White-Box Cryptography

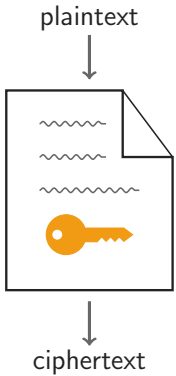
Junwei Wang

CryptoExperts

Outline

1. Introduction
2. Chow *et al.*'s Design
3. Generic Attacks
4. *WhibOx* Contest

Introduction

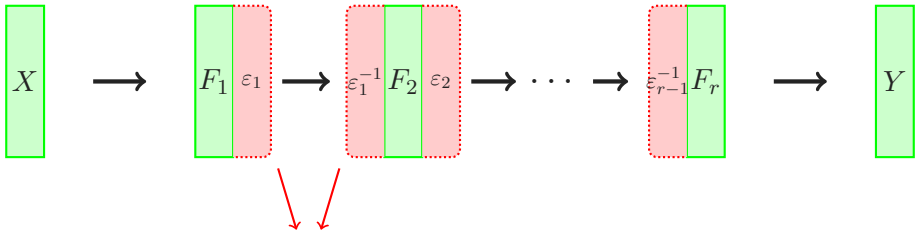


- Resistant against **key extraction** from pure **software** cryptographic implementations [CEJv002]
- Everything in academic is **broken**
- No *provably secure* construction (for standard ciphers)
- Applications: DRM and mobile payment

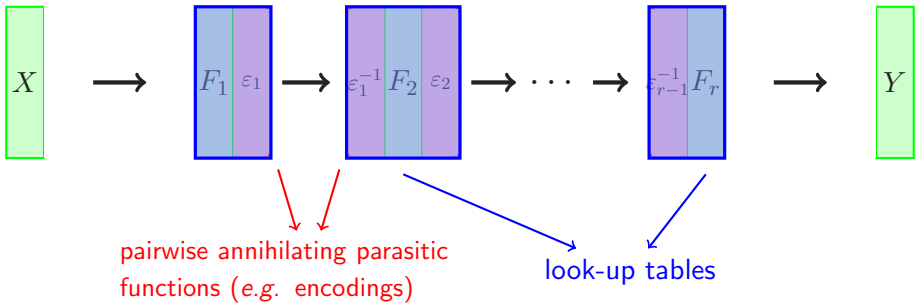
rapid growth of market
⇓
home-made solutions
(security through *obscurity*!)

Outline

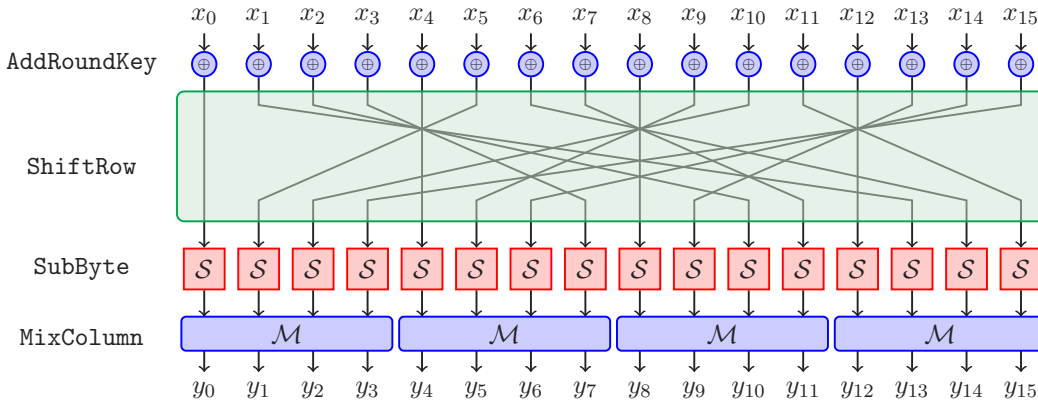
Seminal White-Box Design [CEJv002] (Sketch)



