}$



$$Y = \text{Propagate}(K)$$

Knowledge Propagation

$(\mathcal{X}, \mathcal{R})$: System of relations, $K \subseteq \mathcal{X}$



If $\mathcal{X} = \text{Propagate}(K)$, then we say K is a *Guess Basis*

Naive Approach for GD

Given a system of relations $(\mathcal{X}, \mathcal{R})$, where $|\mathcal{X}| = n$, is there any guess basis of size $\leq m$?

Brute-force

- For $k = 1 \rightarrow m$
 - For each subset $K \subseteq \mathcal{X}$, where $|K| = k$:
 - If $\text{Propagate}(K) = \mathcal{X}$ then return K
- Time complexity $\approx \sum_{k=1}^m \binom{n}{k}$
- Exponential with respect to both n and m

Naive Approach for GD

Given a system of relations $(\mathcal{X}, \mathcal{R})$, where $|\mathcal{X}| = n$, is there any guess basis of size $\leq m$?

Brute-force

- For $k = 1 \rightarrow m$
 - For each subset $K \subseteq \mathcal{X}$, where $|K| = k$:
 - If $\text{Propagate}(K) = \mathcal{X}$ then return K

- Time complexity $\approx \sum_{k=1}^m \binom{n}{k}$
- Exponential with respect to both n and m

Naive Approach for GD

Given a system of relations $(\mathcal{X}, \mathcal{R})$, where $|\mathcal{X}| = n$, is there any guess basis of size $\leq m$?

Brute-force

- For $k = 1 \rightarrow m$
 - For each subset $K \subseteq \mathcal{X}$, where $|K| = k$:
 - If $\text{Propagate}(K) = \mathcal{X}$ then return K
- Time complexity $\approx \sum_{k=1}^m \binom{n}{k}$
- Exponential with respect to both n and m

Previous Works

- Heuristic Approaches:
 - ✔ Dynamic programming: [AE09]
 - ✔ Dedicated algorithm for GD attacks on AES: [BDF11]
- Using off-the-shelf solvers:
 - ✔ MILP: [Cen+20]
 - ✔ Gröbner basis: [DK20]

We borrowed the idea introduced in [Cen+20] to convert the GD problem to the CP/SAT problem.

Previous Works

- Heuristic Approaches:
 - ✔ Dynamic programming: [AE09]
 - ✔ Dedicated algorithm for GD attacks on AES: [BDF11]
- Using off-the-shelf solvers:
 - ✔ MILP: [Cen+20]
 - ✔ Gröbner basis: [DK20]

We borrowed the idea introduced in [Cen+20] to convert the GD problem to the CP/SAT problem.

Constraint Programming Model for GD

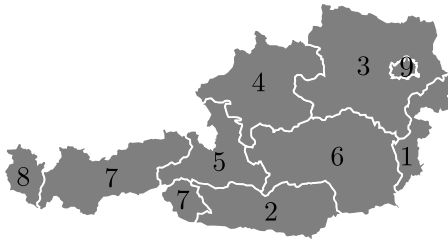


Constraint Programming (CP)

In CP we specify the properties of the solution to be found:

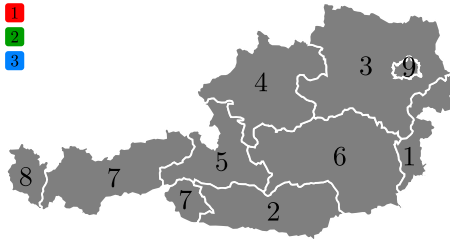
- We define a set of variables: \mathcal{X}
- We specify the domain of each variable: $\mathbb{F}_2, \mathbb{Z}, \mathbb{R}, \dots$
- We define a set of constraints: \mathcal{C}
- We define an objective function (if it is required)

CP Problem - Example



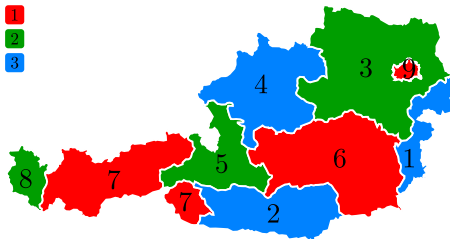
```
int: nc = 3;
array[1..9] of var 1..nc: r;
constraint r[1] != r[3]; constraint r[1] != r[6];
constraint r[2] != r[5]; constraint r[2] != r[6]; constraint r[2] != r[7];
constraint r[3] != r[9]; constraint r[3] != r[6]; constraint r[3] != r[4];
constraint r[4] != r[6]; constraint r[4] != r[5];
constraint r[5] != r[6]; constraint r[5] != r[7];
constraint r[7] != r[8];
solve satisfy;
r = [3, 3, 2, 3, 2, 1, 1, 2, 1];
```

