Strengthening Sequential Side-Channel Attacks Through Change Detection

Luca Frittoli, Matteo Bocchi, Silvia Mella, Diego Carrera, Beatrice Rossi, Pasqualina Fragneto, Ruggero Susella and Giacomo Boracchi

What are sequential attacks?

These **side-channel attacks** target cryptographic algorithms that process the secret key **one bit at a time**

Algorithm 1 Target algorithm (decryption function)

Input: ciphertext c, secret key d
Output: original message m

```
O_1 is initialized
for k = 1 : K do
    O_{k+1} ← operations(d[k], O_k, c)
end for
return m ← O_{K+1}
```

What are sequential attacks?

Sequential attacks find the key bits by **reconstructing** the algorithm steps using a **distinguisher** function (e.g. a correlation coefficient)

Algorithm 2 Sequential attack

Input: target algorithm, ciphertext c, side channel $\{\mathbf{L}_k\}_{k=1}^{K}$, distinguisher $\mathcal{D}$

Output: estimated secret key $\hat{d}$

$\widehat{O}_1$ is initialized as in Algorithm 1

for $k = 1 : K$ **do**

  $\hat{d}[k] \leftarrow \arg\max_{\mathbf{x}\in\mathbf{X}} \mathcal{D}(\mathbf{x}, \widehat{O}_k, \mathbf{L}_k)$

  $\widehat{O}_{k+1} \leftarrow \texttt{operations}(\hat{d}[k], \widehat{O}_k, c)$

end for

return $\hat{d} \leftarrow (\hat{d}[1], \ldots, \hat{d}[K])$

Error propagation

Errors in sequential attacks propagate into the following steps

The bits guessed **after an error** might as well be chosen randomly

Algorithm 2 Sequential attack

Input: target algorithm, ciphertext c, side channel $\{\mathbf{L}_k\}_{k=1}^{K}$, distinguisher $\mathcal{D}$

Output: estimated secret key $\hat{d}$

$\widehat{O}_1$ is initialized as in Algorithm 1

for $k = 1 : K$ **do**

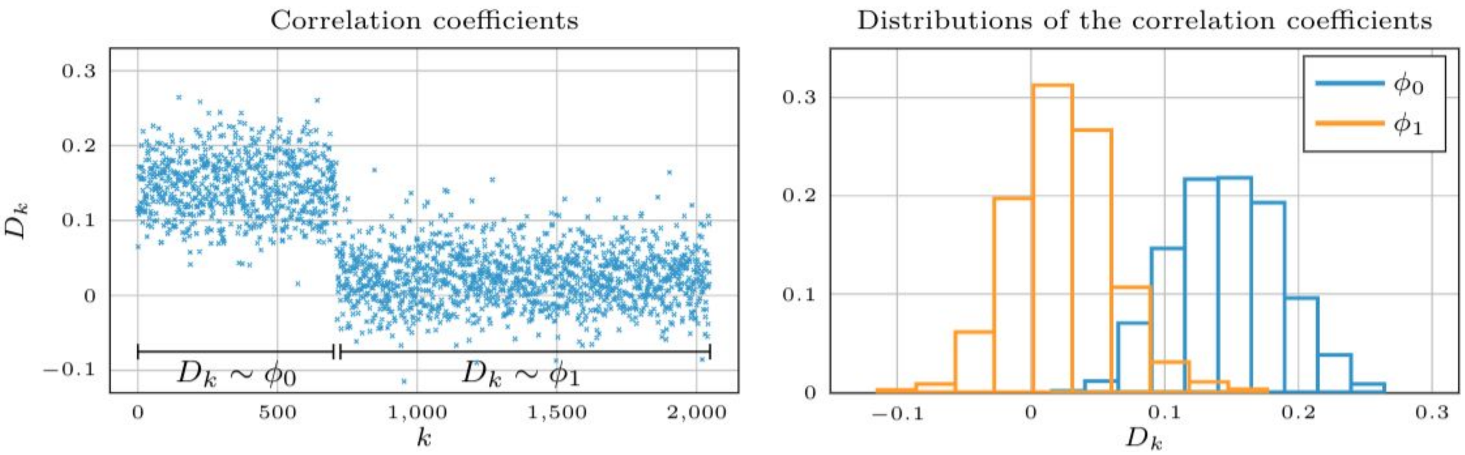
$\quad \hat{d}[k] \leftarrow \arg\max_{\mathbf{x} \in \mathbf{X}} \mathcal{D}(\mathbf{x}, \widehat{O}_k, \mathbf{L}_k)$

$\quad \widehat{O}_{k+1} \leftarrow \texttt{operations}(\hat{d}[k], \widehat{O}_k, c)$

end for

return $\hat{d} \leftarrow (\hat{d}[1], \ldots, \hat{d}[K])$

Can error propagation help attackers?



Yes: error propagation is an "**error-detection property**" [1]

The **sequence of distinguisher** values can tell where the error occurred

[1] Kocher "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems", Annual International Cryptology Conference, 1996

