Side-channel Attacks on Blinded Scalar Multiplications Revisited

Thomas Roche[1] Laurent Imbert[2] Victor Lomné[1]

NinjaLab, Montpellier, France (https://ninjalab.io)

LIRMM, CNRS, University of Montpellier, Montpellier, France

CARDIS

November 12, 2019

Prague, Czech Republic

Outline

Horizontal SCA attacks on ECC

Random-Order Elliptic Curves

Structured-Order Elliptic Curves

Concludion and Future Directions

Outline

Outline

Elliptic Curves and Scalar Multiplication

- Let $\mathcal{E}$ an Elliptic Curve over $\mathbb{F}_p$.
- $Q = d[P]$
  with P, Q two points on $\mathcal{E}$ and d a scalar.
- E is the order of $\mathcal{E}$
  for all $P \in \mathbb{E}, E[P] = \mathcal{O}$.
- Scalar Multiplication is easy to compute (Double and Add algorithm).
- Scalar Multiplication Inverse is hard (ECDLP) $\Rightarrow$ Security of ECC.

Elliptic Curves and Scalar Multiplication

- Let $\mathcal{E}$ an Elliptic Curve over $\mathbb{F}_p$.
- $Q = d[P]$
  with P, Q two points on $\mathcal{E}$ and d a secret scalar.
- E is the order of $\mathcal{E}$
  for all $P \in \mathbb{E}, E[P] = \mathcal{O}$.
- Scalar Multiplication is easy to compute (Double and Add algorithm).
- Scalar Multiplication Inverse is hard (ECDLP) $\Rightarrow$ Security of ECC.

Scalar Multiplication Implementation

Data: d scalar, P point of curve $\mathbb{E}$ defined over $\mathbb{F}_p$
Result: $Q = d\,[P]$
Initialization: $Q = P$

```
for i : size(d) − 2 downto 0 do
    Q = 2 Q
    if d[i] == 1 then
        Q = P + Q
    else
        Do Nothing
    end
end
```

Scalar Multiplication Implementation

Data: d scalar, P point of curve $\mathbb{E}$ defined over $\mathbb{F}_p$
Result: $Q = d\,[P]$
Initialization: $Q = P$

SCA

```
for i : size(d) − 2 downto 0 do
    Q = 2 Q
    if d[i] == 1 then
        Q = P + Q
    else
        Do Nothing
    end
end
```

Scalar Multiplication Implementation

Data: d scalar, P point of curve $\mathbb{E}$ defined over $\mathbb{F}_p$
Result: $Q = d\,[P]$
Initialization: $Q = P$

for i : $size(d) - 2$ *downto* 0 **do**
    $Q = 2\,Q$
    if $d[i] == 1$ **then**
        $Q = P + Q$
    else
        Do Nothing
    end
end

SCA

- time
- power
- electromagnetic

Scalar Multiplication Implementation

Data: d scalar, P point of curve $\mathbb{E}$ defined over $\mathbb{F}_p$
Result: $Q = d\,[P]$
Initialization: $Q = P$

for i : $size(d) - 2$ *downto* 0 **do**
  $Q = 2\,Q$
  if $d[i] == 1$ **then**
    $Q = P + Q$
  else
    Do Nothing
  end
end

SCA

- time
- power
- electromagnetic
- cache behaviour
- speculative execution
- ...

Scalar Multiplication Implementation

Data: d scalar, P point of curve $\mathbb{E}$ defined over $\mathbb{F}_p$
Result: $Q = d\,[P]$
Initialization: $Q = P$

for i : $size(d) - 2$ *downto* 0 **do**
  $Q = 2\,Q$
  if $d[i] == 1$ **then**
    $Q = P + Q$
  else
    Do Nothing
  end
end

SCA

- time
- power
- electromagnetic
- cache behaviour
- speculative execution
- ...

Scalar Multiplication Implementation

Data: d scalar, P point of curve $\mathbb{E}$ defined over $\mathbb{F}_p$
Result: $Q = d\,[P]$
Initialization: $Q = P$

```
for i : size(d) − 2 downto 0 do
    Q = 2 Q
    if d[i] == 1 then
        Q = P + Q
    else
        Dummy = P + Q
    end
end
```

SCA

- time
- power
- electromagnetic
- cache behaviour
- speculative execution
- ...
- constant Time

Scalar Multiplication Implementation

Data: d scalar, P point of curve $\mathbb{E}$ defined over $\mathbb{F}_p$
Result: $Q = d\,[P]$
Initialization: $Q = P$

for $i : size(d) - 2$ *downto* 0 **do**
  $Q = 2\,Q$
  if $d[i] == 1$ **then**
    $Q = P + Q$
  else
    $Dummy = P + Q$
  end
end

SCA

- time
- power
- electromagnetic
- cache behaviour
- speculative execution
- ...
- constant Time
- point randomization
- curve randomization
- address randomization
- ...

Scalar Multiplication Implementation

Data: d scalar, P point of curve $\mathbb{E}$ defined over $\mathbb{F}_p$
Result: $Q = d\,[P]$
Initialization: $Q = P$

for i : $size(d) - 2$ *downto* 0 **do**
  $Q = 2\,Q$
  if $d[i] == 1$ **then**
    $Q = P + Q$
  else
    $Dummy = P + Q$
  end
end

SCA

- time
- power
- electromagnetic
- cache behaviour
- speculative execution
- ...
- constant Time
- point randomization
- curve randomization
- address randomization
- ...