CP Problem - Example



```
int: nc = 3;
array[1..9] of var 1..nc: r;
constraint r[1] != r[3]; constraint r[1] != r[6];
constraint r[2] != r[5]; constraint r[2] != r[6]; constraint r[2] != r[7];
constraint r[3] != r[9]; constraint r[3] != r[6]; constraint r[3] != r[4];
constraint r[4] != r[6]; constraint r[4] != r[5];
constraint r[5] != r[6]; constraint r[5] != r[7];
constraint r[7] != r[8];
solve satisfy;
r = [3, 3, 2, 3, 2, 1, 1, 2, 1];
```

CP Problem - Example



```
int: nc = 3;
array[1..9] of var 1..nc: r;
constraint r[1] != r[3]; constraint r[1] != r[6];
constraint r[2] != r[5]; constraint r[2] != r[6]; constraint r[2] != r[7];
constraint r[3] != r[9]; constraint r[3] != r[6]; constraint r[3] != r[4];
constraint r[4] != r[6]; constraint r[4] != r[5];
constraint r[5] != r[6]; constraint r[5] != r[7];
constraint r[7] != r[8];
solve satisfy;
r = [3, 3, 2, 3, 2, 1, 1, 2, 1];
```

Main Steps of Our Approach

Our method inspired from [Cen+20] has three main phases:

- Convert the system of equations to a system of (implication and symmetric) relations
- Convert the problem of finding a minimal guess basis for the system of relations to a CP problem or a sequence of SAT problems
- Employ the off-the-shelf SAT/CP solvers to solve the problem

Convert GD to a CP Problem

$$r_0 : [x, y, z]$$

$$r_1 : [z, w, y]$$

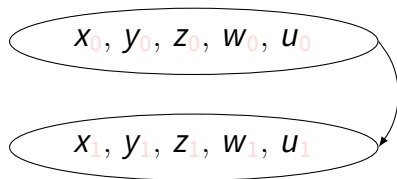
$$r_2 : [w, x, u]$$

Convert GD to a CP Problem

$r_0 : [x, y, z]$

$r_1 : [z, w, y]$

$r_2 : [w, x, u]$



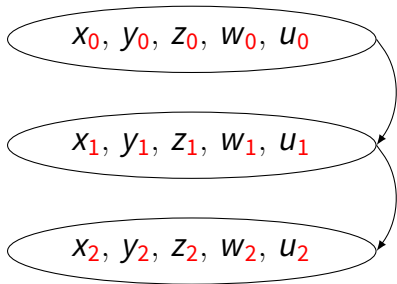
Convert GD to a CP Problem

$$r_0 : [x, y, z]$$

$$r_1 : [z, w, y]$$

$$r_2 : [w, x, u]$$

- Fix the number of steps in knowledge propagation (e.g. 2 here)
- $X = \{x_i, y_i, z_i, w_i, u_i : 0 \leq i \leq 2\}$
- $x_i = 1$ iff x is known after the i th step of knowledge propagation, otherwise $x_i = 0$
- $\mathcal{C} \leftarrow \emptyset$

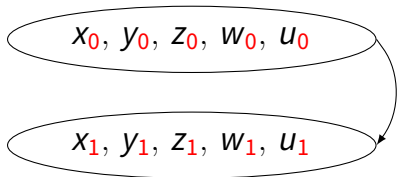


Convert GD to a CP Problem

$$r_0 : [x, y, z]$$

$$r_1 : [z, w, y]$$

$$r_2 : [w, x, u]$$



$$X \leftarrow X \cup \{x_{0,0}, x_{0,1}\}$$

$$\mathcal{C} \leftarrow \mathcal{C} \cup \{x_{0,0} = y_0 \wedge z_0\}$$

$$\mathcal{C} \leftarrow \mathcal{C} \cup \{x_{0,1} = w_0 \wedge u_0\}$$

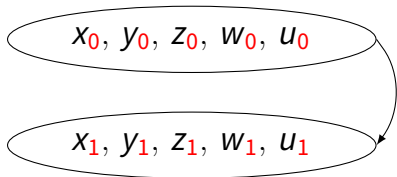
$$\mathcal{C} \leftarrow \mathcal{C} \cup \{x_1 = x_{0,0} \vee x_{0,1}\}$$

