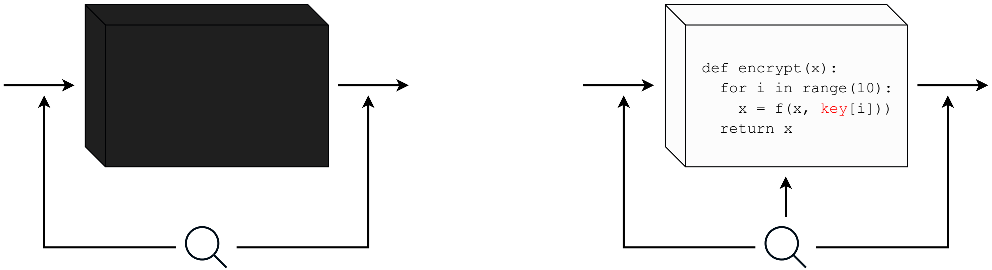
Implicit White-Box Implementations: White-Boxing ARX Ciphers

Adrián Ranea, Joachim Vandersmissen, and Bart Preneel

White-Box Cryptography



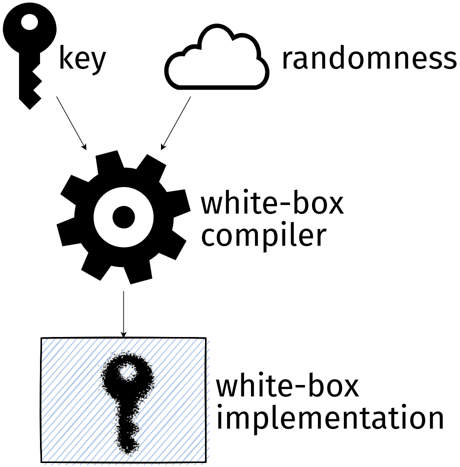
WBC: securing software crypto. implementations in the white-box model.

Applications: DRM, mobile payments, ...

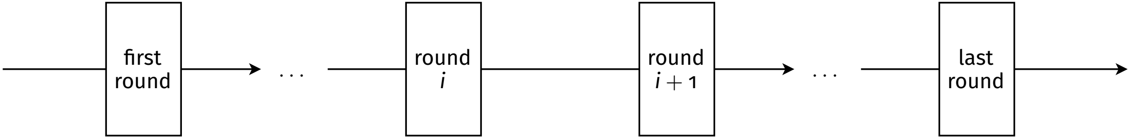
White-Box Implementations of Block Ciphers

Academic white-box implementations:

- Fixed hard-coded cipher key.
- Compiler/method public.
- Security goal: key-extraction resistance.