Scalar Multiplication Implementation

Data: d scalar, P point of curve $\mathbb{E}$ defined over $\mathbb{F}_p$
Result: $Q = d[P]$
Initialization: $Q = P$

for i : $size(d) - 2$ *downto* 0 **do**
  $Q = 2Q$
  if $d[i] == 1$ **then**
    $Q = P + Q$
  else
    $Dummy = P + Q$
  end
end

$$d[P] = d[P]$$
$$d = d + r \times E$$

SCA

- time
- power
- electromagnetic
- cache behaviour
- speculative execution
- ...

- constant Time
- point randomization
- curve randomization
- address randomization
- scalar randomization
- ...

Scalar Multiplication Implementation

Data: d scalar, P point of curve $\mathbb{E}$ defined over $\mathbb{F}_p$
Result: $Q = d\,[P]$
Initialization: $Q = P$

for i : $size(d) - 2$ *downto* 0 **do**
    $Q = 2\,Q$
    if $d[i] == 1$ **then**
        $Q = P + Q$
    else
        $Dummy = P + Q$
    end
end

$$d_\ell[P] = d[P]$$
$$d_\ell = d + r_\ell \times E$$

SCA

- time
- power
- electromagnetic
- cache behaviour
- speculative execution
- ...

- constant Time
- point randomization
- curve randomization
- address randomization
- scalar randomization
- ...

Scalar Multiplication Implementation

Data: d scalar, P point of curve $\mathbb{E}$ defined over $\mathbb{F}_p$

Result: $Q = d\,[P]$

Initialization: $Q = P$

for i : $size(d) - 2$ *downto* 0 **do**

  $Q = 2\,Q$

  if $d[i] == 1$ **then**

    $Q = P + Q$

  else

    $Dummy = P + Q$

  end

end

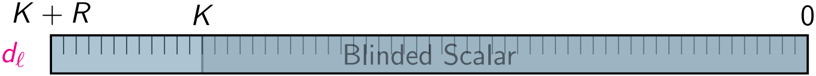
SCA

- time
- power
- electromagnetic
- cache behaviour
- speculative execution
- ...

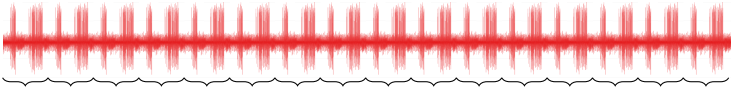
- constant Time
- point randomization
- curve randomization
- address randomization
- scalar randomization
- ...