Convert GD to a CP Problem

$$r_0 : [x, y, z]$$

$$r_1 : [z, w, y]$$

$$r_2 : [w, x, u]$$



$$X \leftarrow X \cup \{y_{0,0}, y_{0,1}\}$$

$$\mathcal{C} \leftarrow \mathcal{C} \cup \{y_{0,0} = x_0 \wedge z_0\}$$

$$\mathcal{C} \leftarrow \mathcal{C} \cup \{y_{0,1} = z_0 \wedge w_0\}$$

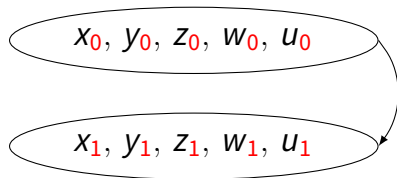
$$\mathcal{C} \leftarrow \mathcal{C} \cup \{y_1 = y_{0,0} \vee y_{0,1}\}$$

Convert GD to a CP Problem

$r_0 : [x, y, z]$

$r_1 : [z, w, y]$

$r_2 : [w, x, u]$



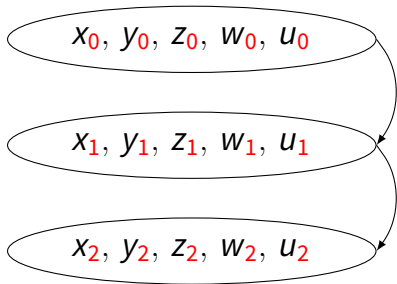
Do it for z, w, u as well

Convert GD to a CP Problem

$r_0 : [x, y, z]$

$r_1 : [z, w, y]$

$r_2 : [w, x, u]$



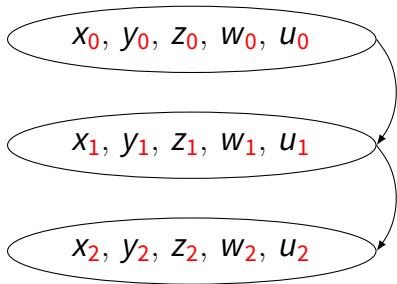
Do it for each transition in
knowledge propagation

Convert GD to a CP Problem

$$r_0 : [x, y, z]$$

$$r_1 : [z, w, y]$$

$$r_2 : [w, x, u]$$



All variables should be known
at the last step:

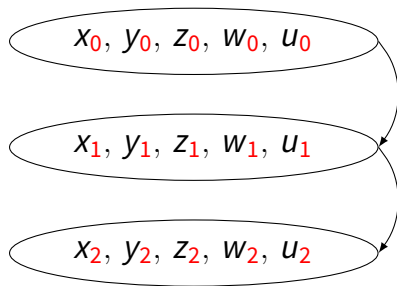
$$\mathcal{C} \leftarrow \mathcal{C} \cup \{x_2 \wedge y_2 \wedge z_2 \wedge w_2 \wedge u_2 = 1\}$$

Convert GD to a CP Problem

$r_0 : [x, y, z]$

$r_1 : [z, w, y]$

$r_2 : [w, x, u]$



$\min x_0 + y_0 + z_0 + w_0 + u_0$

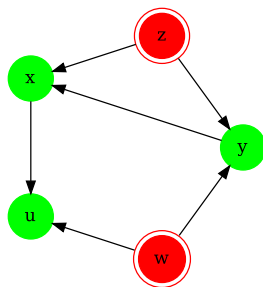
s.t. all constraints in \mathcal{C} are satisfied