Related work

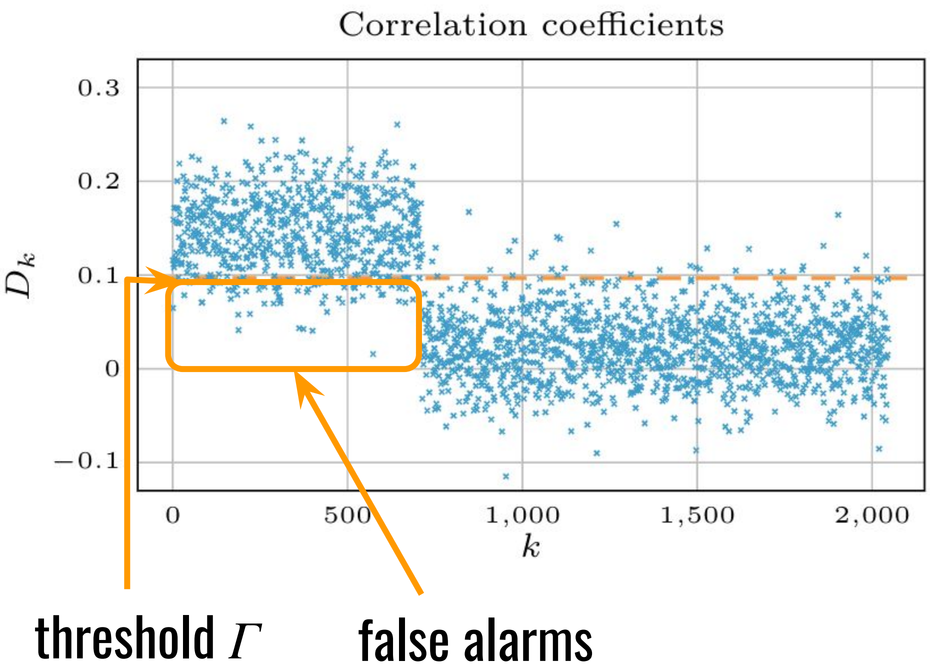
Related work

In 1996, Kocher called error propagation an "**error-detection property**" [1]

After that, several **error-detection** techniques have been proposed, either **ad-hoc** for specific attacks [2] or based on **thresholds** [3,4,5]

[1] Kocher "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems", Annual International Cryptology Conference, 1996
[2] Schindler, Koeune & Quisquater "Improving divide and conquer attacks against cryptosystems by better error detection/correction strategies", IMA International Conference on Cryptography and Coding, 2001
[3] Dhem, Koeune, Leroux, Mestré & Quisquater "A practical implementation of the timing attack", CARDIS 1998
[4] Chen, Wang & Tian "Improving timing attack on RSA-CRT via error detection and correction strategy", Information Sciences, 2013
[5] Luo, Fen & Kaeli "GPU acceleration of RSA is vulnerable to side-channel timing attacks", ICCAD 2018

State of the art: error detection by thresholding



When $D_k < \Gamma$, an error is detected
This approach can detect only **strong changes** and is subject to **false alarms**

Sometimes **the detection is confirmed** only when $D_k < \Gamma$ for a few iterations, but there are **no guarantees** on the **false alarm probability**

Our motivation

The strong **limitations** of state-of-the art solutions motivated us to **investigate** this problem more deeply

Our experience in **datastream analysis** suggested a **better solution** to **strengthen** sequential attacks

Problem formulation

Problem formulation

We address two main problems:

- **error detection**, i.e. estimating the first error location

$$\tau = \min\{k : \hat{d}[k] \neq d[k]\}$$

- **error correction**, i.e. correcting the first error using its estimated location $\hat{\tau}$
  Note that the detection might be **inaccurate**, or even a **false positive**

Proposed solution

Proposed solution

Our work is based on a **statistical analysis** of the distinguisher sequence:

- we propose an automatic **error detection** technique, using an **online change-detection test** on the distinguisher sequence to estimate the first error location
- we propose an **error correction** procedure based on a **brute-force search** over a small window centered at the **detected change point**, and using a **statistical test** on the distinguisher sequence to select the correct combination

Proposed solution

Algorithm 2 Sequential attack

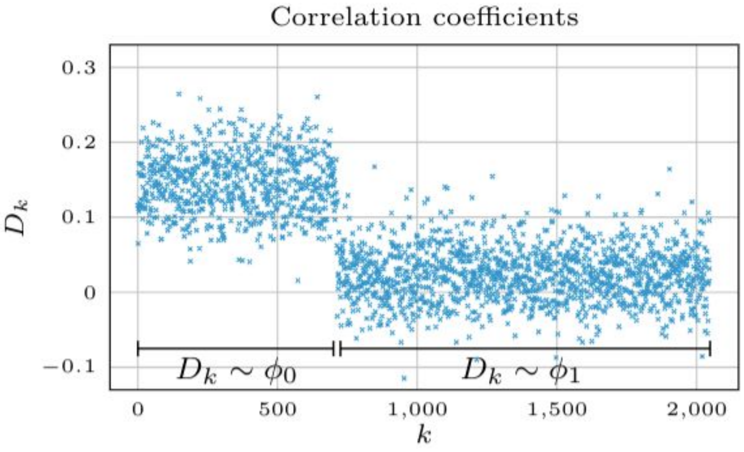
Input: target algorithm, ciphertext c, side channel $\{\mathbf{L}_k\}_{k=1}^{K}$, distinguisher $\mathcal{D}$

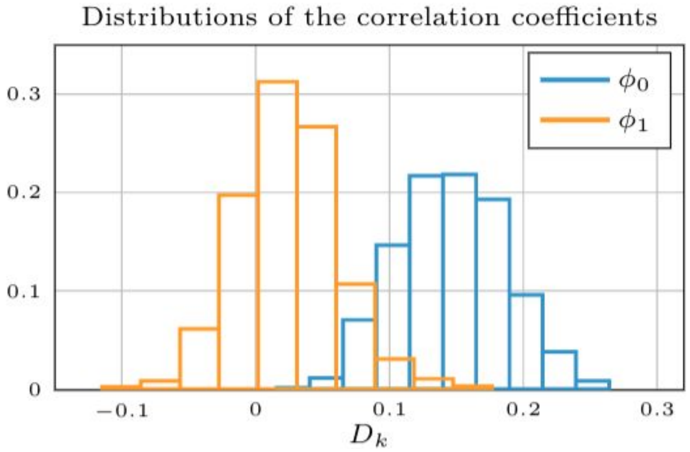
Output: estimated secret key $\hat{d}$

```
Ô_1 is initialized as in Algorithm 1
for k = 1 : K do
    d̂[k] ← arg max_{x∈X} D(x, Ô_k, L_k)
    Ô_{k+1} ← operations(d̂[k], Ô_k, c)
end for
return d̂ ← (d̂[1], ..., d̂[K])
```

- **error detection**: online change detection test on distinguisher sequence D_1, … D_k
- if a change point is found at $\hat{\tau}$, **error correction**: brute force around $\hat{\tau}$ and analysis of D_{k+1}, … to select the correct combination