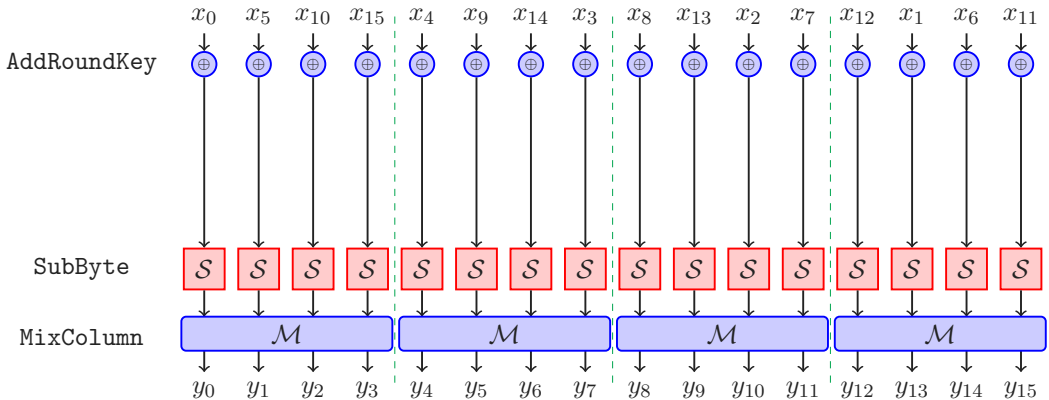
Seminal White-Box Design [CEJv002] (Sketch)



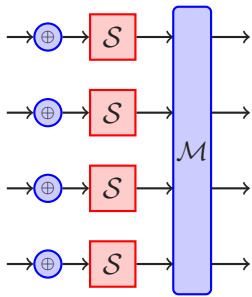
AES Round Operation



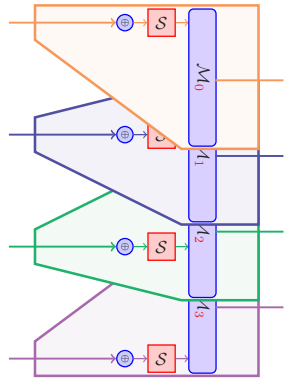
AES Round Operation



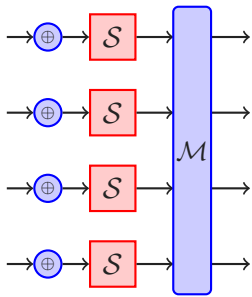
Look-Up Table Implementation



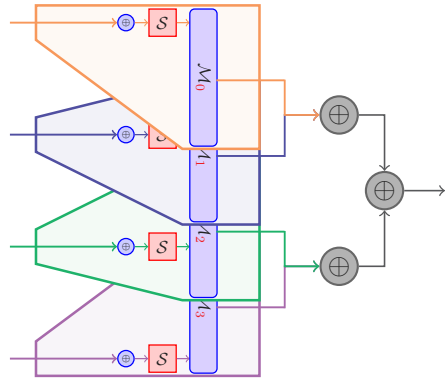
4×T-Boxes



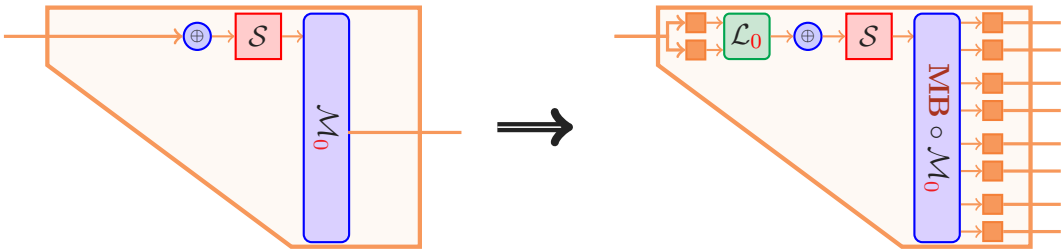
Look-Up Table Implementation



4× T-Boxes 3× `UINT32` XOR



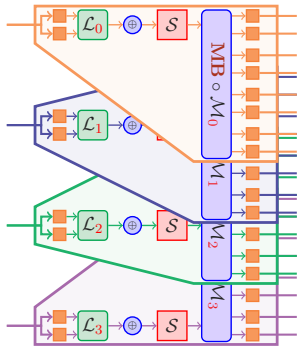
Protecting a LUT



4-bit non-linear bijection 8-bit invertible linear trans.