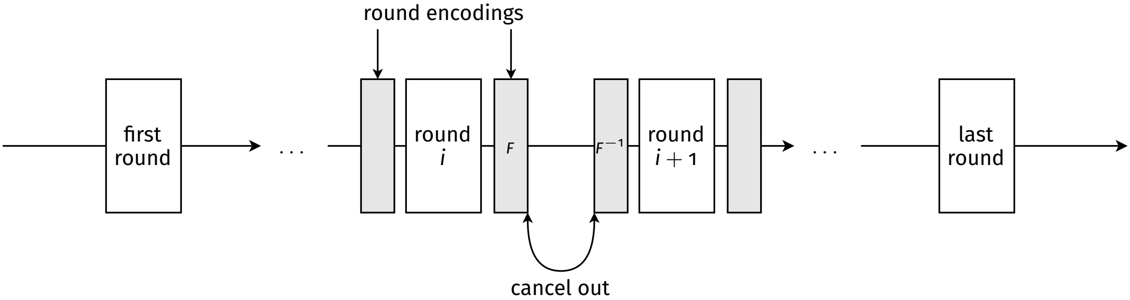


CEJO Implementations



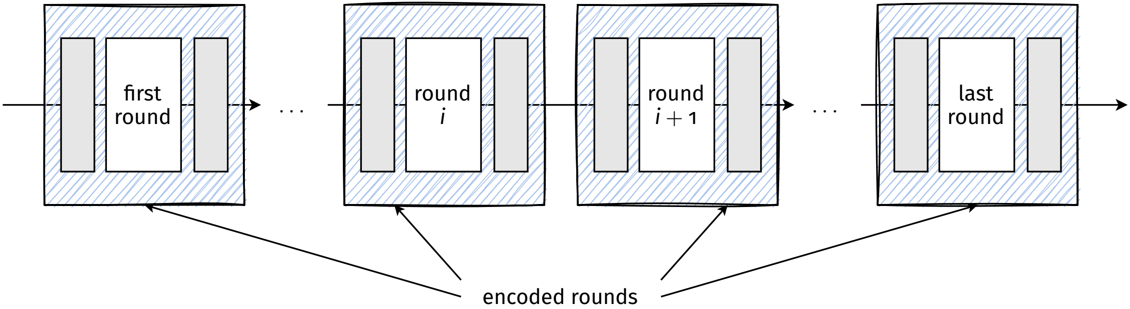
Chow S, Eisen P, Johnson H & Van Oorschot PC. *White-Box Cryptography and an AES Implementation.* SAC 2002.

CEJO Implementations



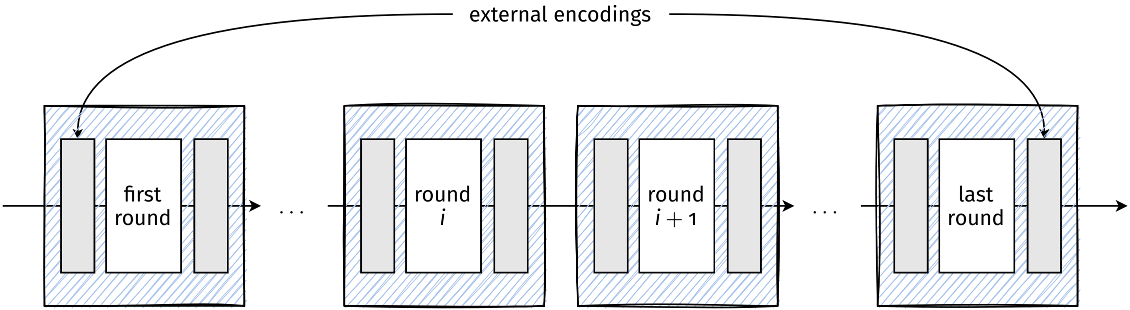
Chow S, Eisen P, Johnson H & Van Oorschot PC. *White-Box Cryptography and an AES Implementation.* SAC 2002.

CEJO Implementations



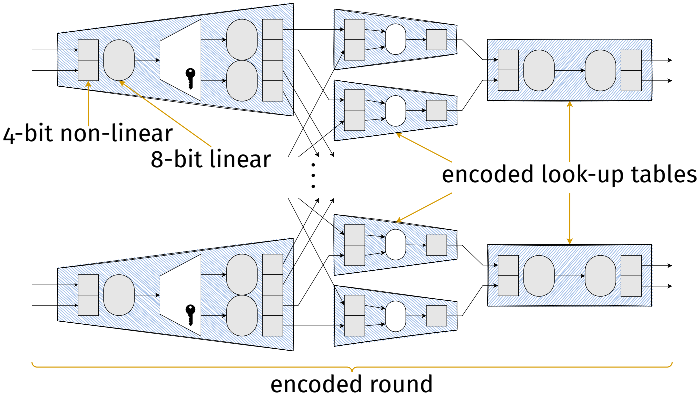
Chow S, Eisen P, Johnson H & Van Oorschot PC. *White-Box Cryptography and an AES Implementation.* SAC 2002.

CEJO Implementations



Chow S, Eisen P, Johnson H & Van Oorschot PC. *White-Box Cryptography and an AES Implementation.* SAC 2002.

