Thursday, 16th July 2020

Building Whiteboxes: attacks and defenses

Matthieu Daumas (me), Adrien Guinet, Philippe Teuwen

Quarkslab
www.quarkslab.com

hardwear.io
Hardware Security Conference and Training

Table of Contents

Whiteboxes: an introduction

Introduction (1/2)

- Why?

Whiteboxes: an introduction

Introduction (1/2)

- Why? "please protect my cryptographic implementation"

Whiteboxes: an introduction

Introduction (1/2)

- Why? "please protect my cryptographic implementation"
- Wait, protect what?

Whiteboxes: an introduction

Introduction (1/2)

- Why? "please protect my cryptographic implementation"
- Wait, protect what? the keys? the data?

Whiteboxes: an introduction

Introduction (1/2)

- Why? "please protect my cryptographic implementation"
- Wait, protect what? the keys? the data? from what kind of abuse?

Whiteboxes: an introduction

Introduction (1/2)

- Why? "please protect my cryptographic implementation"
- Wait, protect what? the keys? the data? from what kind of abuse?

We need to better define our context...

Whiteboxes: an introduction

Introduction (1/2)

- Why? "please protect my cryptographic implementation"
- Wait, protect what? the keys? the data? from what kind of abuse?

We need to better define our context...
...and our **attacker model**

Attacker Model

Often a client application is in an “unknown” environment:

Attacker Model

Often a client application is in an “unknown” environment:

Attacker Model

- **Full read/write access** to the binary

Attacker Model

Often a client application is in an “unknown” environment:

Attacker Model

- **Full read/write access** to the binary
- **Full control** over the **operating system** and **hardware**

Attacker Model

Often a client application is in an “unknown” environment:

Attacker Model

- **Full read/write access** to the binary
- **Full control** over the **operating system** and **hardware**
- . . . and the application runs with the least possible privileges [1]

[1] for example, in *userland* on modern x86 systems

Attacker Model

Often a client application is in an "unknown" environment:

Attacker Model

- **Full read/write access** to the binary
- **Full control** over the **operating system** and **hardware**
- ...and the application runs with the least possible privileges [1]

To sum up: think of the **worst possible situation**!

[1] for example, in *userland* on modern x86 systems

In a nutshell...

Example: standard AES

```
1  C=AES.new("this is some key")
2  C.encrypt("data to encrypt!")
```

In a nutshell...

Example: standard AES

```
1  C=AES.new("this is some key")
2  C.encrypt("data to encrypt!")
```

Example: “whiteboxed” AES

```
1  C=WBAES.new()
2  C.encrypt("data to encrypt!")
```

In a nutshell...

Example: standard AES

```
1  C=AES.new("this is some key")
2  C.encrypt("data to encrypt!")
```

Example: “whiteboxed” AES

```
1  C=WBAES.new() # extracting a key from WBAES is **hard enough**
2  C.encrypt("data to encrypt!")
```

Whiteboxes: introduction

Introduction (2/2)

- Idea: generate cryptographic algorithms that process encoded [2] keys

[2] read this as "somewhat concealed from such kind of omniscient attacker"

Whiteboxes: introduction

Introduction (2/2)

- Idea: generate cryptographic algorithms that process encoded [2] keys
- Ultimate goal: theoretically impossible to recover those keys in our attacker model

[2] read this as "somewhat concealed from such kind of omniscient attacker"

Whiteboxes: introduction

Introduction (2/2)

- Idea: generate cryptographic algorithms that process encoded [2] keys
- Ultimate goal: theoretically impossible to recover those keys in our attacker model
- In practice: a compromise between mathematics and obfuscations

[2] read this as "somewhat concealed from such kind of omniscient attacker"

Whiteboxes: introduction

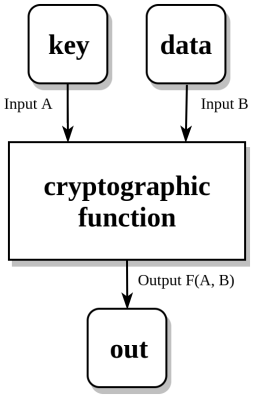
Introduction (2/2)

- Idea: generate cryptographic algorithms that process encoded [2] keys
- Ultimate goal: theoretically impossible to recover those keys in our attacker model
- In practice: a compromise between mathematics and obfuscations

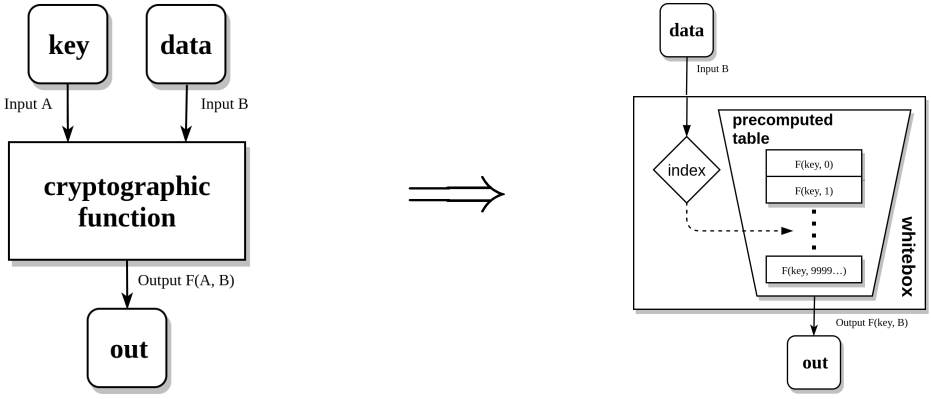
Making the attacker's work *hard enough*

[2] read this as "somewhat concealed from such kind of omniscient attacker"