Error detection

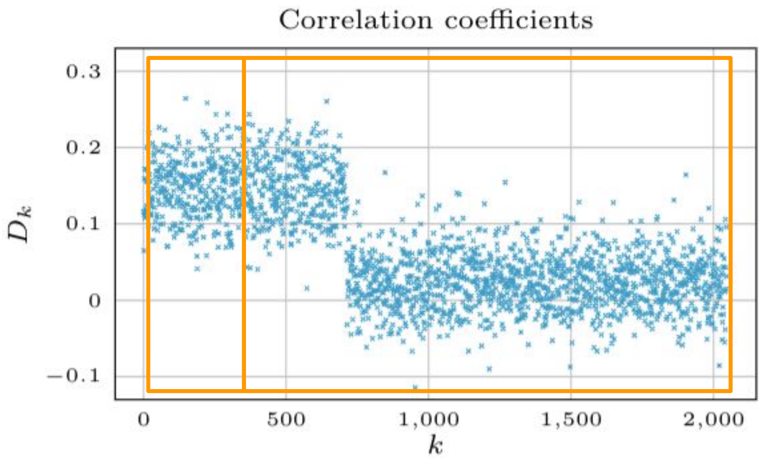


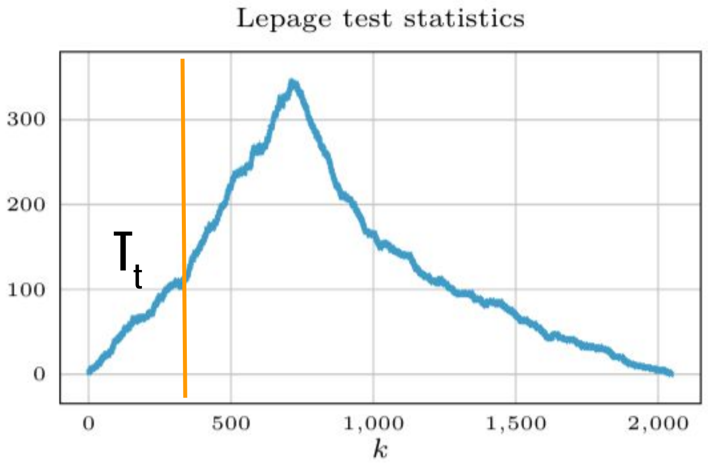


To overcome the limitations of thresholds, we use a **statistical test** that:

- can detect **slight changes**
- controls **false alarms**

Error detection

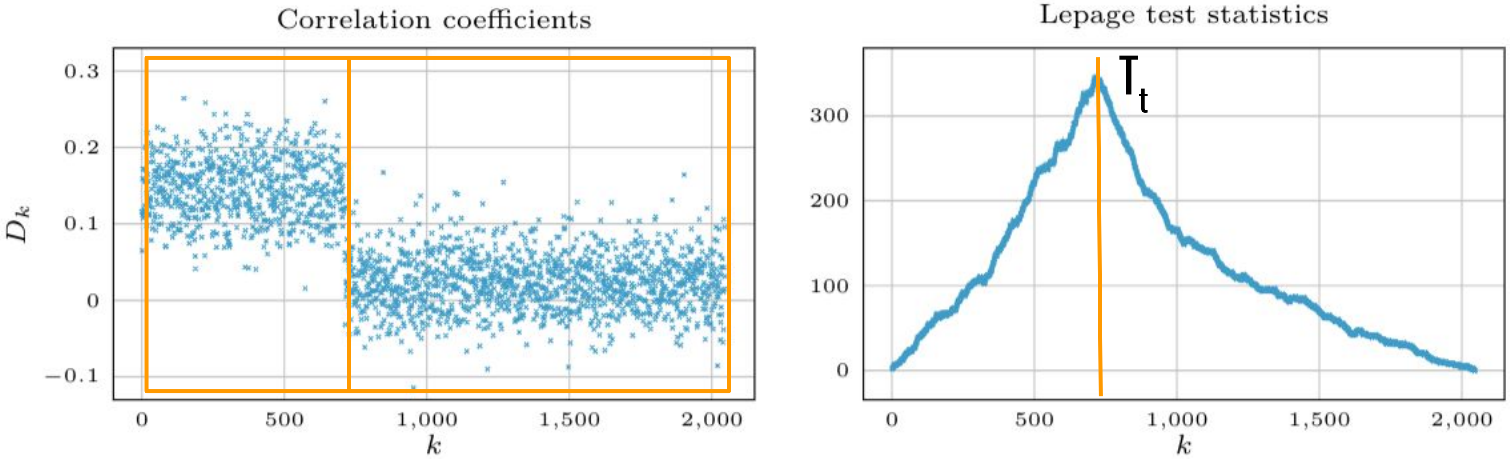




We monitor the sequence D_1, ...D_k using a **Change Point Model** (CPM) with the **Lepage test statistic** [2]

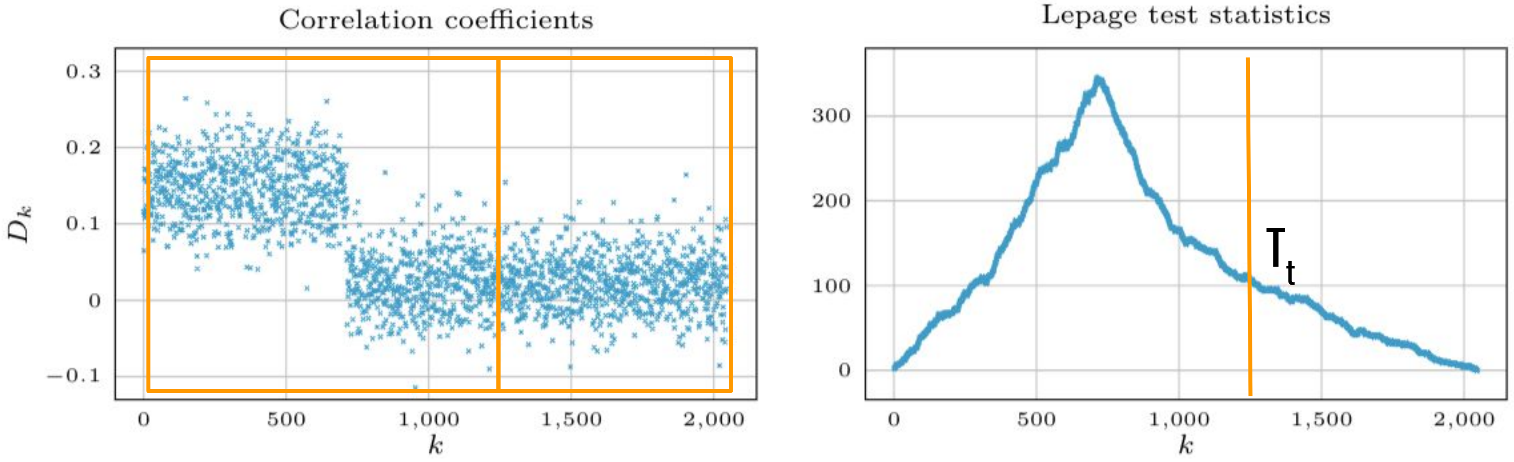
[2] Ross, Tasoulis & Adams "Nonparametric monitoring of data streams for changes in location and scale" Technometrics, 2011