$$d_\ell[P] = d[P]$$

$$d_\ell = d + r_\ell \times E$$



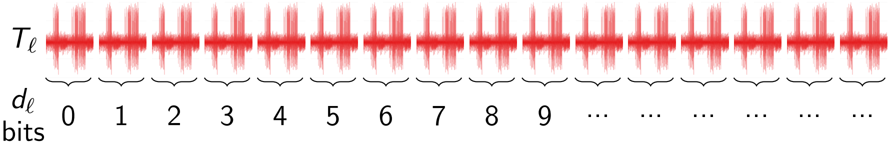
Horizontal SCA



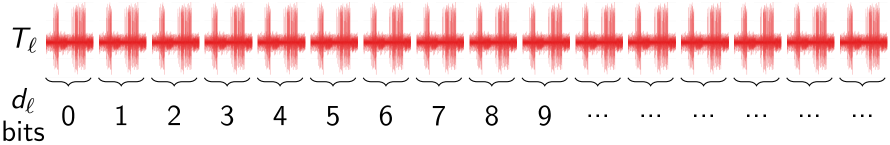
T_ℓ

d_ℓ bits: 0 1 2 3 4 5 6 7 8 9 … … … … … …

Horizontal SCA



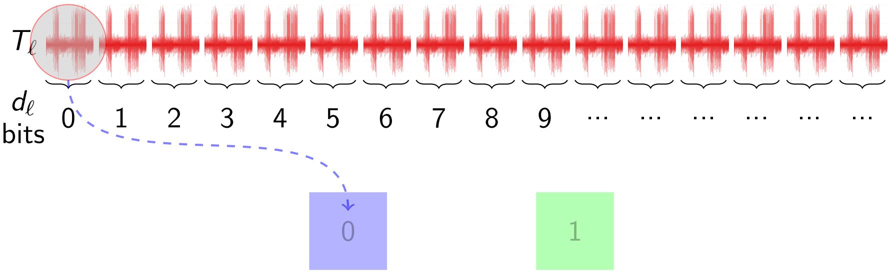
Horizontal SCA



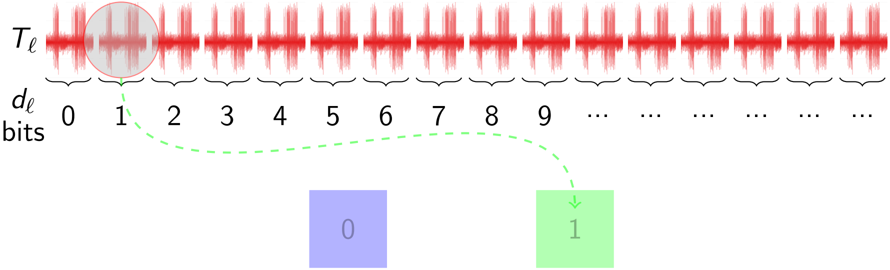
0

1

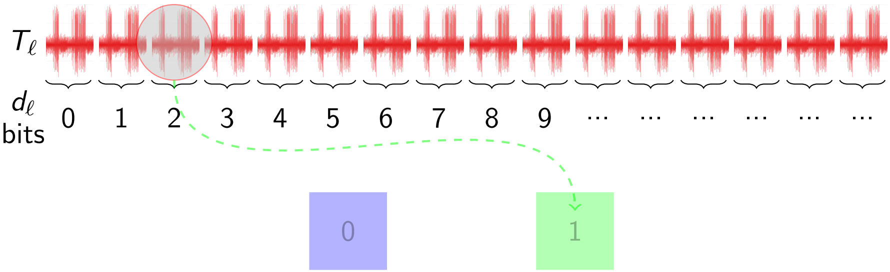
Horizontal SCA



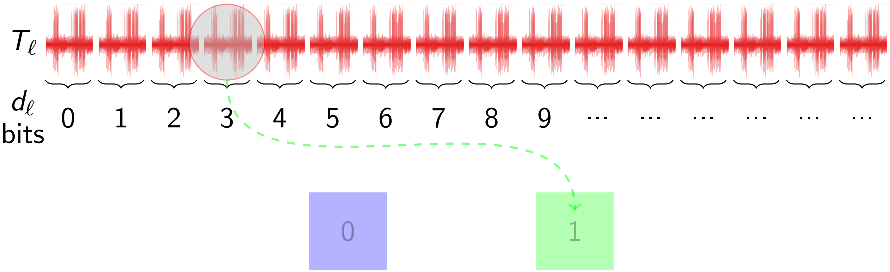
Horizontal SCA



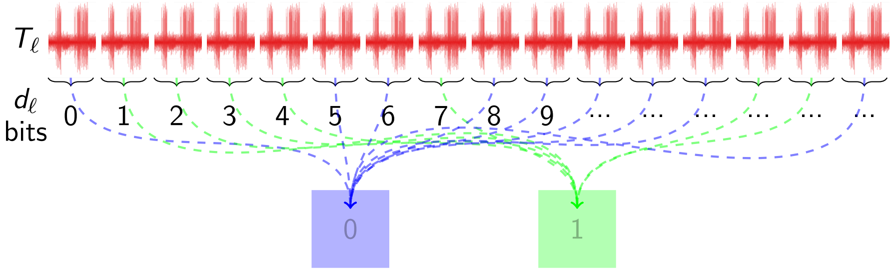
Horizontal SCA



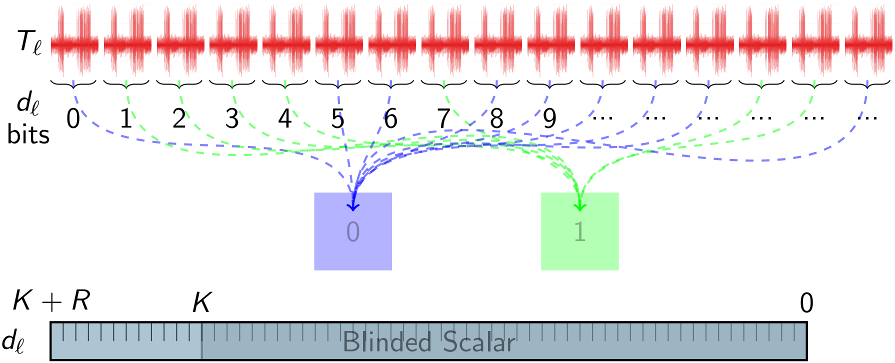
Horizontal SCA



Horizontal SCA



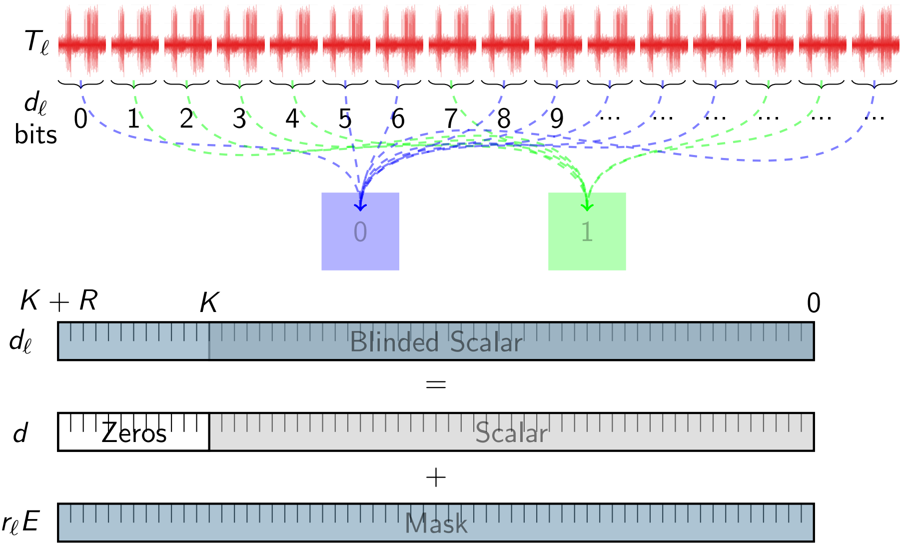
Horizontal SCA



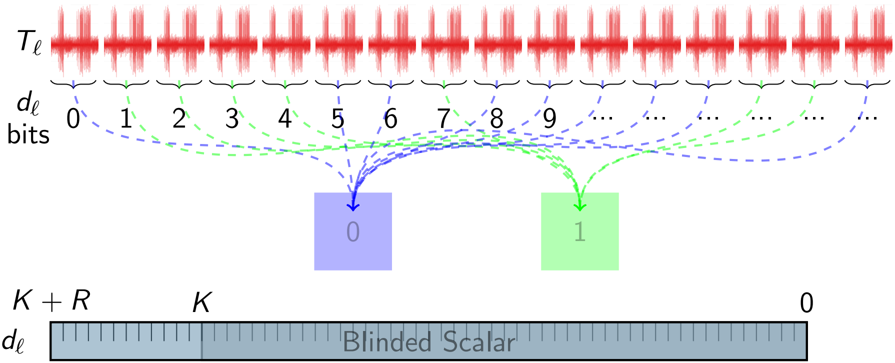
[Bauer *et al.* 2013]
[Weissbart *et al.* 2019]
[Carbone *et al.* 2019]