Building Whiteboxes with Tables

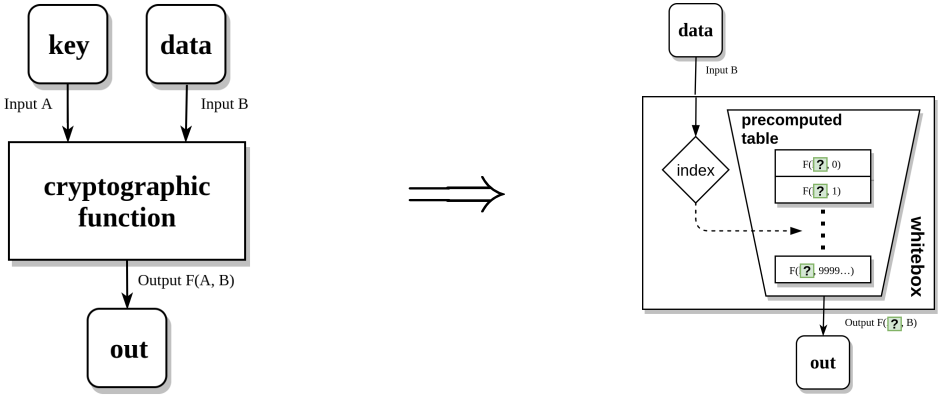


Building Whiteboxes with Tables



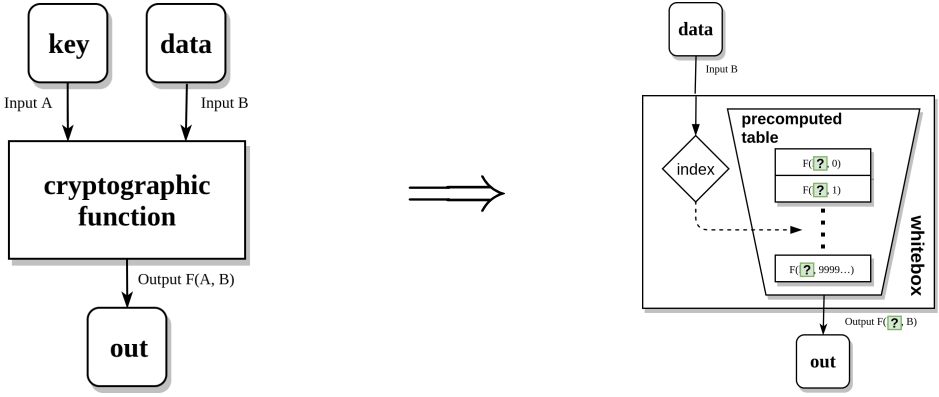
Let's **tabulate** our function!

Building Whiteboxes with Tables



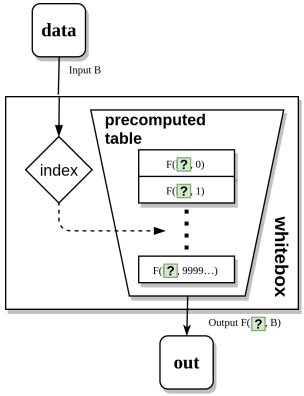
Let's **tabulate** our function!

Building Whiteboxes with Tables



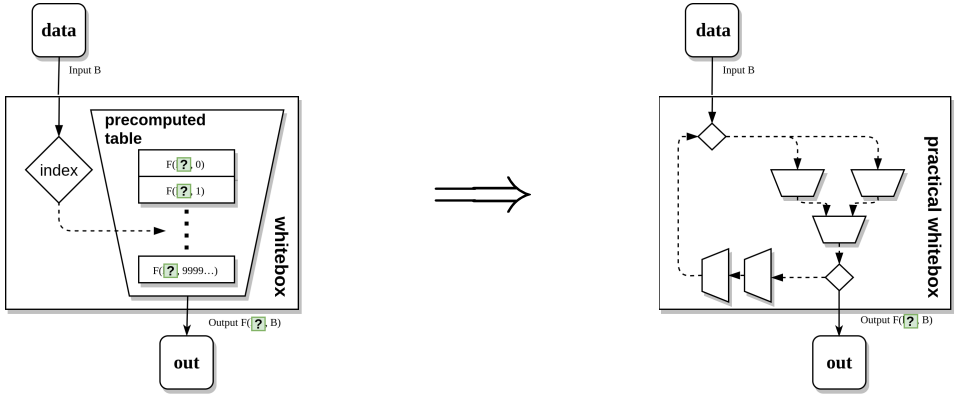
Unpractical for a **large input size**
(AES-128 $\approx 10^{24}$ petabytes table)

Building Whiteboxes with Tables



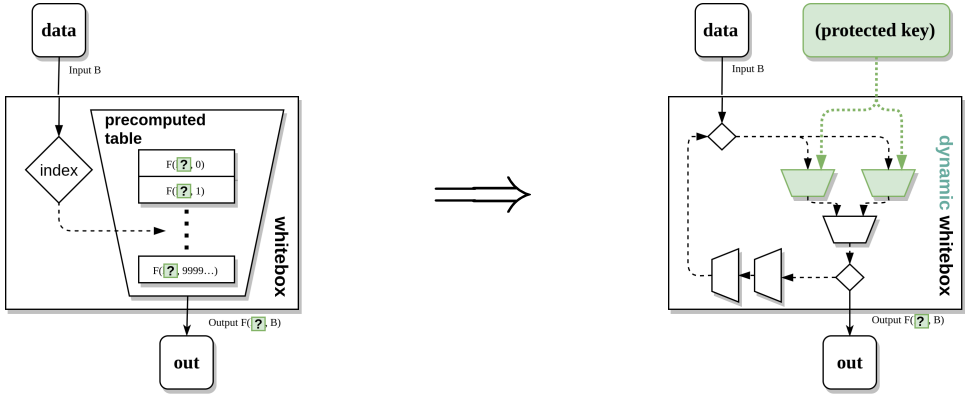
Unpractical for a **large input size**
(AES-128 $\approx 10^{24}$ petabytes table)

Building Whiteboxes with Tables



Tabulate **smaller operations** on **internal states**

Building Whiteboxes with Tables



Tabulate **smaller operations** on **internal states**
(you can even make a **dynamic** whitebox)

Building Whiteboxes: Tradeoffs

Practical whiteboxes

- Work on *encoded* internal states of the algorithm

Building Whiteboxes: Tradeoffs

Practical whiteboxes

- Work on *encoded* internal states of the algorithm
- Tabulate smaller operations that process these states