Convert GD to a CP Problem

$r_0 : [x, y, z]$

$r_1 : [z, w, y]$

$r_2 : [w, x, u]$



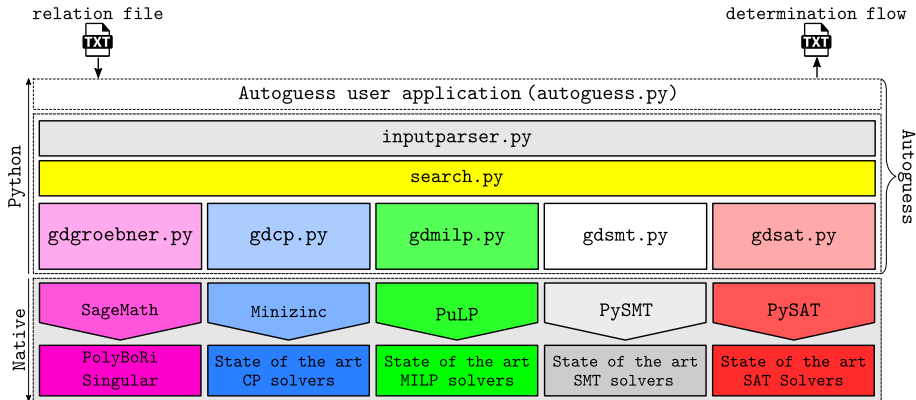
$$\min x_0 + y_0 + z_0 + w_0 + u_0$$

s.t. all constraints in \mathcal{C} are satisfied

Autoguess



Autoguess



Autoguess - Simple User Interface

$$\left\{ \begin{array}{ll} F(u + v) \oplus G(x) \oplus y \oplus (z \lll 7) & = c_1 \\ G(u \oplus w) + (y \lll 3) + z & = c_2 \\ F(w \oplus x) + y \oplus z & = c_3 \\ F(u) \oplus G(w + z) & = c_4 \\ (F(u) \times G(w \lll 7)) + H(z \oplus v) & = c_5 \end{array} \right.$$

Autoguess - Simple User Interface

Input file (relations.txt):

```
1 # Comments
2 connection relations
3 u, v, x, y, z
4 u, w, y, z
5 w, x, y, z
6 u, w, z
7 u, w => t
8 t, z, v
9 end
```

Run Autoguess:

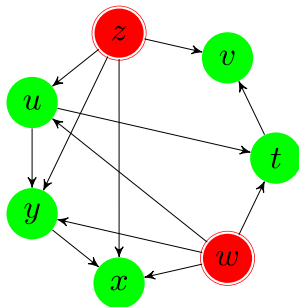
```
python3 autoguess.py -i relations.txt --maxsteps 5 --solver cp
```

Autoguess - Simple User Interface

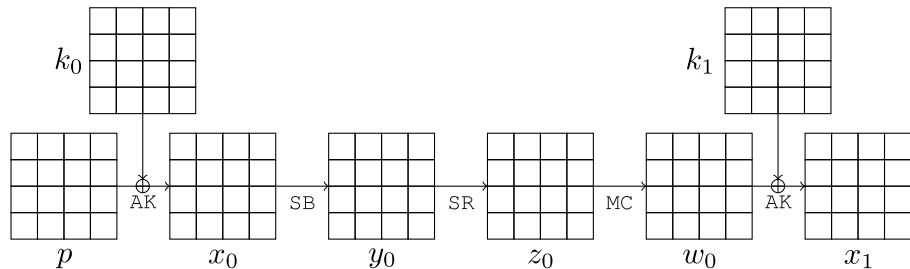
Input file (relations.txt):

```
1 # Comments
2 connection relations
3 u, v, x, y, z
4 u, w, y, z
5 w, x, y, z
6 u, w, z
7 u, w => t
8 t, z, v
9 end
```

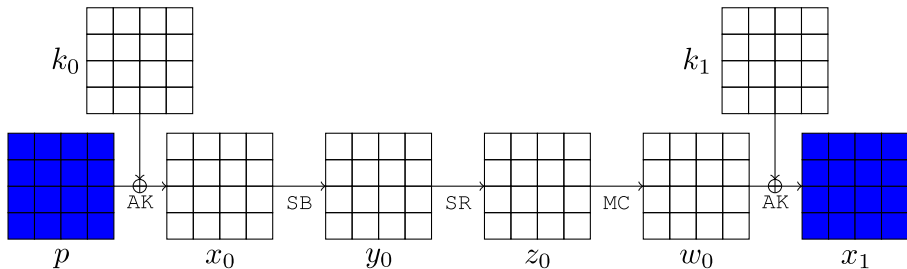
Output:



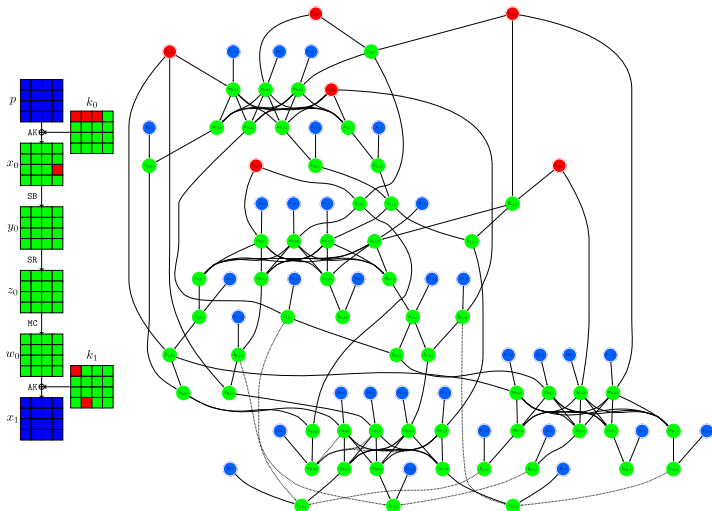
GD Attack on Block Ciphers (1 round of AES)



GD Attack on Block Ciphers (1 round of AES)

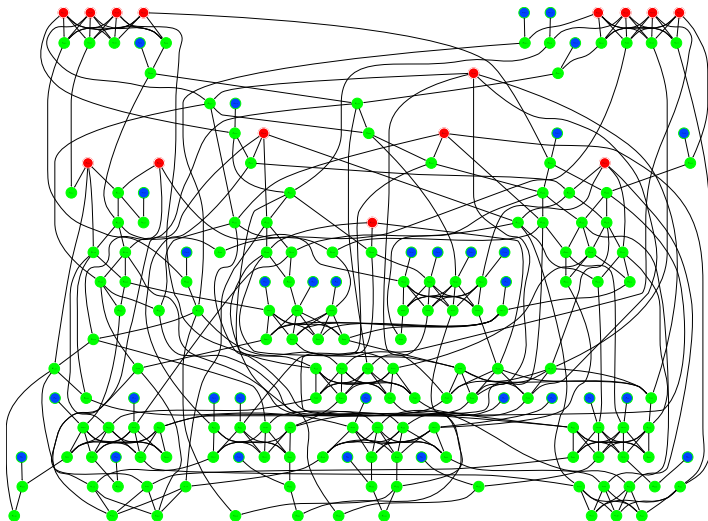


GD Attack on Block Ciphers (1 round of AES)



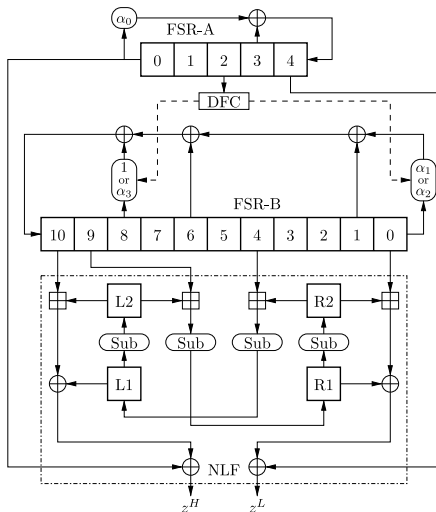
Found in **0.02 seconds** on a 2.8 GHz i7-1165G7 CPU

GD Attack on Block Ciphers (3 Rounds of AES)



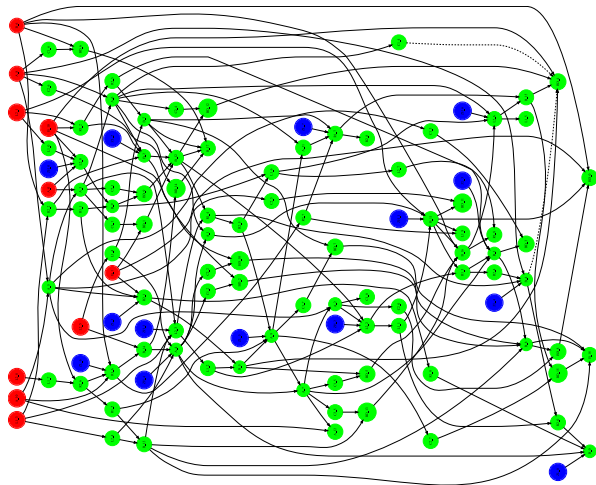
Found in **34.51 seconds** on a 2.8 GHz i7-1165G7 CPU