[Heyszl *et al.* 2013]
[Perin *et al.* 2014]
[Specht *et al.* 2015]
[Nascimento *et al.* 2016]
[Järvinen *et al.* 2016]
[Nascimento *et al.* 2017]

Horizontal SCA

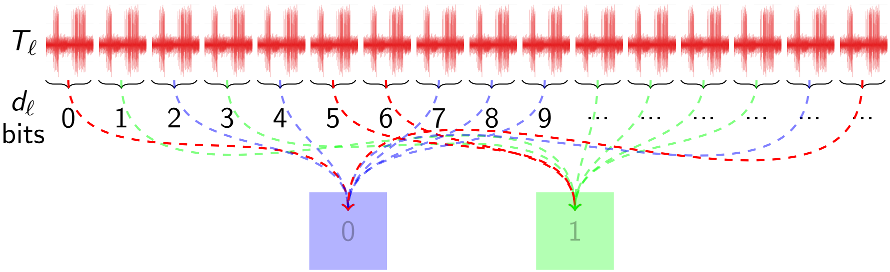


Horizontal SCA

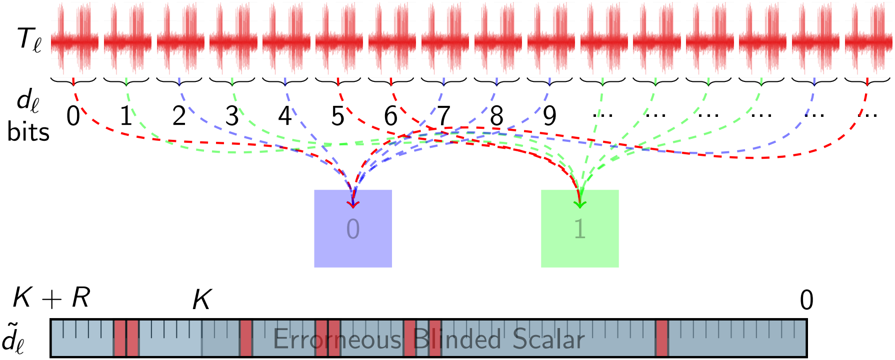


$$d = d_\ell \bmod E$$

Horizontal SCA

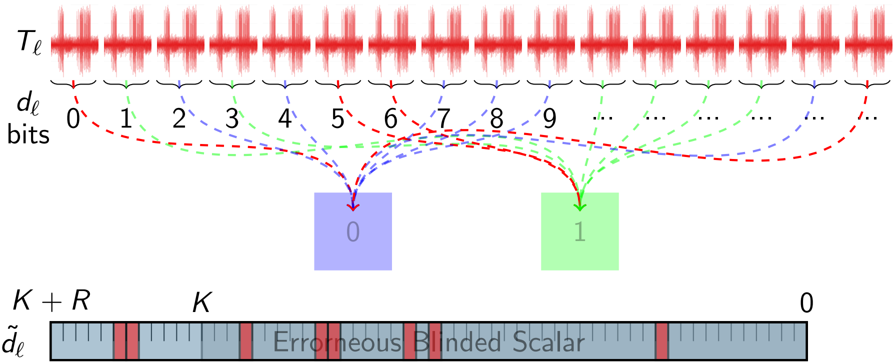


Horizontal SCA



ϵ_b: bit-error probability

Horizontal SCA

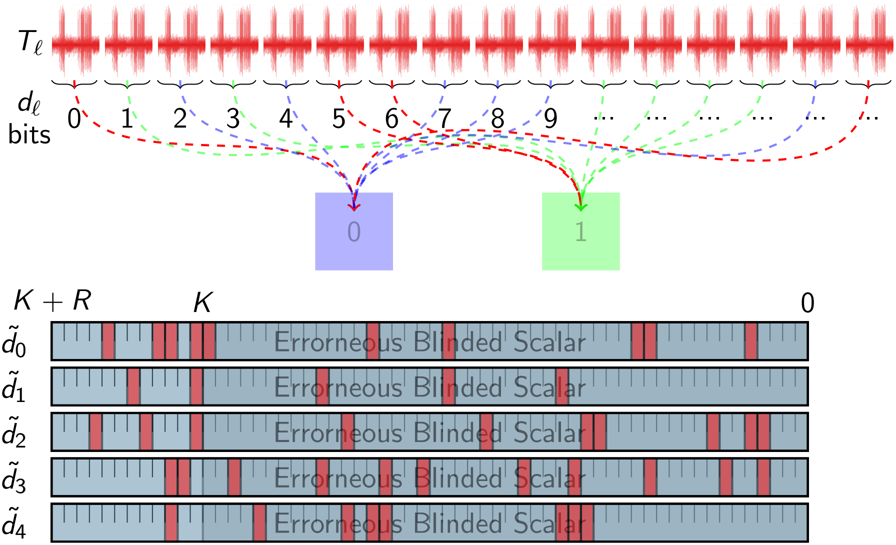


ϵ_b: bit-error probability

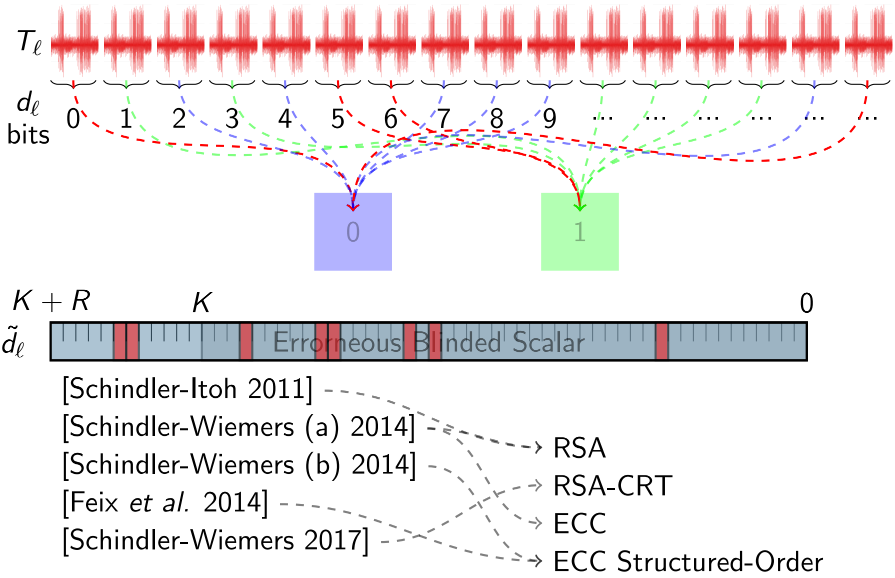
[Bauer *et al.* 2013]
[Weissbart *et al.* 2019]
[Carbone *et al.* 2019]

[Heyszl *et al.* 2013]
[Perin *et al.* 2014]
[Specht *et al.* 2015]
[Nascimento *et al.* 2016]
[Järvinen *et al.* 2016]
[Nascimento *et al.* 2017]

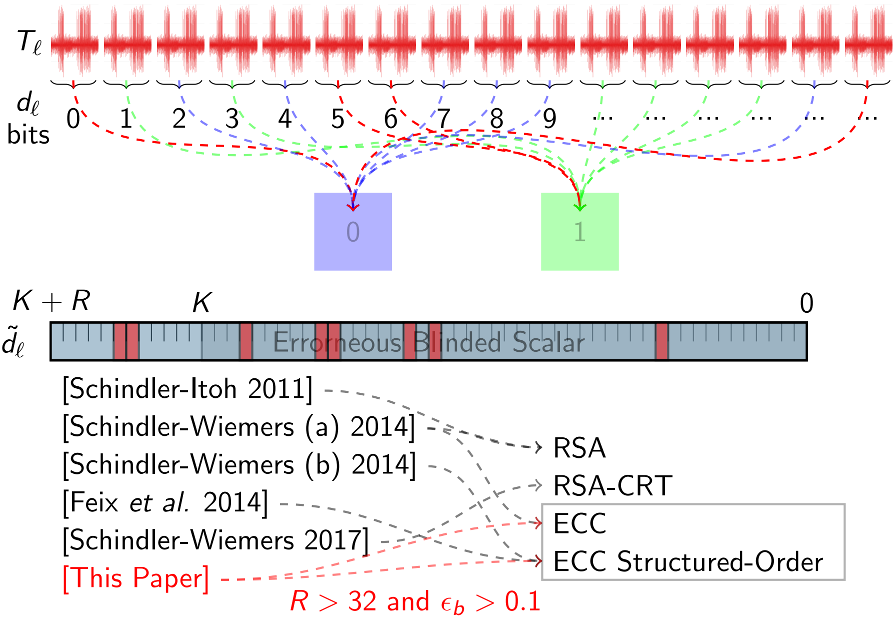
Horizontal SCA