Error detection



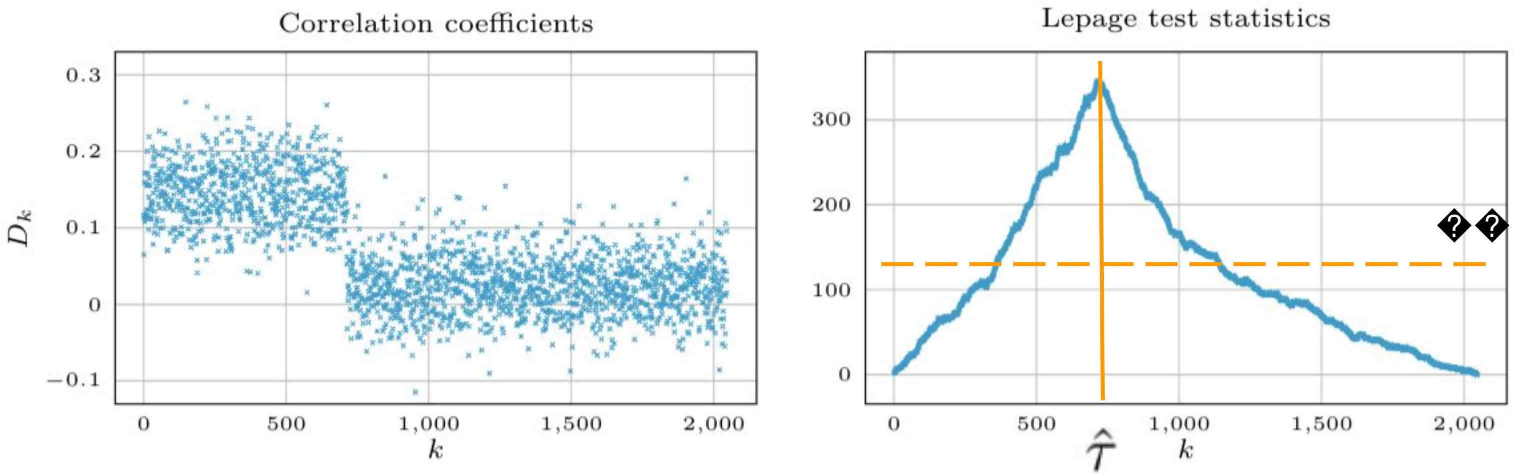
The statistic T_t **compares the distribution** of two consecutive windows separated by index $t < k$

Error detection



The statistic T_t **compares the distribution** of two consecutive windows separated by index $t < k$

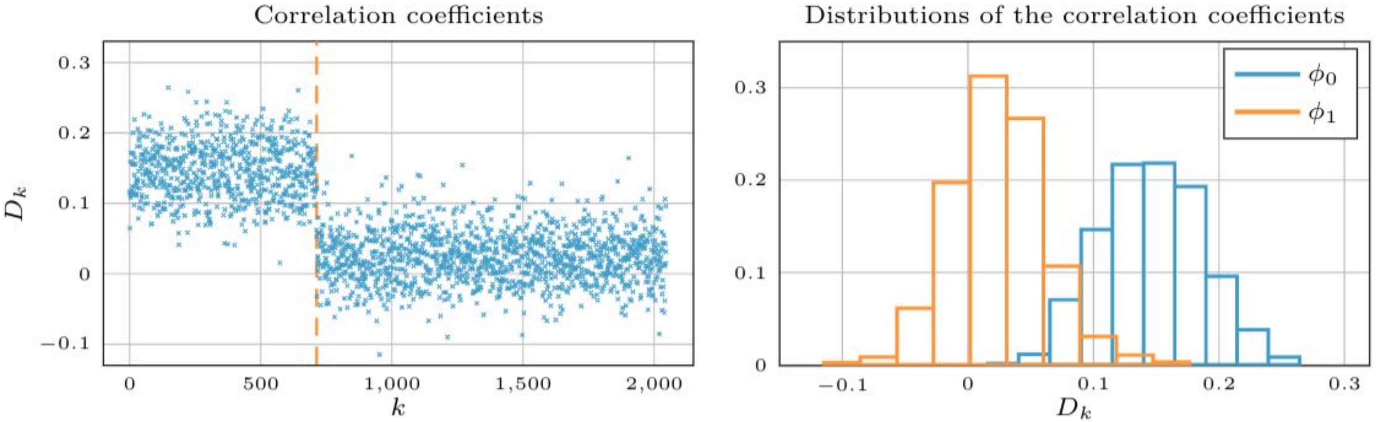
Error detection



When $\max_{t<k} T_t > \Gamma$, a **change point** is detected at the corresponding index

When a change is flagged in $D_1, \dots D_k$, **an error is detected**

Error detection



The CPM uses the Lepage test statistic to **compare the distributions** of consecutive windows within a sequence. When a change is detected, the test provides **an estimate of the change point**