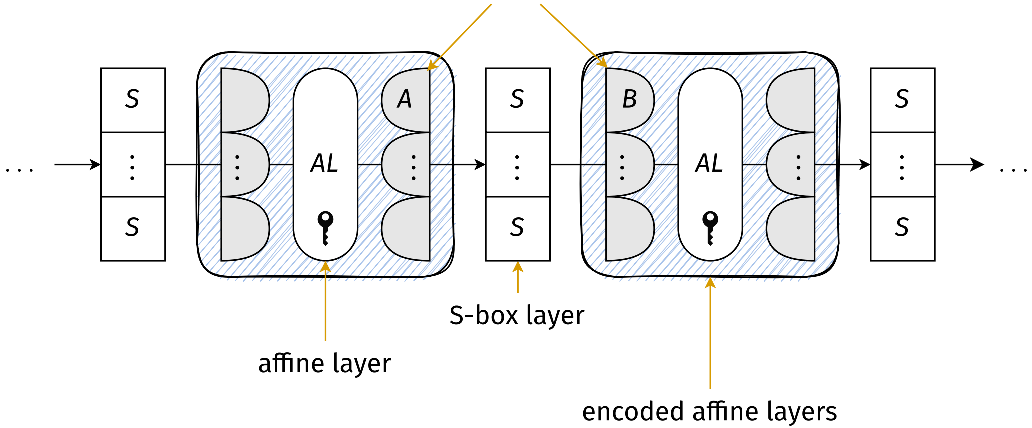
CEJO round



All CEJO implementations have been broken.

Self-Equivalence Implementations

(A, B) self-equivalence of $S \iff S = B \circ S \circ A$



Self-Equivalence (SE) Implementations

- No look-up tables,
  SE efficient with large encodings.
- CEJO can be reduced to SE,
  but the converse doesn't hold.

- **Problem:** difficult to find
  non-linear layers with
  many and large self-equivalences.

Self-Equivalence (SE) Implementations

- No look-up tables,
  SE efficient with large encodings.
- CEJO can be reduced to SE,
  but the converse doesn't hold.

- **Problem:** difficult to find
  non-linear layers with
  many and large self-equivalences.

Candidate non-linear layer: permuted modular addition $x, y \mapsto (x \boxplus y, y)$

Finding Self-equivalences of $x, y \mapsto (x \boxplus y, y)$

F is CCZ-equivalent to G if the graph of F, $\{(x, F(x))\}$, is equal to the graph of G up to an affine permutation.

- $F(x, y) = (x \boxplus y, y)$ is CCZ-equivalent to a quadratic function G.

Finding Self-equivalences of $x, y \mapsto (x \boxplus y, y)$

F is CCZ-equivalent to G if the graph of F, $\{(x, F(x))\}$, is equal to the graph of G up to an affine permutation.

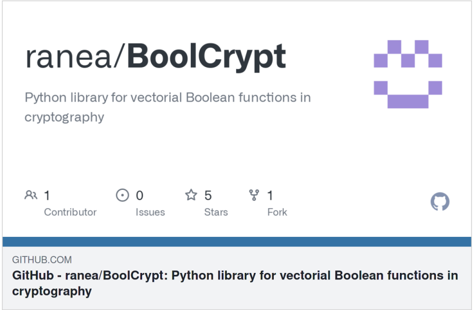
- $F(x, y) = (x \boxplus y, y)$ is CCZ-equivalent to a quadratic function G.

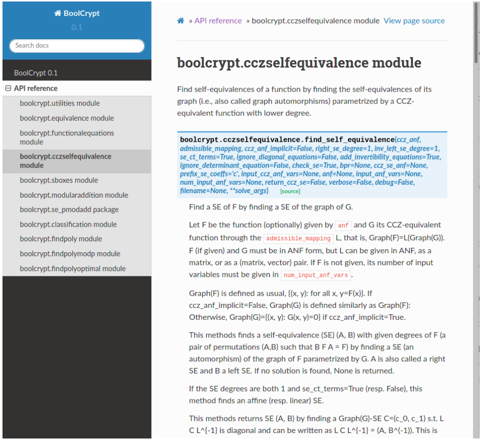
A graph automorphism of F is an affine permutation mapping the graph of F to itself.