Building Whiteboxes: Tradeoffs

Practical whiteboxes

- Work on *encoded* internal states of the algorithm
- Tabulate smaller operations that process these states
- Apply software protections

Building Whiteboxes: Tradeoffs

Practical whiteboxes

- Work on *encoded* internal states of the algorithm
- Tabulate smaller operations that process these states
- Apply software protections (remember: a mix of mathematics and obfuscation)

Building Whiteboxes: Tradeoffs

Practical whiteboxes

- Work on *encoded* internal states of the algorithm
- Tabulate smaller operations that process these states
- Apply software protections (remember: a mix of mathematics and obfuscation)

Security/size/performance compromise

- Extreme 1 (very hard to break):

  The "perfect" whitebox with an **unpractical** size

Building Whiteboxes: Tradeoffs

Practical whiteboxes

- Work on *encoded* internal states of the algorithm
- Tabulate smaller operations that process these states
- Apply software protections (remember: a mix of mathematics and obfuscation)

Security/size/performance compromise

- Extreme 1 (very hard to break):

  The "perfect" whitebox with an **unpractical** size

- Extreme 2 (very easy to break):

  The **unprotected** OpenSSL AES implementation (`aes_core.c`)

Building Whiteboxes: Tradeoffs

Practical whiteboxes

- Work on *encoded* internal states of the algorithm
- Tabulate smaller operations that process these states
- Apply software protections (remember: a mix of mathematics and obfuscation)

Security/size/performance compromise

- Extreme 1 (very hard to break):

  The "perfect" whitebox with an **unpractical** size

- Extreme 2 (very easy to break):

  The **unprotected** OpenSSL AES implementation (`aes_core.c`)

The goal is to give the user the choice of a **tradeoff**!

Whiteboxes: Usage

Digital Right Managements

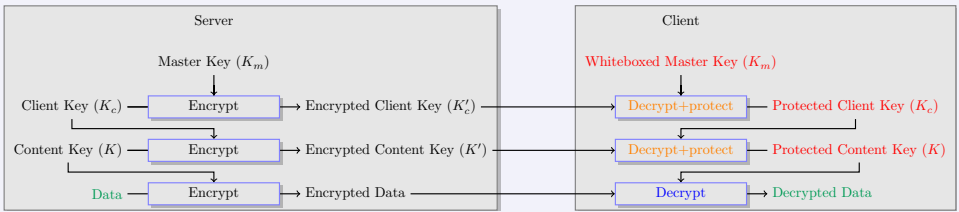
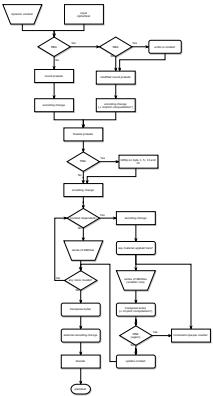


Table of Contents

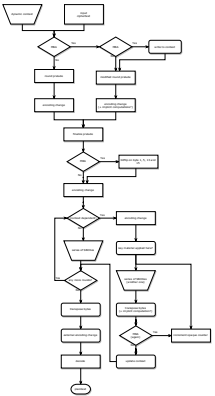
Table of Contents

Code Lifting: Example



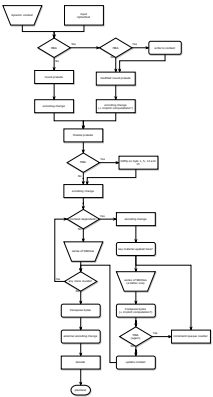
Imagine a complex whitebox decrypting a media stream...

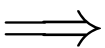
Code Lifting: Example

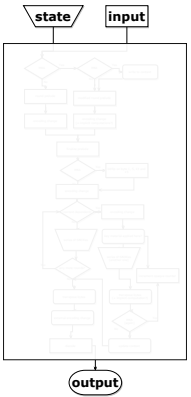


Too hard to understand! I'm a lazy attacker!

Code Lifting: Example

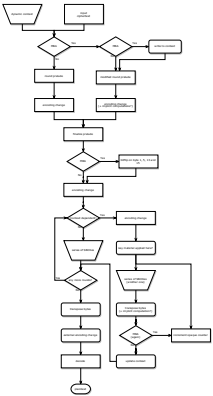


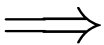


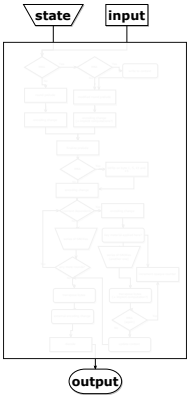


Too hard to understand! I'm a lazy attacker! No problem!

Code Lifting: Example

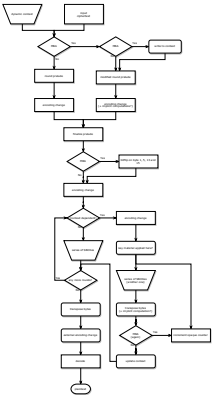


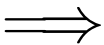


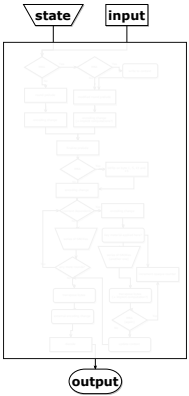


Me attacker can steal movie now,

Code Lifting: Example







Me attacker can steal movie now, without the key 

Code Lifting: In Practice

Code Lifting

- Idea: extract useful high-level primitives from the whitebox

Code Lifting: In Practice

Code Lifting

- Idea: extract useful high-level primitives from the whitebox
- Ultimate goal: use these primitives as "black boxes" to achieve the desired result [3]

[3] for example, decrypting a movie stream (without extracting the key, in this case)

Code Lifting: In Practice

Code Lifting

- Idea: extract useful high-level primitives from the whitebox
- Ultimate goal: use these primitives as "black boxes" to achieve the desired result [3]
- In practice: can compromise whitebox security while not requiring an in-depth analysis

[3] for example, decrypting a movie stream (without extracting the key, in this case)

Code Lifting: In Practice

Code Lifting

- Idea: extract useful high-level primitives from the whitebox
- Ultimate goal: use these primitives as "black boxes" to achieve the desired result [3]
- In practice: can compromise whitebox security while not requiring an in-depth analysis
- Countermeasure: "bind" in some ways the whitebox with **environment** and **client code**

[3] for example, decrypting a movie stream (without extracting the key, in this case)

Code Lifting: In Practice

Code Lifting

- Idea: extract useful high-level primitives from the whitebox
- Ultimate goal: use these primitives as "black boxes" to achieve the desired result [3]
- In practice: can compromise whitebox security while not requiring an in-depth analysis
- Countermeasure: "bind" in some ways the whitebox with **environment** and **client code**

Ask yourself: is my whitebox protected against code lifting?