The CPM is a statistical test that has a **control over false alarms**

Error correction

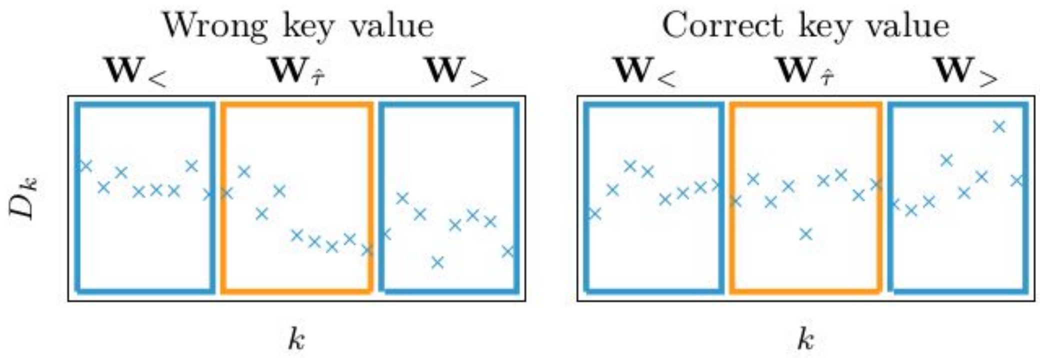
Algorithm 4 Correction procedure

Input: target algorithm, ciphertext c, side channel $\{\mathbf{L}_k\}_{k=1}^{K}$, distinguisher $\mathcal{D}$, change point $\hat{\tau}$, $\mathbf{W}_{\hat{\tau}}$ with size $w = 2u + 1$, distinguisher sequence $\mathbf{D}$, predicted output $\widehat{O}_{\hat{\tau}-u}$

Output: correction goodness (succ_correction), best estimated key $\mathbf{x}_{\text{best}}$ over $\mathbf{W}_{\hat{\tau}}$

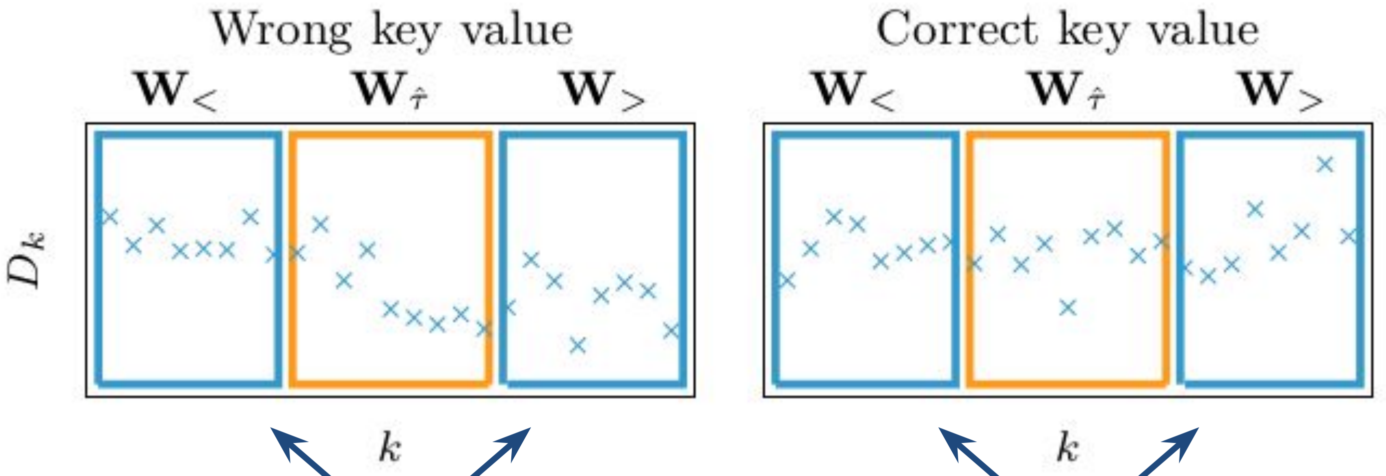
1: **for** $\mathbf{x} \in \mathbf{X}^w$ **do**
2: set $(\hat{d}^{\mathbf{x}}[\hat{\tau} - u], \ldots, \hat{d}^{\mathbf{x}}[\hat{\tau} + u]) = \mathbf{x}$ // initialization
3: compute $\widehat{O}^{\mathbf{x}}_{\hat{\tau}-u+1}, \ldots, \widehat{O}^{\mathbf{x}}_{\hat{\tau}+u+1}$ using `operations` // as in Algorithm 2, line 4
4: restart the attack from step $k = \hat{\tau} + u + 1$
5: select the two windows $\mathbf{W}_< \leftarrow \{D_k\}_{k<\hat{\tau}-u}$, $\mathbf{W}_> \leftarrow \{D^{\mathbf{x}}_k\}_{k>\hat{\tau}+u}$
6: run the statistical test $\mathcal{S}(\mathbf{W}_<, \mathbf{W}_>)$
7: **if** the test yields enough statistical evidence **then**
8: **return** true, $\mathbf{x}$
9: **end if**
10: **end for**
11: **return** false, the $\mathbf{x}$ maximizing the statistic in line 6

Error correction



When a change is detected, we propose to **correct** the error by a **brute-force search** over a window centered at the **detected change point**

Error correction



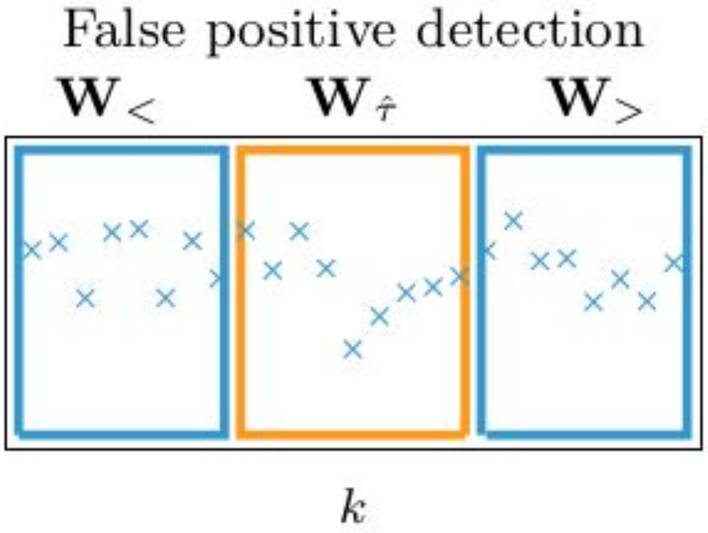
different distributions

similar distributions

Error correction: handling false positives

To handle **false alarms** raised by the CPM, we **exclude the brute-force window** from the monitoring