CCZ-based method to find self-equivalences of F:

- Find graph automorphisms of low-degree G by solving a functional equation.
- Transform graph automorphisms of G into self-equivalences of F using CCZ-equivalence.

github.com/ranea/Boolcrypt





Self-Equivalences of the Permuted Modular Addition

SE found for wordsize $4 \leq n \leq 64$:

Type	$\#\{(A,B) : S = B \circ S \circ A\}$
Linear	$3 \times 2^{2n+2}$
Affine	$3 \times 2^{2n+8}$
Affine-quadratic	$3^2 \times 2^{3n+14} - 3 \times 2^{2n+8}$

Open problem: prove these SE subsets are the full SE groups for $n \geq 4$.

Self-Equivalences of the Permuted Modular Addition

SE found for wordsize $4 \leq n \leq 64$:

Type	$\#\{(A, B) : S = B \circ S \circ A\}$
Linear	$3 \times 2^{2n+2}$
Affine	$3 \times 2^{2n+8}$
Affine-quadratic	$3^2 \times 2^{3n+14} - 3 \times 2^{2n+8}$

Open problem: prove these SE subsets are the full SE groups for $n \geq 4$.

$$
\left(\begin{array}{ccccc|ccccc}
\star & & & & & \star & & & & \\
\star & 1 & & & & \star & & & & \\
\vdots & & \ddots & & & \vdots & & & & \\
\star & & & 1 & & \star & & & & \\
\star & & & \star & 1 & \star & \star & \dots & \star & \star \\
\hline
\star & & & & & \star & & & & \\
\star & & & & & \star & 1 & & & \\
\vdots & & & & & \vdots & & \ddots & & \\
\star & & & & & \star & & & 1 & \\
\star & & & & & \star & & & \star & 1
\end{array}\right)
$$

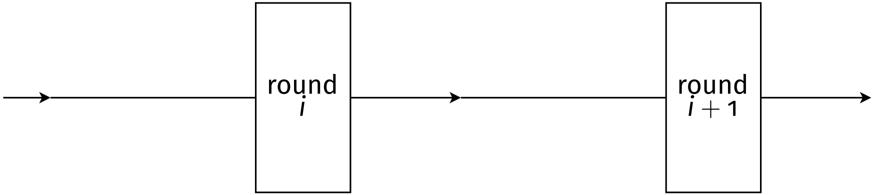
$$
\left(\star \; b \; \cdots \; b \; \star \; \star \,\middle|\, \star \; b \; \cdots \; b \; \star \; \star\right)
$$

Implicit framework

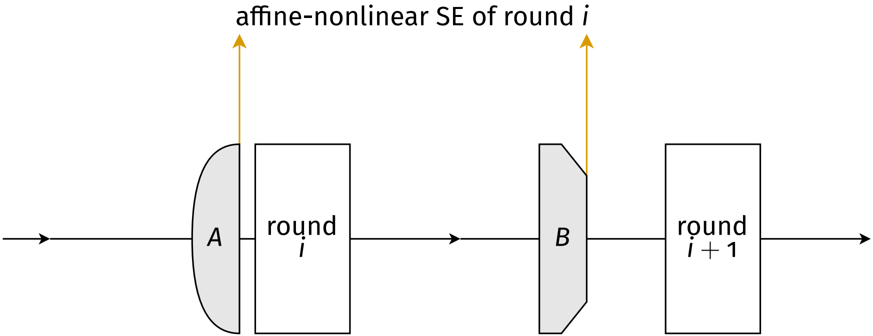
Implicit implementation:

- an encoded implementation where
- the round encodings are the composition of affine permutations and affine-nonlinear self-equivalences,
- and the encoded round functions are implemented by systems of low-degree equations.