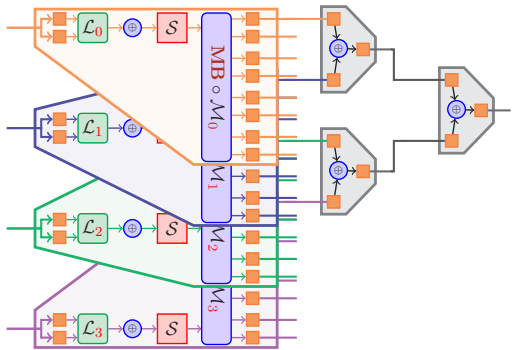
Merging

4× protected T-Boxes



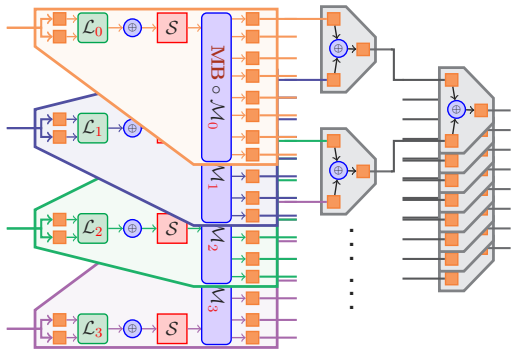
Merging

4×protected T-Boxes 24×XOR tables



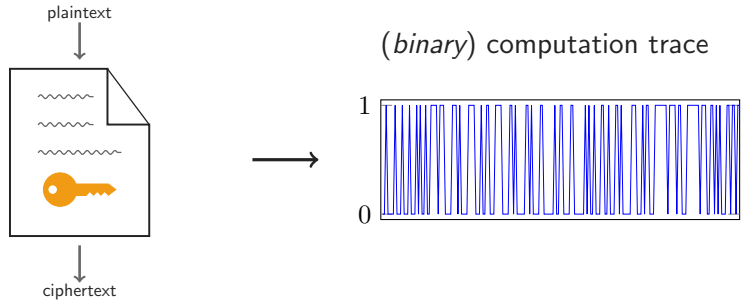
Merging

4×protected T-Boxes 24×XOR tables



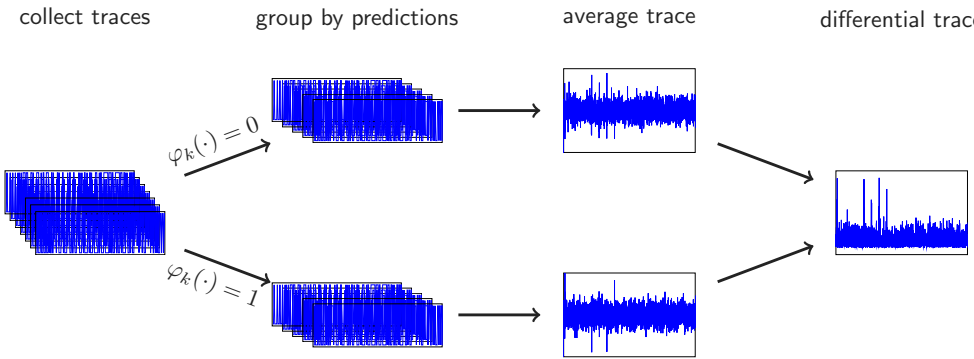
Outline

Differential Computation Analysis (DCA)