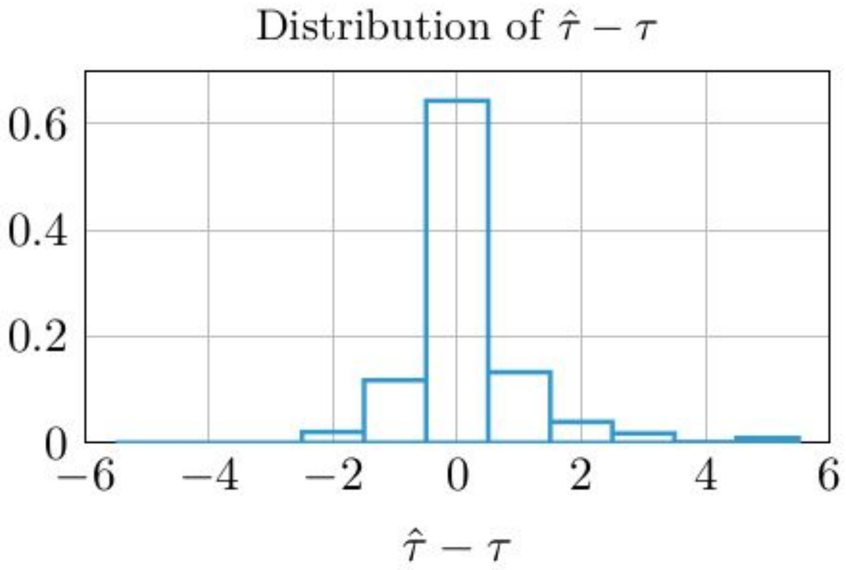
We exclude the distinguisher values leading to the false positive detection



Error correction: managing the brute-force window size

We start from a **small window** and increase its size when the **correction is unsuccessful**

We increase the window size **only when it is necessary**, reducing the overall computational cost



Two strengthened sequential attacks

Horizontal Correlation Power Analysis (H-CPA) against RSA [3]

Algorithm 5 Square and multiply always exponentiation (left-to-right)

Input: ciphertext c, key d, modulus n
Output: $m = c^d \bmod n$

$R \leftarrow 1$
for $k = 1 : K$ **do**
  $R \leftarrow R^2 \bmod n$
  if $d[k] = 1$ **then**
    $R \leftarrow R \cdot c \bmod n$
  else
    $\text{aux} \leftarrow R \cdot c \bmod n$
  end if
end for
return $m \leftarrow R$

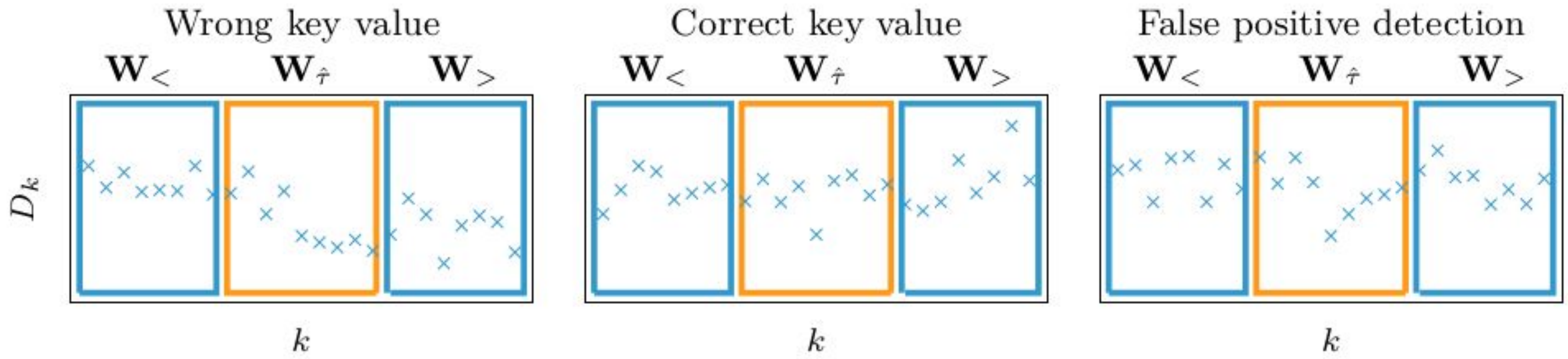
The attacker simulates the **intermediate products** depending on the key bits

The distinguisher is the **correlation coefficient** between the Hamming weights of the **operand**'s words and the **power consumption**

[3] Clavier, Feix, Gagnerot, Roussellet & Verneuil "Horizontal correlation analysis on exponentiation", ICICS 2010

Strengthened H-CPA

For **each combination** in the brute-force search, we **continue the attack** for 30 steps, and use the Mann-Whitney statistic to test the distribution of the distinguisher **before and after the brute-force window**



Timing Attack against RSA [4]

For each possible value of the **sliding window**, the attacker determines whether a **subtraction** is done in the **Montgomery** multiplication, depending on ciphertext c

The **distinguisher** is the **difference** between the **average computation times** with and without subtraction

[4] Dhem, Koeune, Leroux, Mestré & Quisquater "A practical implementation of the timing attack", CARDIS 1998

Algorithm 7 Sliding window exponentiation

Input: ciphertext c, key d, modulus n
Output: $m = c^d \bmod n$

for $j = 0 : 7$ **do**
  $Q[j] \leftarrow (j+8) \cdot c \bmod n$
end for
$R \leftarrow 1$
for $k = 1 : K$ **do**
  if $d[k] = 1$ **then**
    $R \leftarrow R^{16} \bmod n$
    $xyz \leftarrow (d[k+1], d[k+2], d[k+3])$
    $R \leftarrow R \cdot Q[xyz] \bmod n$
    $k \leftarrow k + 3$
  else
    $R \leftarrow R^2 \bmod n$
  end if
end for
return $m \leftarrow R$