GD Attack on Stream Ciphers (KCipher-2)



ISO/IEC 18033-4

GD Attack on Stream Ciphers (KCipher-2)

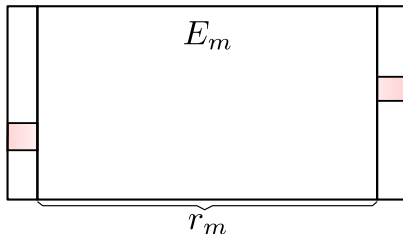


Found in **7 seconds** on a 2.8 GHz i7-1165G7 CPU

Key-Bridging (KB)

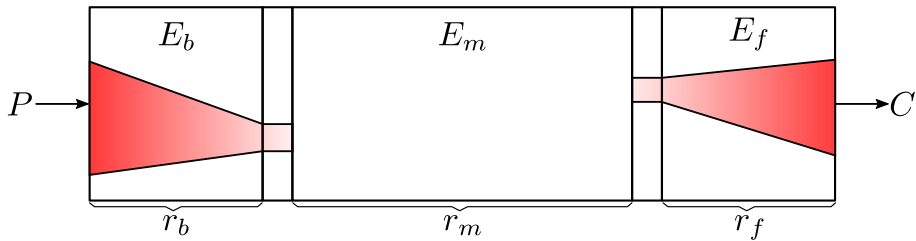


Key-Bridging (KB)



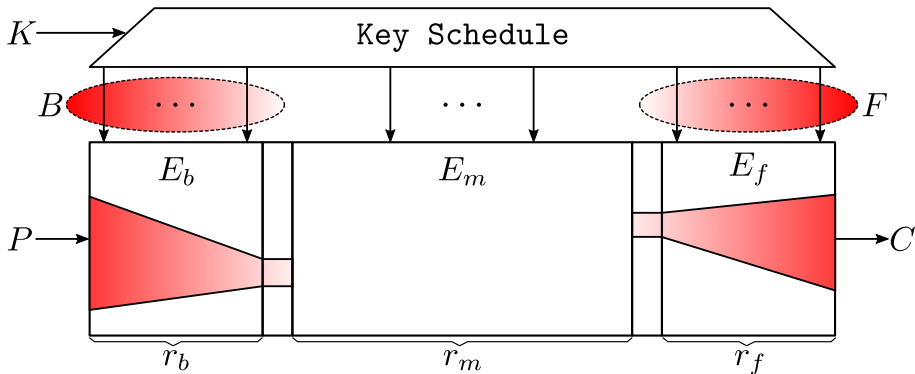
- We want to determine a subset of sub-key bits: $B \cup F$
- Key schedule implies some relations between the sub-key bits in $B \cup F$

Key-Bridging (KB)



- We want to determine a subset of sub-key bits: $B \cup F$
- Key schedule implies some relations between the sub-key bits in $B \cup F$

Key-Bridging (KB)



- We want to determine a subset of sub-key bits: $B \cup F$
- Key schedule implies some relations between the sub-key bits in $B \cup F$
- Find a minimal guess basis for $B \cup F$ (not for the entire set of variables)

\mathcal{DS} -MITM Attacks On SKINNY and TWINE

- We combined our CP-based method for KB with CP-based method to search for distinguishers
- We applied our method to optimize \mathcal{DS} -MITM attack on SKINNY [Bei+16] and TWINE [Suz+11]

Cipher	#Rounds	Data	Memory	Time	Attack	Setting	Reference
SKINNY-128-256	19	2^{96} CP	$2^{210.99}$	$2^{238.26}$	\mathcal{DS} -MITM	ST	This paper
SKINNY-64-192	21	2^{60} CP	$2^{133.99}$	$2^{186.63}$	\mathcal{DS} -MITM	ST	This paper
SKINNY-64-128	18	2^{32} CP	$2^{61.91}$	$2^{126.32}$	\mathcal{DS} -MITM	ST	This paper
TWINE-80	20	2^{32} CP	$2^{62.91}$	$2^{76.92}$	\mathcal{DS} -MITM	-	This paper
TWINE-80	20	2^{32} CP	$2^{82.91}$	$2^{77.44}$	\mathcal{DS} -MITM	-	[Shi+18]