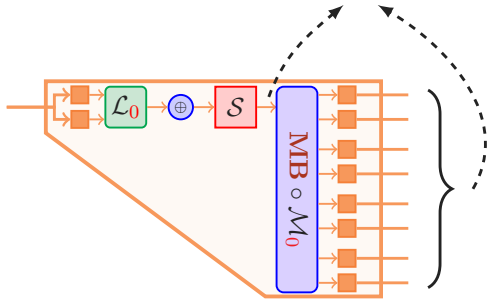
- DPA techniques in white-box context [BHMT16]
- Instead of *power traces*, using *computation traces* usually consisting of runtime memory information

DCA Techniques



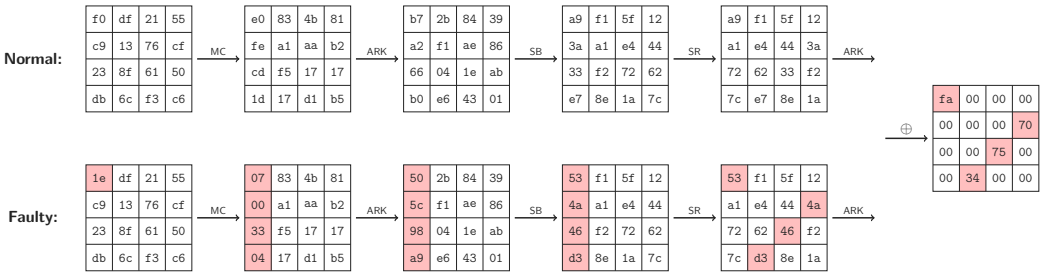
DCA Explanation

Very likely to be linearly correlated [BBMT18]



- Implying nibble encodings are insufficient in white-box context.

Differential Fault Attack (DFA)



- Modify a state byte between last two `MixColumns`
  - How: statically / dynamically
  - Expecting certain differential patterns (thanks to ShiftRow)
  - Very few faulty executions are required to recover a column of key bytes
- Many other fault injection techniques


CHES 2017
Capture the Flag
Challenge
The WhibOx Contest
An ECRYPT White-Box Cryptography Competition

WhibOx Contest

- **Goal**: confront designers and attackers in the secret design paradigm
- **Designers**: invited to submit AES-128 implementations in C
  - with secret chosen key
  - source code $\leq$ 50MB
  - compiled binary $\leq$ 20MB
  - RAM consumption $\leq$ 20MB
  - execution time $\leq$ 1 second
- **Breakers**: invited to recover the hidden keys
- *Not required to* disclose their identity & underlying techniques

WhibOx Contest

- The competition lasted for about 4 months.
- Results:
  - 94 submissions were **all broken** by 877 individual breaks
  - most (86%) of them were alive for < 1 day
- Scoreboard (top 5): ranked by **surviving time**

id	designer	first breaker	score	#days	#breaks
777	cryptolux	team_cryptoexperts	406	28	1
815	grothendieck	cryptolux	78	12	1
753	sebastien-riou	cryptolux	66	11	3
877	chaes	You!	55	10	2
845	team4	cryptolux	36	8	2

🎉 **cryptolux**: Biryukov, Udovenko

🎉 **team_cryptoexperts**: Goubin, Paillier, Rivain, Wang

Conclusion

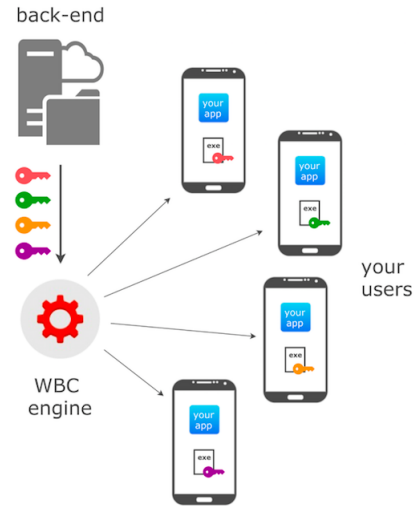
- White-box cryptography against key extraction is important to security of pure software
  - Academia: everything is broken
  - Industry: security through obscurity
- Chow *et al.*'s implementation
  - Was broken by structural analyses many times
  - Still play an important role in the subsequent designs
- DCA and DFA are automatic and generic
  - Not required to know the underlying techniques
- Previous *WhibOx* contest was quite successful, and new edition comes back in next February!