[3] for example, decrypting a movie stream (without extracting the key, in this case)

Code Lifting: In Practice

Code Lifting

- Idea: extract useful high-level primitives from the whitebox
- Ultimate goal: use these primitives as "black boxes" to achieve the desired result [3]
- In practice: can compromise whitebox security while not requiring an in-depth analysis
- Countermeasure: "bind" in some ways the whitebox with **environment** and **client code**

Ask yourself: is my whitebox protected against code lifting?
. . . and protections are not trivial to implement [4]

[3] for example, decrypting a movie stream (without extracting the key, in this case)
[4] especially when integrating with large projects or third-party code

Table of Contents

The dataflow problem

In practice: whiteboxing "end-to-end"

- Cryptographic schemes involve more **than one primitive**,

The dataflow problem

In practice: whiteboxing “end-to-end”

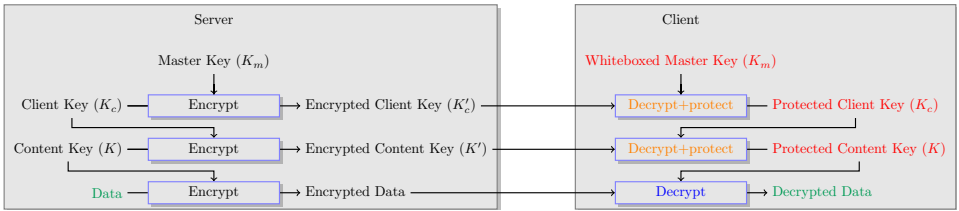
- Cryptographic schemes involve more **than one primitive**,
- with potentially digital signature and key exchange algorithms,

The dataflow problem

In practice: whiteboxing "end-to-end"

- Cryptographic schemes involve more **than one primitive**,
- with potentially digital signature and key exchange algorithms,
- and complex key derivation processes...

The dataflow problem: Example

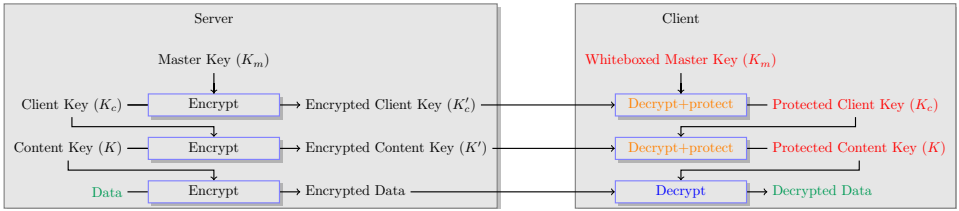


Example

In the example above:

- K_m is used to decrypt+protect K_c (also called *unwrapping*)
- K_c is used to decrypt+protect K
- K is used to decrypt the encrypted data to plaintext data

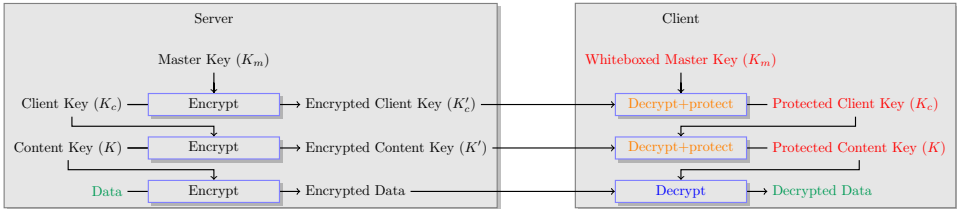
The dataflow problem: Example



In the whitebox code, we thus have **two functions**:

1. `decrypt+protect` that given protected key can decrypt then protect data
2. `decrypt` that given protected key can decrypt data

The dataflow problem: Example

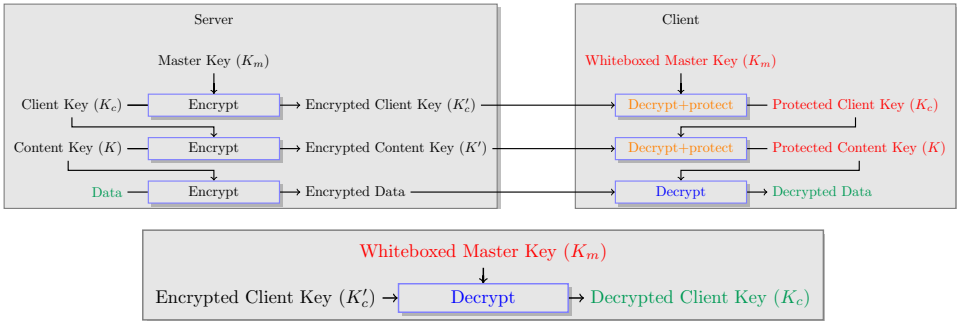


In the whitebox code, we thus have **two functions**:

1. `decrypt+protect` that given protected key can decrypt then protect data
2. `decrypt` that given protected key can decrypt data

The attacker **modifies the dataflow** to extract an unprotected copy of K_c:

The dataflow problem: Example



The dataflow problem: Solutions

- Code lifting protections only help to somewhat **mitigate** this problem
- Tweaking how each key is unwrapped can **enforce** where it is supposed to be used

The dataflow problem: Example

The dataflow problem: Solutions

- Code lifting protections only help to somewhat **mitigate** this problem
- Tweaking how each key is unwrapped can **enforce** where it is supposed to be used

Ask yourself: is my whitebox dataflow properly enforced?

Table of Contents

Design-Specific

AES was never designed to be easy to whitebox...