Strengthened Timing Attack

In this case we consider an **open right window** $W_>$ due to the **computational cost** of each attacking step

For **each combination** of the brute-force window, we **continue the attack** and the monitoring with the CPM **until a new change point** is found

When the **new change point is greater** than the previous one, the corresponding combination is selected, otherwise a new combination is tested

Experiments

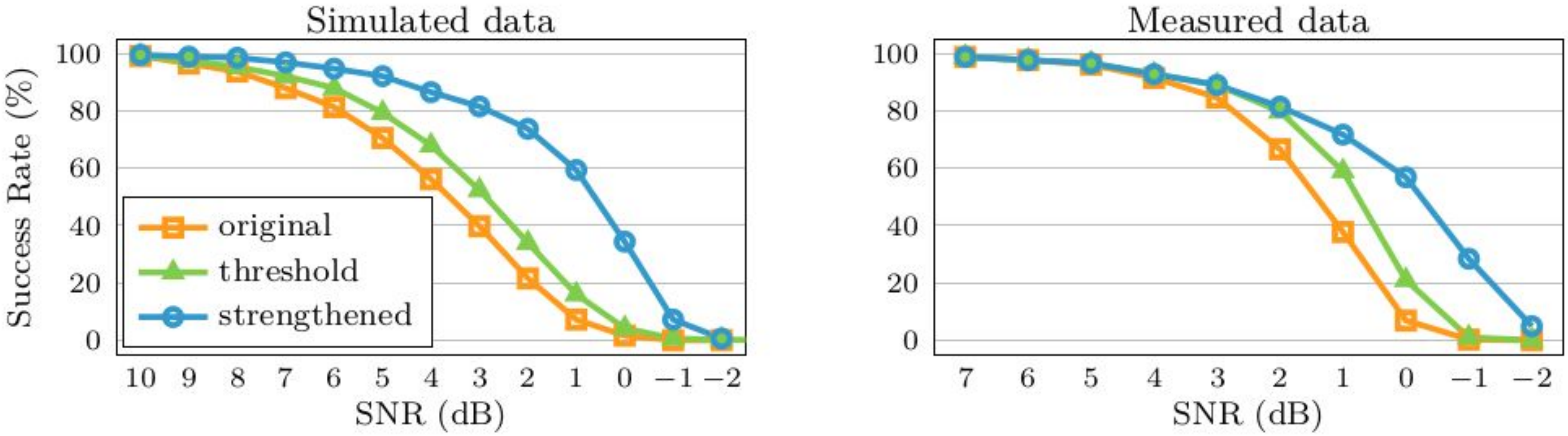
Experiments on H-CPA

We tested our strengthened H-CPA on two sets of **power consumption data**:

- two **simulated** traces of RSA-2048 using the schoolbook multiplication, obtained by Synopsys PrimeTime
- ten **real** traces of RSA-2048 using Montgomery multiplication, measured by ChipWhisperer®-Pro

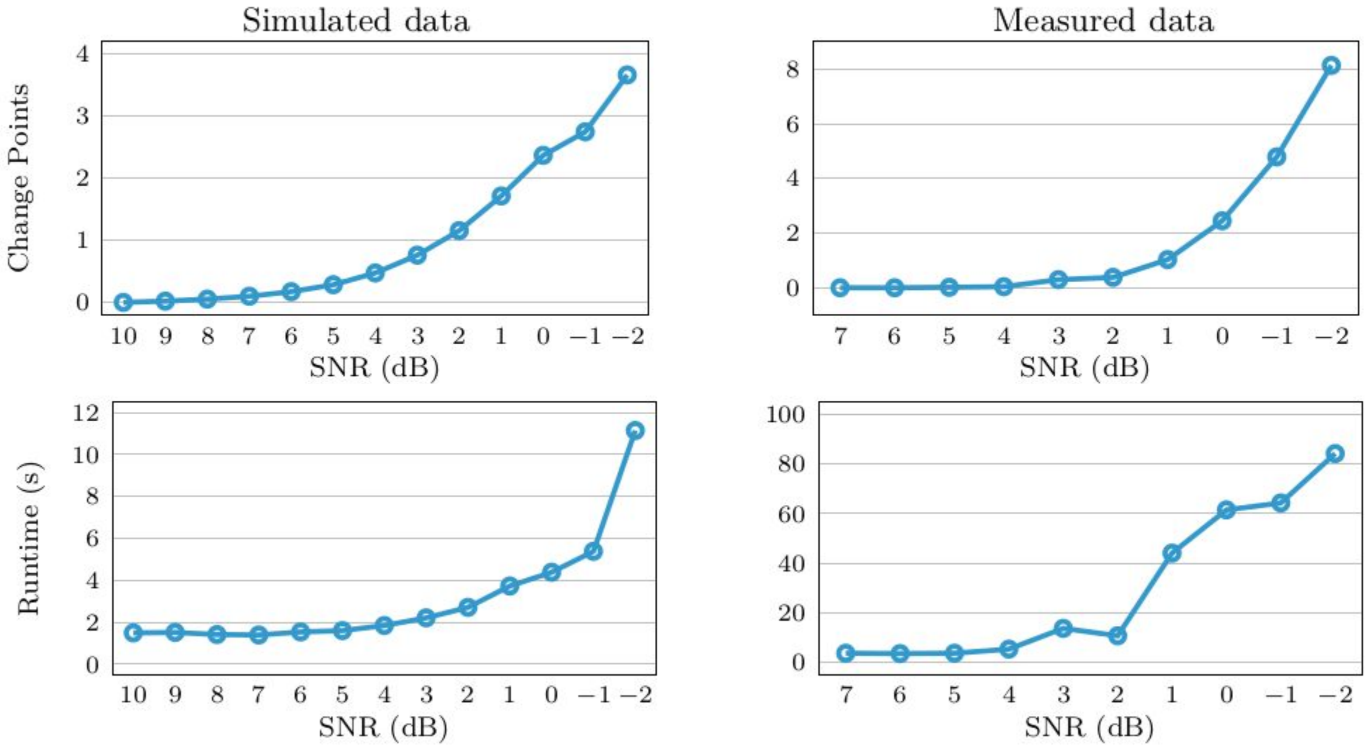
We artificially added **Gaussian noise** to obtain more realistic **Signal-to-Noise Ratios** (SNR)