Conclusion



Our Contributions - I

- ✔ We introduced two new encoding methods for GD technique (CP & SAT)
- ✔ We provided the open-source tool Autoguess integrating our new methods as well as almost all of the previous methods for GD technique
- ✔ We applied our tool on a wide variety of symmetric primitives:
 - Improving the GD attack on ZUC [ETS11; Tea18]
 - Rediscovering the GD attack on 3 rounds of AES in less than a minute
- ✔ We applied Autoguess to find key-bridges in key recovery attacks on block ciphers

Our Contributions - I

- ✔ We introduced two new encoding methods for GD technique (CP & SAT)
- ✔ We provided the open-source tool Autoguess integrating our new methods as well as almost all of the previous methods for GD technique
- ✔ We applied our tool on a wide variety of symmetric primitives:
 - Improving the GD attack on ZUC [ETS11; Tea18]
 - Rediscovering the GD attack on 3 rounds of AES in less than a minute
- ✔ We applied Autoguess to find key-bridges in key recovery attacks on block ciphers

Our Contributions - I

- ✔ We introduced two new encoding methods for GD technique (CP & SAT)
- ✔ We provided the open-source tool Autoguess integrating our new methods as well as almost all of the previous methods for GD technique
- ✔ We applied our tool on a wide variety of symmetric primitives:
 - Improving the GD attack on ZUC [ETS11; Tea18]
 - Rediscovering the GD attack on 3 rounds of AES in less than a minute
- ✔ We applied Autoguess to find key-bridges in key recovery attacks on block ciphers

Our Contributions - II

- ✔ We combined our CP-based approach for key-bridging with the CP-based methods to search for distinguishers, and introduced a unified method to find key-recovery friendly distinguishers:

Thanks for your attention!

: <https://github.com/hadipourh/autoguess>

: <https://ia.cr/2021/1529>

Our Contributions - II

- ✔ We combined our CP-based approach for key-bridging with the CP-based methods to search for distinguishers, and introduced a unified method to find key-recovery friendly distinguishers:

Thanks for your attention!

🐙: <https://github.com/hadipourh/autoguess>

📄: <https://ia.cr/2021/1529>

Bibliography I

- [AE09] Hadi Ahmadi and Taraneh Eghlidos. **Heuristic guess-and-determine attacks on stream ciphers.** IET Information Security 3.2 (2009), pp. 66–73.
- [BDF11] Charles Bouillaguet, Patrick Derbez, and Pierre-Alain Fouque. **Automatic search of attacks on round-reduced AES and applications.** Advances in Cryptology – CRYPTO 2011. Springer. 2011, pp. 169–187.
- [Bei+16] Christof Beierle et al. **The SKINNY family of block ciphers and its low-latency variant MANTIS.** Advances in Cryptology – CRYPTO 2016. Springer. 2016, pp. 123–153.
- [Cen+20] Zhe Cen et al. **Minimizing Deduction System and its Application.** arXiv preprint arXiv:2006.05833 (2020). URL: <https://arxiv.org/abs/2006.05833>.
- [DK20] J Danner and M Kreuzer. **A fault attack on KCipher-2.** International Journal of Computer Mathematics: Computer Systems Theory (2020), pp. 1–22.

Bibliography II

- [ETS11] ETSI/SAGE. **Specification of the 3GPP confidentiality and integrity algorithms 128-EEA3 and 128-EIA3: ZUC specification.** ETSI/SAGE, Document 2, Version 1.6 (2011).
- [Shi+18] Danping Shi et al. **Programming the Demirci-Selçuk meet-in-the-middle attack with constraints.** Advances in Cryptology – ASIACRYPT 2018. Springer. 2018, pp. 3–34.
- [Suz+11] Tomoyasu Suzaki et al. **Twine: A lightweight, versatile block cipher.** ECRYPT workshop on lightweight cryptography. Vol. 2011. 2011.
- [Tea18] ZUC Design Team. **The ZUC-256 Stream Cipher.** (2018). <http://www.is.cas.cn/ztzl2016/zouchongzhi/201801/W020180126529970733243.pdf>.