Design-Specific

AES was never designed to be easy to whitebox...
...20-year history of academic schemes broken by "mathematical" attacks [5]

[5] called *algebraic attacks* in the literature, each attack often focusing on particular designs

Design-Specific

AES was never designed to be easy to whitebox...
...20-year history of academic schemes broken by "mathematical" attacks [5]

"Mathematical" Attacks: In Practice

- First reverse all obfuscation layers to reach the underlying design

[5] called *algebraic attacks* in the literature, each attack often focusing on particular designs

Design-Specific

AES was never designed to be easy to whitebox...
...20-year history of academic schemes broken by "mathematical" attacks [5]

"Mathematical" Attacks: In Practice

- First reverse all obfuscation layers to reach the underlying design
- ...then **understand** and **implement** the corresponding academic papers

[5] called *algebraic attacks* in the literature, each attack often focusing on particular designs

Design-Specific

AES was never designed to be easy to whitebox...
...20-year history of academic schemes broken by "mathematical" attacks [5]

"Mathematical" Attacks: In Practice

- First reverse all obfuscation layers to reach the underlying design
- ...then **understand** and **implement** the corresponding academic papers
- ...then tweak the attack to fit the specificities of the design

[5] called *algebraic attacks* in the literature, each attack often focusing on particular designs

Design-Specific

AES was never designed to be easy to whitebox...
...20-year history of academic schemes broken by "mathematical" attacks [5]

"Mathematical" Attacks: In Practice

- First reverse all obfuscation layers to reach the underlying design
- ...then **understand** and **implement** the corresponding academic papers
- ...then tweak the attack to fit the specificities of the design

Time consuming, not off-the-shelf, large effort of reverse-engineering

[5] called *algebraic attacks* in the literature, each attack often focusing on particular designs

Design-Specific

AES was never designed to be easy to whitebox...
...20-year history of academic schemes broken by "mathematical" attacks [5]

"Mathematical" Attacks: In Practice

- First reverse all obfuscation layers to reach the underlying design
- ...then **understand** and **implement** the corresponding academic papers
- ...then tweak the attack to fit the specificities of the design

Time consuming, not off-the-shelf, large effort of reverse-engineering

Can we do **cheaper?**

[5] called *algebraic attacks* in the literature, each attack often focusing on particular designs

Greybox Attacks

Bringing back the **Greybox Attack Model** from the hardware world:

Greybox Attacks

Bringing back the **Greybox Attack Model** from the hardware world:

- By observing the power consumption of a chip

Greybox Attacks

Bringing back the **Greybox Attack Model** from the hardware world:

- By observing the power consumption of a chip
- By perturbing its behaviour with electromagnetic pulses

Greybox Attacks

Bringing back the **Greybox Attack Model** from the hardware world:

- By observing the power consumption of a chip
- By perturbing its behaviour with electromagnetic pulses

Several advantages:

- Very few assumptions on the target

Greybox Attacks

Bringing back the **Greybox Attack Model** from the hardware world:

- By observing the power consumption of a chip
- By perturbing its behaviour with electromagnetic pulses

Several advantages:

- Very few assumptions on the target
- Attack Campaign **Automation** 

Tracing

Introduced in 2015 by Bos, Hubain, Michiels & Teuwen ...

Tracing

Introduced in 2015 by Bos, Hubain, Michiels & Teuwen ...

Differential Computation Analysis

- We want to apply regular DPA (*Differential Power Analysis*)

Tracing

Introduced in 2015 by Bos, Hubain, Michiels & Teuwen ...

Differential Computation Analysis

- We want to apply regular DPA (*Differential Power Analysis*)
- Tracing an execution (record memory / register accesses) is different

Tracing

Introduced in 2015 by Bos, Hubain, Michiels & Teuwen ...

Differential Computation Analysis

- We want to apply regular DPA (*Differential Power Analysis*)
- Tracing an execution (record memory / register accesses) is different
- Serialize bits, apply regular DPA on the result

Tracing

Introduced in 2015 by Bos, Hubain, Michiels & Teuwen ...

Differential Computation Analysis

- We want to apply regular DPA (*Differential Power Analysis*)
- Tracing an execution (record memory / register accesses) is different
- Serialize bits, apply regular DPA on the result

No understanding of the design required

Tracing

Introduced in 2015 by Bos, Hubain, Michiels & Teuwen ...

Differential Computation Analysis

- We want to apply regular DPA (*Differential Power Analysis*)
- Tracing an execution (record memory / register accesses) is different
- Serialize bits, apply regular DPA on the result

No understanding of the design required

No big reverse engineering

Tracing

Today's DCA class of attacks

- Started from empirical results with simple DPA

Tracing

Today's DCA class of attacks

- Started from empirical results with simple DPA
- Completed by many papers bringing math foundations, generalizations, optimizations...

Tracing

Today's DCA class of attacks

- Started from empirical results with simple DPA
- Completed by many papers bringing math foundations, generalizations, optimizations...
- Along with many new open-source tools [6]

[6] in SideChannelMarvels, but also Hulk, Jlsca, qscat, DATA, Lascar, Rainbow...
(and several other toolboxes – some published jointly with some of these papers)

Tracing

Today's DCA class of attacks

- Started from empirical results with simple DPA
- Completed by many papers bringing math foundations, generalizations, optimizations...
- Along with many new open-source tools [6]

Many can be automated blindly

[6] in SideChannelMarvels, but also Hulk, Jlsca, qscat, DATA, Lascar, Rainbow...
(and several other toolboxes – some published jointly with some of these papers)

Tracing

Today's DCA class of attacks

- Started from empirical results with simple DPA
- Completed by many papers bringing math foundations, generalizations, optimizations...
- Along with many new open-source tools [6]

Many can be automated blindly

script-kiddie p0w3r 

[6] in SideChannelMarvels, but also Hulk, Jlsca, qscat, DATA, Lascar, Rainbow...
(and several other toolboxes – some published jointly with some of these papers)

Fault Injection

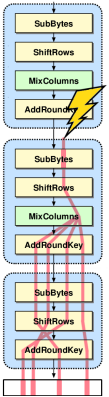
Introduced in 2015 by de Haas, Mune & Sanfelix...