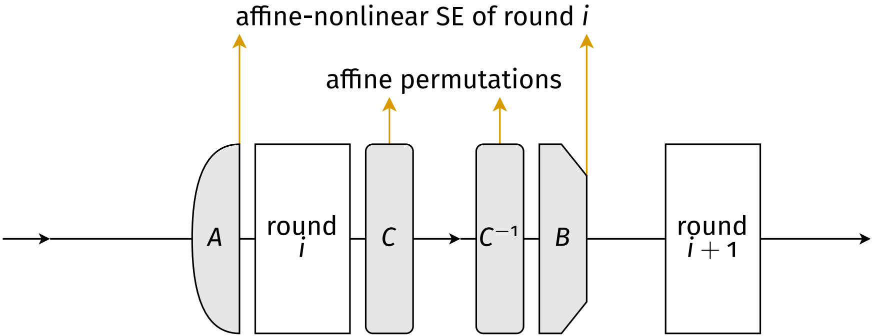
Round Encodings in an Implicit Implementation



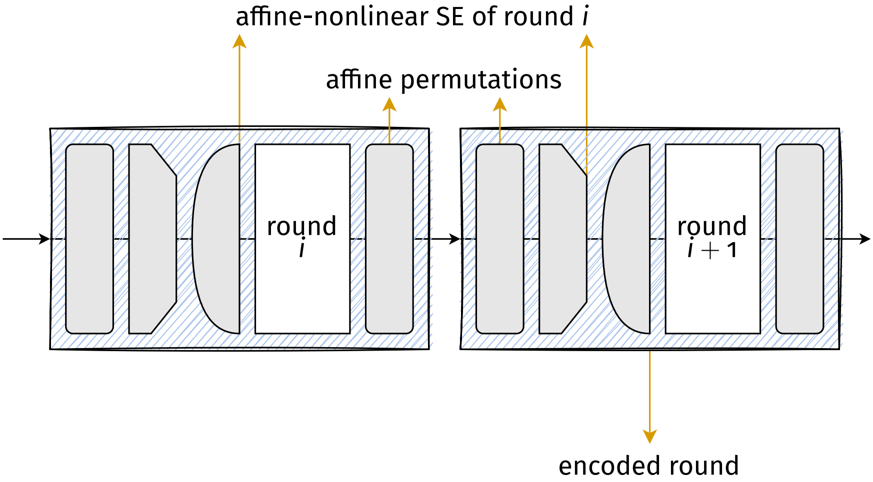
Round Encodings in an Implicit Implementation



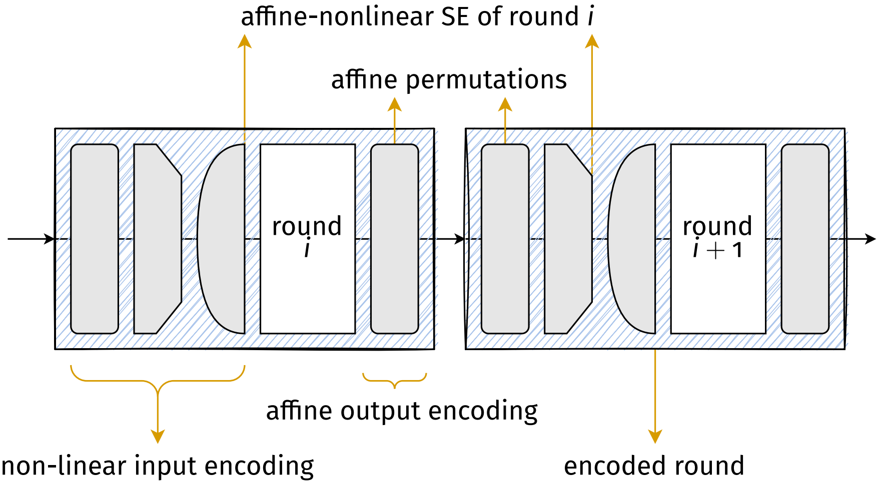
Round Encodings in an Implicit Implementation



Round Encodings in an Implicit Implementation



Round Encodings in an Implicit Implementation



Implicit Round Functions

Implemented by low-degree quasilinear implicit functions:

- P is an implicit function of F if

$$P(x, y) = 0 \iff y = F(x)\,.$$

- Evaluate $F(x_0)$ by substituting $x_0 = x$ and solving $P(x_0, y) = 0$ for y.
- Fast solving if P is quasilinear:

$$\forall x, \text{ the function } y \mapsto P(x, y) \text{ is affine}\,.$$

Permuted modular addition has a quasilinear quadratic implicit function!