Experimental results on H-CPA



We compare the success rates of our **strengthened H-CPA** to those obtained using a **state-of-the-art error detector** based on thresholds

Experimental results on H-CPA



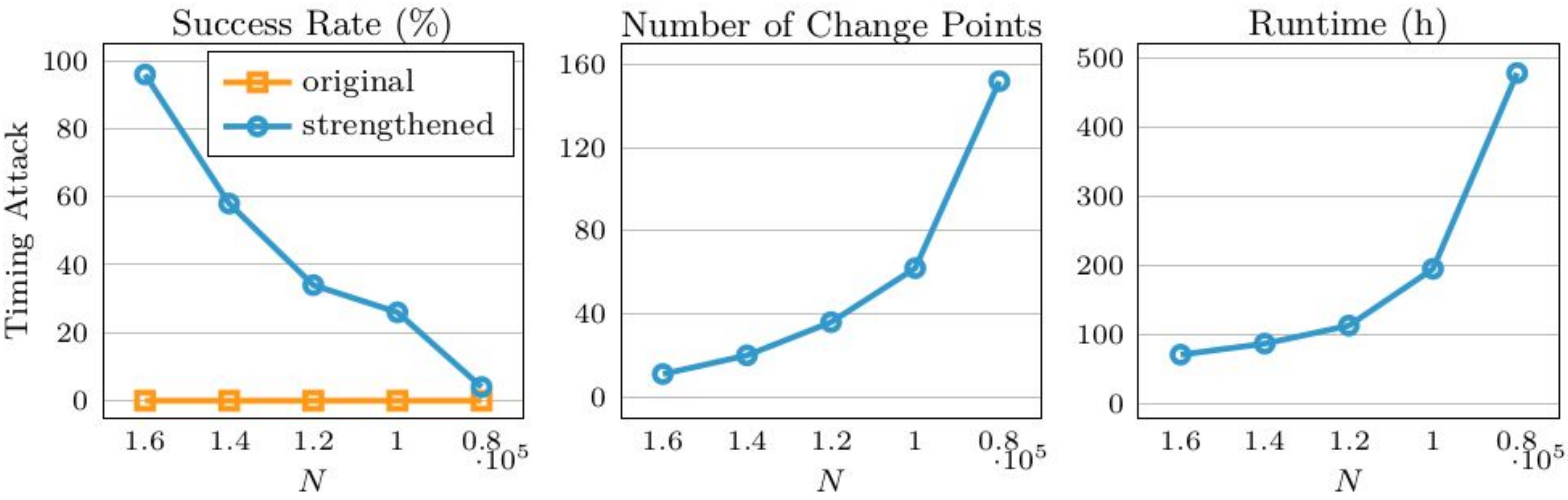
Experiments on Timing Attack

We tested our strengthened Timing Attack using large sets of **ciphertexts**, along with the respective **exponentiation times**

Timing measurements were taken on a Cortex®-M7 **microcontroller**

The **amount of measurements** N used for the attack has the same role of the SNR in H-CPA

Experimental results on Timing Attack

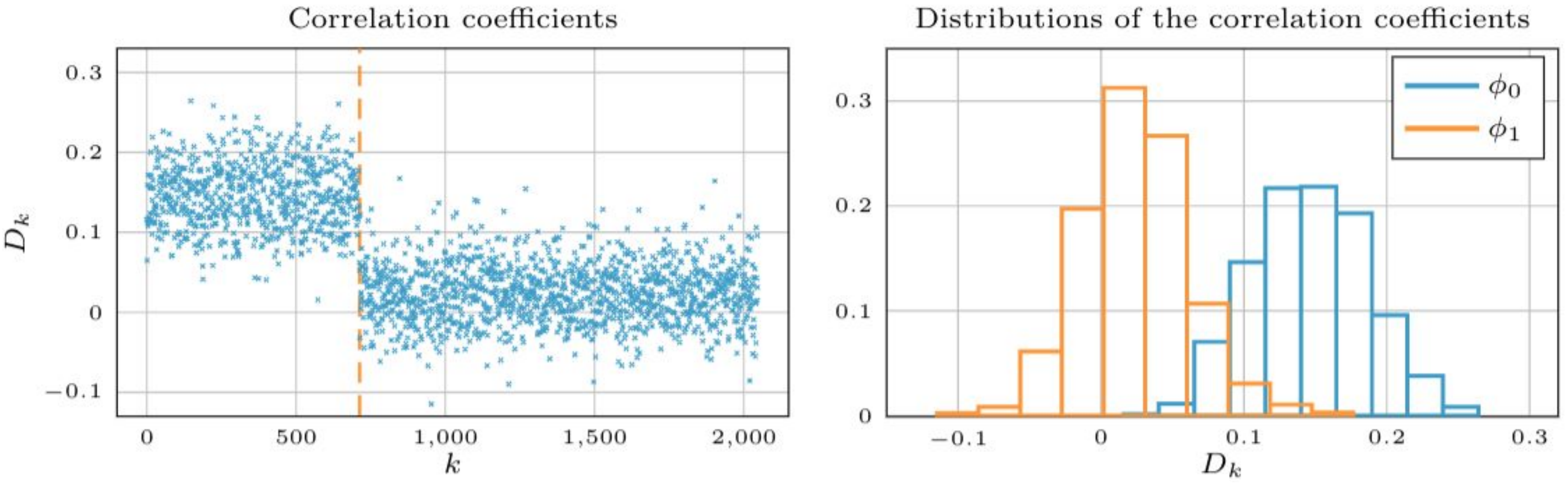


Conclusions

We introduce a general **error detection and correction** methodology for sequential attacks, based on **statistically sound** change-detection tests

We show that our **strengthened attacks** perform **significantly better** than their original counterparts, while **threshold-based techniques** yield only a **marginal improvement**

Our findings show that **blinding countermeasures** should be employed whenever possible, even when sequential attacks are considered infeasible



Thanks for your attention