Fault Injection

Introduced in 2015 by de Haas, Mune & Sanfelix...

Differential Fault Analysis

- Faults injected statically or dynamically in the last rounds

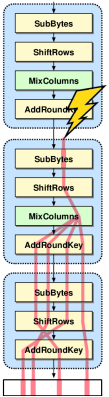


Fault Injection

Introduced in 2015 by de Haas, Mune & Sanfelix...

Differential Fault Analysis

- Faults injected statically or dynamically in the last rounds
- Today: optimized, also available as tooling

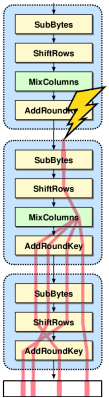


Fault Injection

Introduced in 2015 by de Haas, Mune & Sanfelix...

Differential Fault Analysis

- Faults injected statically or dynamically in the last rounds
- Today: optimized, also available as tooling
- Can be automated blindly – often easier than DFA



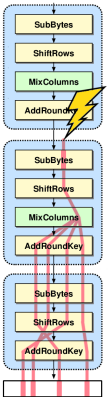
Fault Injection

Introduced in 2015 by de Haas, Mune & Sanfelix...

Differential Fault Analysis

- Faults injected statically or dynamically in the last rounds
- Today: optimized, also available as tooling
- Can be automated blindly – often easier than DFA

script-kiddie p0w3r (again)



Counter-Measures

"**Ask yourself**" checklist:

Counter-Measures

"**Ask yourself**" checklist:

- is my whitebox protected against **code lifting**?

Counter-Measures

"**Ask yourself**" checklist:

- is my whitebox protected against **code lifting**?
- is my whitebox dataflow properly **enforced**?

Counter-Measures

"**Ask yourself**" checklist:

- is my whitebox protected against **code lifting**?
- is my whitebox dataflow properly **enforced**?
- and my **AES whitebox**, is it protected against **DCA**?

Counter-Measures

"**Ask yourself**" checklist:

- is my whitebox protected against **code lifting**?
- is my whitebox dataflow properly **enforced**?
- and my **AES whitebox**, is it protected against **DCA**? and **DFA**? [7]

[7] not an exhaustive checklist for AES, other attacks exist (BGE...)

Counter-Measures

"**Ask yourself**" checklist:

- is my whitebox protected against **code lifting**?
- is my whitebox dataflow properly **enforced**?
- and my **AES whitebox**, is it protected against **DCA**? and **DFA**? [7]
- ...

[7] not an exhaustive checklist for AES, other attacks exist (BGE...)

Counter-Measures

"**Ask yourself**" checklist:

- is my whitebox protected against **code lifting**?
- is my whitebox dataflow properly **enforced**?
- and my **AES whitebox**, is it protected against **DCA**? and **DFA**? [7]
- ...
- **add other items here** (from standards: obfuscating, breaking attackers tools...)

[7] not an exhaustive checklist for AES, other attacks exist (BGE...)

Counter-Measures

"**Ask yourself**" checklist:

- is my whitebox protected against **code lifting**?
- is my whitebox dataflow properly **enforced**?
- and my **AES whitebox**, is it protected against **DCA**? and **DFA**? [7]
- ...
- **add other items here** (from standards: obfuscating, breaking attackers tools...)

If not protected **explicitly**, my whitebox will likely be an easy break!

[7] not an exhaustive checklist for AES, other attacks exist (BGE...)

Table of Contents

Enter The Void

What about whiteboxing more exotic cryptography...

Enter The Void

What about whiteboxing more exotic cryptography...

Elliptic Curve Digital Signature Algorithm

- Why?

Enter The Void

What about whiteboxing more exotic cryptography...

Elliptic Curve Digital Signature Algorithm

- Why? The need of "end-to-end" whitebox – protecting a whole protocol

Enter The Void

What about whiteboxing more exotic cryptography...

Elliptic Curve Digital Signature Algorithm

- Why? The need of "end-to-end" whitebox – protecting a whole protocol
- How to build one?

Enter The Void

What about whiteboxing more exotic cryptography...

Elliptic Curve Digital Signature Algorithm

- Why? The need of "end-to-end" whitebox – protecting a whole protocol
- How to build one? Little to no literature, but commercially available

Enter The Void

What about whiteboxing more exotic cryptography...

Elliptic Curve Digital Signature Algorithm

- Why? The need of "end-to-end" whitebox – protecting a whole protocol
- How to build one? Little to no literature, but commercially available

How to attack an "ECDSA whitebox"?

Enter The Void

What about whiteboxing more exotic cryptography...

Elliptic Curve Digital Signature Algorithm

- Why? The need of "end-to-end" whitebox – protecting a whole protocol
- How to build one? Little to no literature, but commercially available

How to attack an "ECDSA whitebox"?

Just as AES: **Inspire yourself from hardware attacks**

Enter The Void

What about whiteboxing more exotic cryptography...

Elliptic Curve Digital Signature Algorithm

- Why? The need of "end-to-end" whitebox – protecting a whole protocol
- How to build one? Little to no literature, but commercially available

How to attack an "ECDSA whitebox"?

(but first, a crash-course on ECDSA)

ECDSA

Signing with ECDSA

- To sign the message m with your private key p_k...

ECDSA

Signing with ECDSA

- To sign the message m with your private key p_k...
- First pick a **secret unique** random k, compute $r = pow(k, G)$ [8]

[8] G defined in the standard, *pow* being *double-and-add* on curve points

ECDSA

Signing with ECDSA

- To sign the message m with your private key p_k...
- First pick a **secret unique** random k, compute $r = pow(k, G)$ [8]
- Then compute $s = sign_{p_k}(k, m, r)$, publish (r, s)

[8] G defined in the standard, *pow* being *double-and-add* on curve points

ECDSA

Signing with ECDSA

- To sign the message m with your private key p_k...
- First pick a **secret unique** random k, compute $r = pow(k, G)$ [8]
- Then compute $s = sign_{p_k}(k, m, r)$, publish (r, s)
- Skip understanding the mathematics: $<$*not required to mount an attack*$>$

[8] G defined in the standard, *pow* being *double-and-add* on curve points

ECDSA

Signing with ECDSA

- To sign the message m with your private key p_k...
- First pick a **secret unique** random k, compute $r = pow(k, G)$ [8]
- Then compute $s = sign_{p_k}(k, m, r)$, publish (r, s)
- Skip understanding the mathematics: $<$*not required to mount an attack*$>$

How to attack an "ECDSA whitebox"?