Horizontal SCA



Horizontal SCA



Outline

Outline

Horizontal SCA attacks on ECC

Random-Order Elliptic Curves

Structured-Order Elliptic Curves

Concludion and Future Directions

Outline

Horizontal SCA attacks on ECC

~~Random-Order Elliptic Curves~~

Structured-Order Elliptic Curves

Concludion and Future Directions

NAF Distinguisher issue
$\rightarrow$ invalidates the contribution

currently working on a patch
eprint to be updated shortly

Outline

Structured-Order Elliptic Curves

- SEC2 curves (CERTICOM 2000)
- NIST curves (NIST FIPS 186-4)
- Curve25519 (Bernstein 2005)
- Brainpool curves (BSI RFC 5639)

$\Rightarrow$ **order E structured:** $E = 2^K \pm E_0$, where E_0 close to $2^{K/2}$

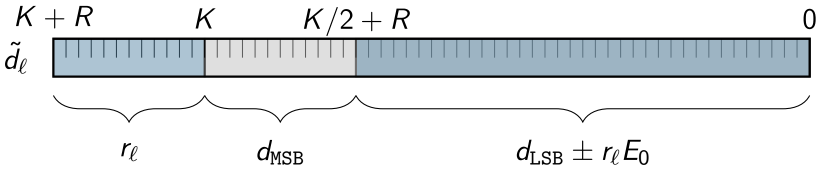
example - order of curve SEC2 secp256k1 (Bitcoin curve):

E = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141

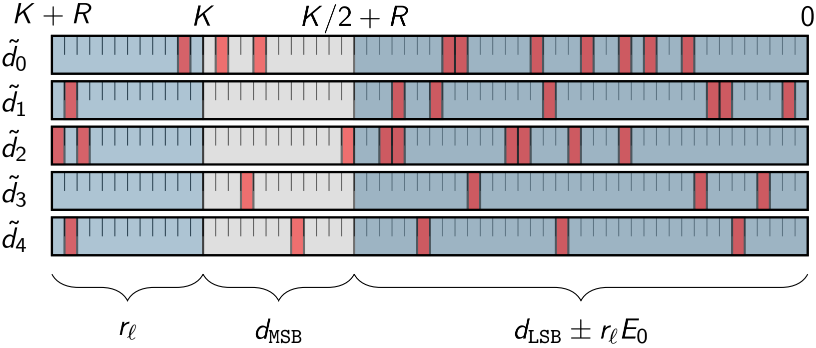
Structured-Order Effect on Scalar Randomization

- (d, E) of length K, r_ℓ of length R (with $R < K/2$)
- $d_\ell = d + r_\ell \times E$
- $E = 2^K \pm E_0$

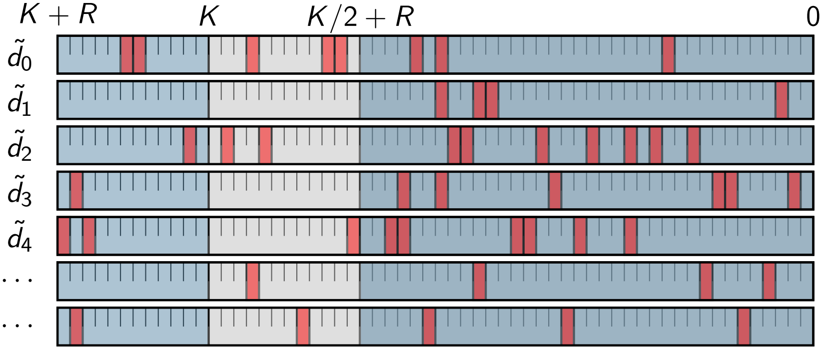
$$d_\ell = r_\ell \times 2^K + d \pm r_\ell \times E_0$$



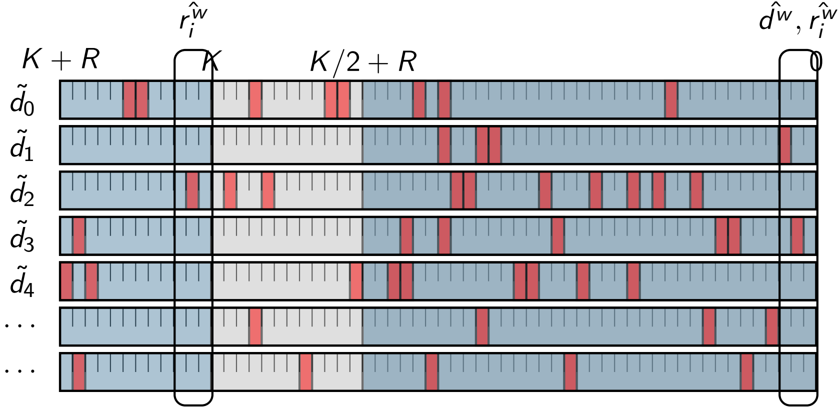
Problem Definition



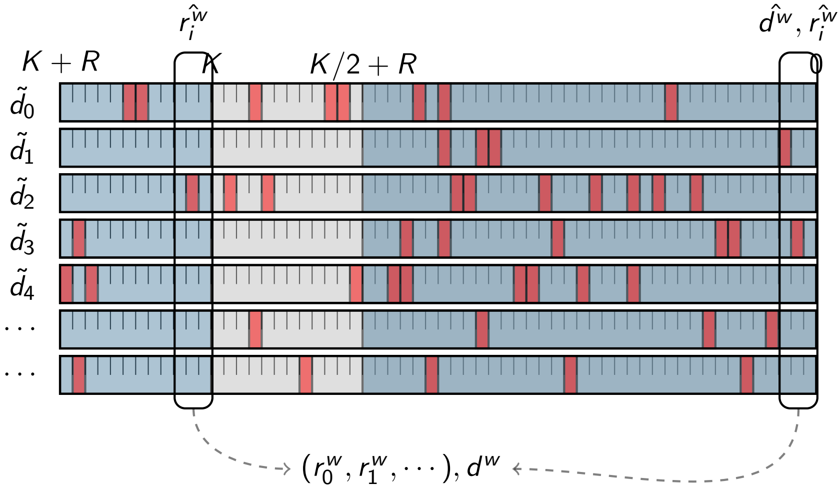
A divide and conquer algorithm [Schindler-Wiemers 2014]



A divide and conquer algorithm [Schindler-Wiemers 2014]