Size of an Implicit Round Function

Upper bound on the size of a $(2n, n)$-bit P for an n-bit cipher.

Cipher blocksize	Degree of P	Size of P
64	2	0.05 MB
64	3	1.42 MB
64	4	6.50 MB
128	2	0.40 MB
128	3	22.50 MB
128	4	193.19 MB

For the permuted modular addition, P is cubic or quartic if affine-quadratic self-equivalences are used.

Security of Implicit Implementations

Security goal: key-extraction resistance.

An implicit implementation is secure against known generic attacks if

- quadratic input encodings OR
- large non-linear layer

Implicit framework cannot secure SPN ciphers with affine encodings.

Security of Implicit Implementations

New generic attack (reduction to self-equivalence implementations) based on functional equations (affine equivalence problems):

known functions

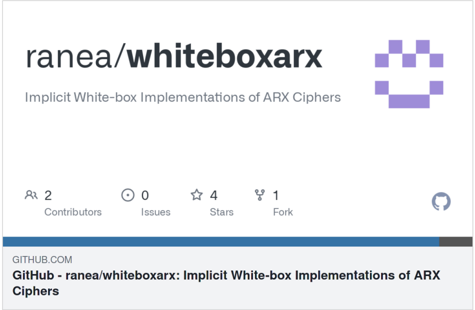
$$G = Y \circ F \circ X$$

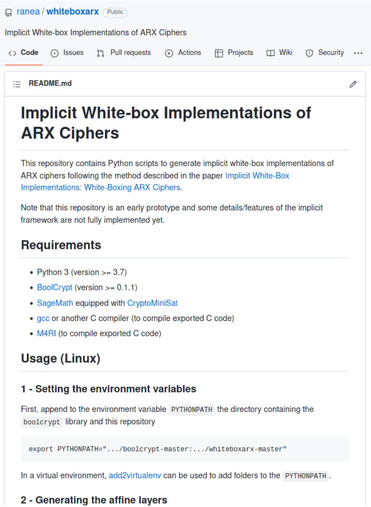
unknown affine permutations

Security of Implicit Implementations

function	$\overline{E} = O \circ (L \circ S) \circ \oplus_k \circ I$	$P = V \circ T \circ U \circ (\oplus_k \parallel L^{-1}) \circ (I \parallel O^{-1})$
equation	$\overline{E} = Y \circ (L \circ S) \circ X$	$P = Y \circ T \circ X$
degree	high	low
access	black-box	white-box
goal	find any solution	find any solution and guess U

github.com/ranea/whiteboxarx





Implicit White-box Implementations of ARX Ciphers

This repository contains Python scripts to generate implicit white-box implementations of ARX ciphers following the method described in the paper Implicit White-Box Implementations: White-Boxing ARX Ciphers.

Note that this repository is an early prototype and some details/features of the implicit framework are not fully implemented yet.

Requirements

- Python 3 (version >= 3.7)
- BoolCrypt (version >= 0.1.1)
- SageMath equipped with CryptoMiniSat
- gcc or another C compiler (to compile exported C code)
- M4RI (to compile exported C code)

Usage (Linux)

1 - Setting the environment variables

First, append to the environment variable `PYTHONPATH` the directory containing the `boolcrypt` library and this repository

```
export PYTHONPATH=".../boolcrypt-master:.../whiteboxarx-master"
```

In a virtual environment, add2virtualenv can be used to add folders to the `PYTHONPATH`.

2 - Generating the affine layers

Conclusion

- Implicit framework: new design that prevents generic attacks, first method applicable to ARX ciphers.
- New method to find self-equivalences based on the CCZ-equivalence, applied to the permuted modular addition.
- Two open-source tools: `BoolCrypt` and `whiteboxarx`.
- Future work: new attacks, other non-linear layers, …