[8] G defined in the standard, *pow* being *double-and-add* on curve points

ECDSA

Signing with ECDSA

- To sign the message m with your private key p_k...
- First pick a **secret unique** random k, compute $r = pow(k, G)$ [8]
- Then compute $s = sign_{p_k}(k, m, r)$, publish (r, s)
- Skip understanding the mathematics: $<$*not required to mount an attack*$>$

How to attack an "ECDSA whitebox"?

Let's read literature on **ECDSA hardware attacks**!

[8] G defined in the standard, *pow* being *double-and-add* on curve points

ECDSA vs Fault Injection

ECDSA Textbook Fault Injection

- Shooting lasers at the $s = sign_{p_k}(k, m, r)$ computation.

ECDSA vs Fault Injection

ECDSA Textbook Fault Injection

- Shooting lasers at the $s = sign_{p_k}(k, m, r)$ computation.
- From Wikipedia, you need one $s = sign_{p_k}(k, m, r)$ and one faulted $s' = sign_{p_k}(k, m', r)$

ECDSA vs Fault Injection

ECDSA Textbook Fault Injection

- Shooting lasers at the $s = sign_{p_k}(k, m, r)$ computation.
- From Wikipedia, you need one $s = sign_{p_k}(k, m, r)$ and one faulted $s' = sign_{p_k}(k, m', r)$
- Conditions to extract p_k private key:

Same k, but **different** m and m' [9]

[9] just for fun, remove your whitebox entropy source, it may always pick the same k

ECDSA vs Fault Injection

ECDSA Textbook Fault Injection

- Shooting lasers at the $s = sign_{p_k}(k, m, r)$ computation.
- From Wikipedia, you need one $s = sign_{p_k}(k, m, r)$ and one faulted $s' = sign_{p_k}(k, m', r)$
- Conditions to extract p_k private key:

Same k, but **different** m and m' [9]

ECDSA Textbook Whitebox Break?

- First, clone the state of your application before computing $sign_{p_k}(k, m, r)$

[9] just for fun, remove your whitebox entropy source, it may always pick the same k

ECDSA vs Fault Injection

ECDSA Textbook Fault Injection

- Shooting lasers at the $s = sign_{p_k}(k, m, r)$ computation.
- From Wikipedia, you need one $s = sign_{p_k}(k, m, r)$ and one faulted $s' = sign_{p_k}(k, m', r)$
- Conditions to extract p_k private key:

Same k, but **different** m and m' [9]

ECDSA Textbook Whitebox Break?

- First, clone the state of your application before computing $sign_{p_k}(k, m, r)$
- Then, compute s once, then repeat while flipping bits until you get $s' = sign_{p_k}(k, m', r)$

[9] just for fun, remove your whitebox entropy source, it may always pick the same k

ECDSA vs Fault Injection

ECDSA Textbook Fault Injection

- Shooting lasers at the $s = sign_{p_k}(k, m, r)$ computation.
- From Wikipedia, you need one $s = sign_{p_k}(k, m, r)$ and one faulted $s' = sign_{p_k}(k, m', r)$
- Conditions to extract p_k private key:

  Same k, but **different** m and m' [9]

ECDSA Textbook Whitebox Break?

- First, clone the state of your application before computing $sign_{p_k}(k, m, r)$
- Then, compute s once, then repeat while flipping bits until you get $s' = sign_{p_k}(k, m', r)$

Bonus: works on actual designs commercially available!

[9] just for fun, remove your whitebox entropy source, it may always pick the same k

ECDSA vs Basic Tracing

From Wikipedia again: value k must be kept **secret**...

ECDSA vs Basic Tracing

From Wikipedia again: value k must be kept **secret**... what about doing some tracing? 

ECDSA vs Basic Tracing

From Wikipedia again: value k must be kept **secret**... what about doing some tracing? 

Elliptic Curve Scalar Multiplication

- Computing $pow(k, G)$ with **double-and-add** leaks k in execution traces [10]

[10] in a similar way, leaking RSA private keys with **square-and-multiply** is textbook side-channel

ECDSA vs Basic Tracing

From Wikipedia again: value k must be kept **secret**... what about doing some tracing? 

Elliptic Curve Scalar Multiplication

- Computing $pow(k, G)$ with **double-and-add** leaks k in execution traces [10]
- Usual countermeasures [11] don't work out-of-the-box in a whitebox context

[10] in a similar way, leaking RSA private keys with **square-and-multiply** is textbook side-channel
[11] like windowing, also known as *double-and-add, but with chunks*

ECDSA vs Basic Tracing

From Wikipedia again: value k must be kept **secret**... what about doing some tracing? 

Elliptic Curve Scalar Multiplication

- Computing $pow(k, G)$ with **double-and-add** leaks k in execution traces [10]
- Usual countermeasures [11] don't work out-of-the-box in a whitebox context

Same side-channels, but harder to mitigate!

[10] in a similar way, leaking RSA private keys with **square-and-multiply** is textbook side-channel
[11] like windowing, also known as *double-and-add, but with chunks*

ECDSA vs Basic Tracing

From Wikipedia again: value k must be kept **secret**... what about doing some tracing? 

Elliptic Curve Scalar Multiplication

- Computing $pow(k, G)$ with **double-and-add** leaks k in execution traces [10]
- Usual countermeasures [11] don't work out-of-the-box in a whitebox context

Same side-channels, but harder to mitigate!

Spoilers: ECDSA tricks you don't know about
(the last one will surprise you)

[10] in a similar way, leaking RSA private keys with **square-and-multiply** is textbook side-channel
[11] like windowing, also known as *double-and-add, but with chunks*

Whitebox. All. The. Things.

How to build a whitebox of anything?

Whitebox. All. The. Things.

How to build a whitebox of anything?

Building Whiteboxes

- Learn everything you can about attacks (doing faults, tracing leaks, . . .)