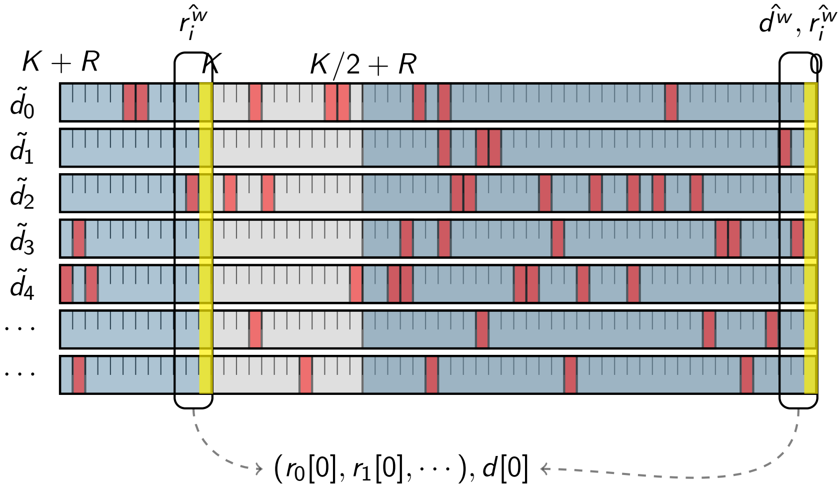


A divide and conquer algorithm [Schindler-Wiemers 2014]



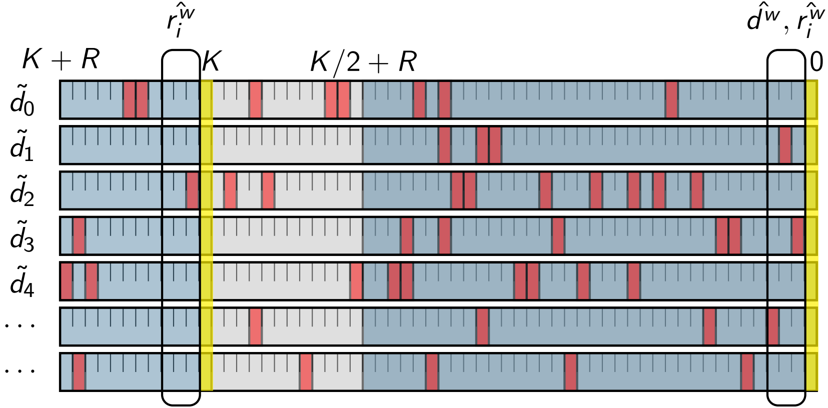
Complexity: $O(N2^{2w})$

A divide and conquer algorithm [Schindler-Wiemers 2014]



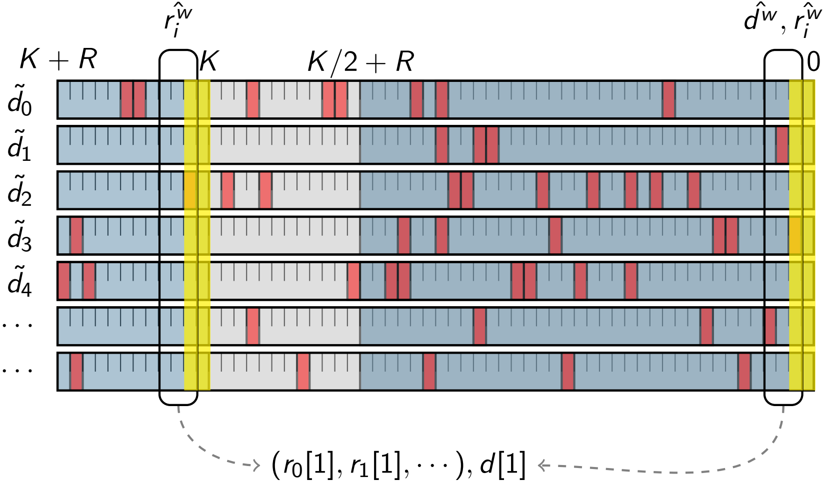
Complexity: $O(N2^{2w})$

A divide and conquer algorithm [Schindler-Wiemers 2014]



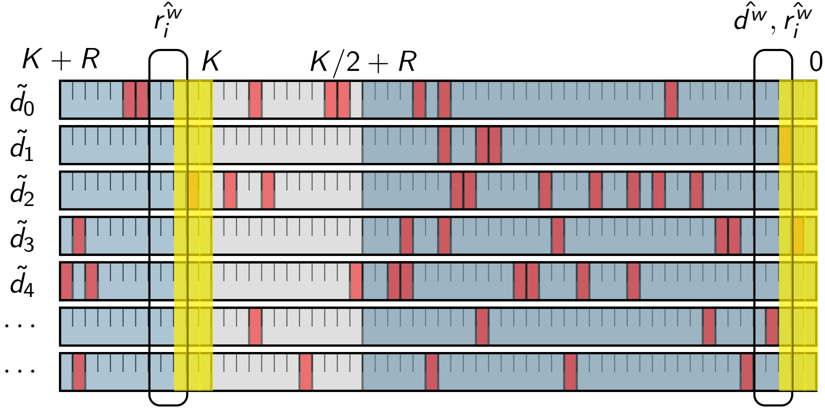
Complexity: $O(N2^{2w})$

A divide and conquer algorithm [Schindler-Wiemers 2014]