But wait!

White-Box Technology by CRYPTOEXPERTS

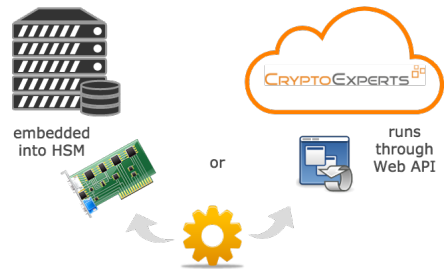
Our technology

- enjoys both performance and security
- can be deployed anywhere
  - smartphones, desktops, wearable devices
- covers standard cryptographic algorithms:
  - block cipherts (AES, 3DES, SM4, ...)
  - message authentication code (ISO 9797 3DES MAC, AES CMAC, HMAC, ...)
  - signature schemes (RSA, ECDSA, SM2, ...)
- supports specific algorithms on demand
- We can customize WBC-friendly cryptographic algorithm if allowed

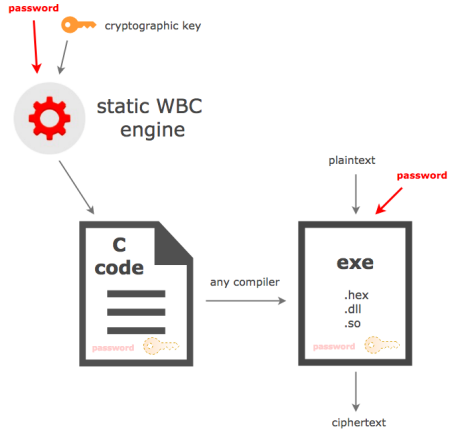


Deployment Options

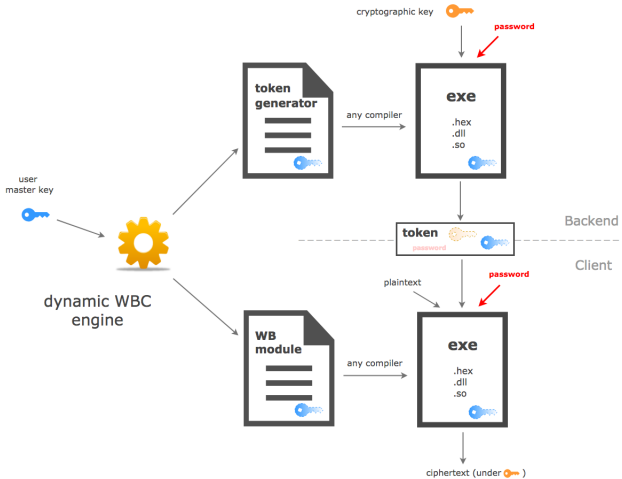
- WebAPI maintained by CryptoExperts
- Embedded into an HSM
  - PCI Express plug-in cards
  - LAN appliance for immediate use in your data centers
- 100% compatible
- upgradable over time
- Performance, technical support may vary



Static WBC


password
cryptographic key
static WBC
engine
plaintext
password
C
code
password
any compiler
exe
.hex
.dll
.so
password
ciphertext

Dynamic WBC


cryptographic key
password
token generator
any compiler
exe
.hex
.dll
.so
user master key
token
password
Backend
Client
plaintext
password
dynamic WBC engine
WB module
any compiler
exe
.hex
.dll
.so
ciphertext (under)

Security Assurance

- Not vulnerable to DCA, DFA and structural attacks
- Preventing reproducibility of any analysis by customizing the WBC engine for each application
- Obtaining best performance-security trade-off by a fine-tuning of different parameters depending on your constraints
- Each delivered instance of WBC engine goes through a security evaluation performed by an accredited ITSEF security lab.
- We also provide security upgrades against new discovered attacks.

Thank you!