Whitebox. All. The. Things.

How to build a whitebox of anything?

Building Whiteboxes

- Learn everything you can about attacks (doing faults, tracing leaks, . . .)
- **Not all algorithms are created equal!** [12]

[12] some interact poorly with the whitebox context – whiteboxing those may not be *a good idea*

Whitebox. All. The. Things.

How to build a whitebox of anything?

Building Whiteboxes

- Learn everything you can about attacks (doing faults, tracing leaks, . . .)
- **Not all algorithms are created equal!** [12]
- You work with limited knowledge, same as the attackers, and you must stay ahead!

[12] some interact poorly with the whitebox context – whiteboxing those may not be *a good idea*

Whitebox. All. The. Things.

How to build a whitebox of anything?

Building Whiteboxes

- Learn everything you can about attacks (doing faults, tracing leaks, . . .)
- **Not all algorithms are created equal!** [12]
- You work with limited knowledge, same as the attackers, and you must stay ahead!
- Such R&D has a cost,

[12] some interact poorly with the whitebox context – whiteboxing those may not be *a good idea*

Whitebox. All. The. Things.

How to build a whitebox of anything?

Building Whiteboxes

- Learn everything you can about attacks (doing faults, tracing leaks, . . .)
- **Not all algorithms are created equal!** [12]
- You work with limited knowledge, same as the attackers, and you must stay ahead!
- Such R&D has a cost, trickery & whitebox-fu required,

[12] some interact poorly with the whitebox context – whiteboxing those may not be *a good idea*

Whitebox. All. The. Things.

How to build a whitebox of anything?

Building Whiteboxes

- Learn everything you can about attacks (doing faults, tracing leaks, . . .)
- **Not all algorithms are created equal!** [12]
- You work with limited knowledge, same as the attackers, and you must stay ahead!
- Such R&D has a cost, trickery & whitebox-fu required, attackers don't give feedback[13]

[12] some interact poorly with the whitebox context – whiteboxing those may not be *a good idea*
[13] adorable, 5-star, very cosy, would pwn again

Whitebox. All. The. Things.

How to build a whitebox of anything?

Building Whiteboxes

- Learn everything you can about attacks (doing faults, tracing leaks, . . .)
- **Not all algorithms are created equal!** [12]
- You work with limited knowledge, same as the attackers, and you must stay ahead!
- Such R&D has a cost, trickery & whitebox-fu required, attackers don't give feedback[13]

You must care about
protecting the thing that protects your secrets

[12] some interact poorly with the whitebox context – whiteboxing those may not be *a good idea*
[13] adorable, 5-star, very cosy, would pwn again

Table of Contents

Conclusion

Fig. 1: whitebox cryptography

Conclusion

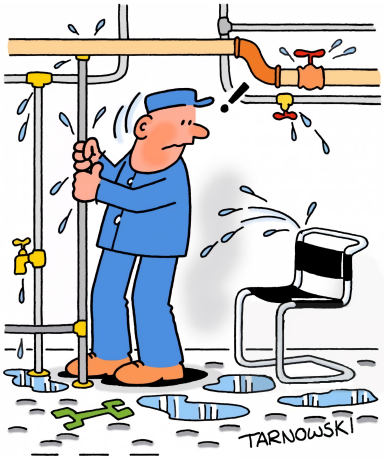


Fig. 1: whitebox cryptography
(without proper software protections)

Conclusion

Status of whiteboxes

- Multiple industrial-ready solutions,

Conclusion

Status of whiteboxes

- Multiple industrial-ready solutions, now found in global standards (e.g. PCI SPoC)

Conclusion

Status of whiteboxes

- Multiple industrial-ready solutions, now found in global standards (e.g. PCI SPoC)
- Ask the right questions,

Conclusion

Status of whiteboxes

- Multiple industrial-ready solutions, now found in global standards (e.g. PCI SPoC)
- Ask the right questions, play the cat'n'mouse game: *audit*, *break*, *fix*, *repeat*

Conclusion

Status of whiteboxes

- Multiple industrial-ready solutions, now found in global standards (e.g. PCI SPoC)
- Ask the right questions, play the cat'n'mouse game: *audit, break, fix, repeat*

Mixed Results: Hardware Security

- Provide hardware isolation for cryptographic operations, but some fragmentation

Conclusion

Status of whiteboxes

- Multiple industrial-ready solutions, now found in global standards (e.g. PCI SPoC)
- Ask the right questions, play the cat'n'mouse game: *audit, break, fix, repeat*

Mixed Results: Hardware Security

- Provide hardware isolation for cryptographic operations, but some fragmentation
- Often put behind closed doors for security reasons (restricted to some editors only...)

Conclusion

Status of whiteboxes

- Multiple industrial-ready solutions, now found in global standards (e.g. PCI SPoC)
- Ask the right questions, play the cat'n'mouse game: *audit, break, fix, repeat*

Mixed Results: Hardware Security

- Provide hardware isolation for cryptographic operations, but some fragmentation
- Often put behind closed doors for security reasons (restricted to some editors only...)

You need to protect the thing that protects your secrets...

Conclusion

Status of whiteboxes

- Multiple industrial-ready solutions, now found in global standards (e.g. PCI SPoC)
- Ask the right questions, play the cat'n'mouse game: *audit, break, fix, repeat*

Mixed Results: Hardware Security

- Provide hardware isolation for cryptographic operations, but some fragmentation
- Often put behind closed doors for security reasons (restricted to some editors only...)

You need to protect the thing that protects your secrets...
...and **applications are best protected as a whole**

Conclusion

Status of whiteboxes

- Multiple industrial-ready solutions, now found in global standards (e.g. PCI SPoC)
- Ask the right questions, play the cat'n'mouse game: *audit, break, fix, repeat*

Mixed Results: Hardware Security

- Provide hardware isolation for cryptographic operations, but some fragmentation
- Often put behind closed doors for security reasons (restricted to some editors only...)

You need to protect the thing that protects your secrets...
...and **applications are best protected as a whole**

Questions?
contact@quarkslab.com

Contact: contact@quarkslab.com

SECURING EVERY BIT OF YOUR DATA