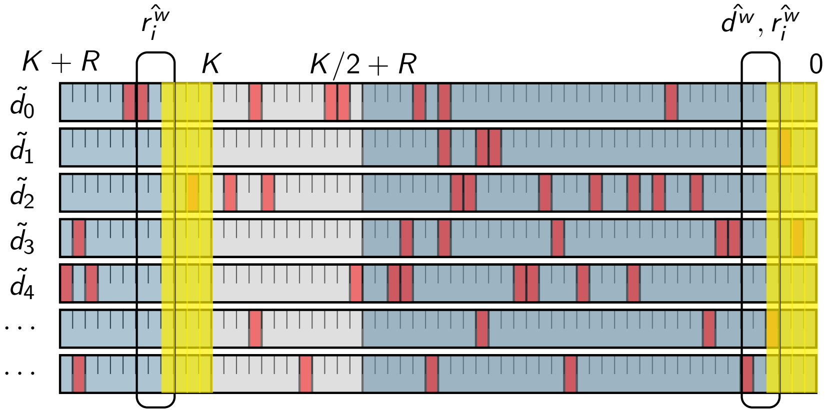
Complexity: $O(N2^{2w})$

A divide and conquer algorithm [Schindler-Wiemers 2014]



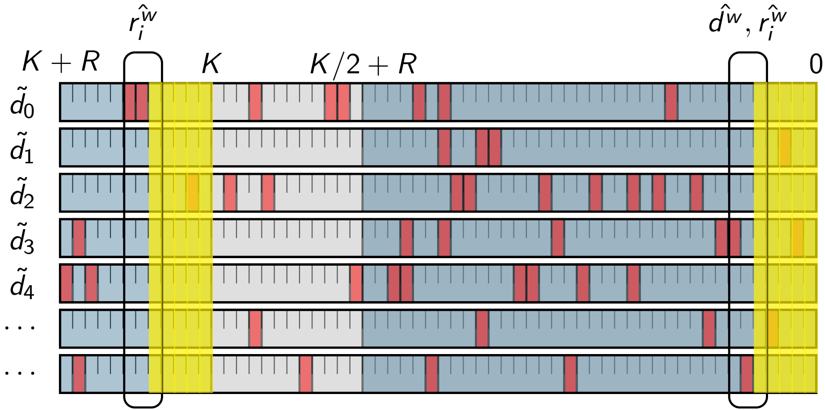
Complexity: $O(N2^{2w})$

A divide and conquer algorithm [Schindler-Wiemers 2014]



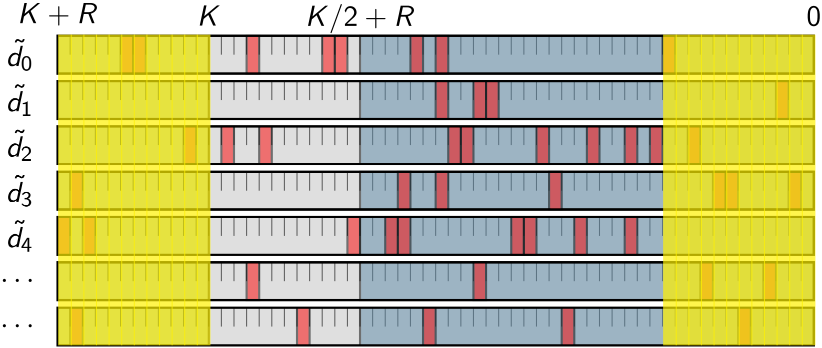
Complexity: $O(N2^{2w})$

A divide and conquer algorithm [Schindler-Wiemers 2014]



Complexity: $O(N2^{2w})$

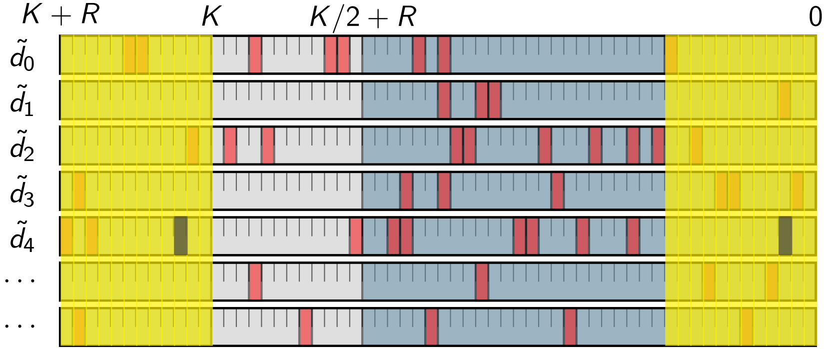
A divide and conquer algorithm [Schindler-Wiemers 2014]



$(r_0, r_1, \cdots), d^R$

Complexity: $O(RN2^{2w})$

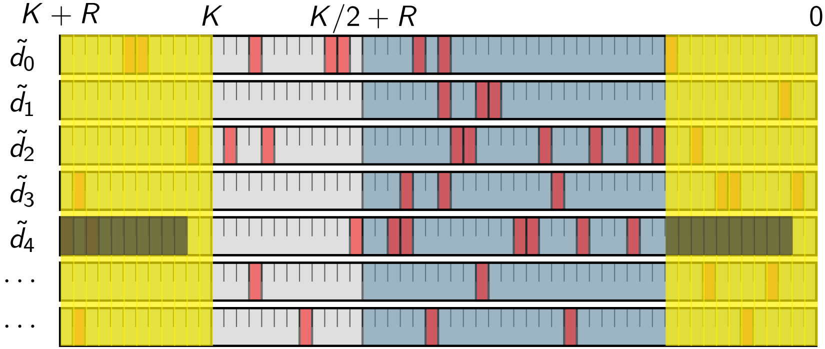
A divide and conquer algorithm [Schindler-Wiemers 2014]



$(r_0, r_1, \cdots), d^R$

Complexity: $O(RN2^{2w})$

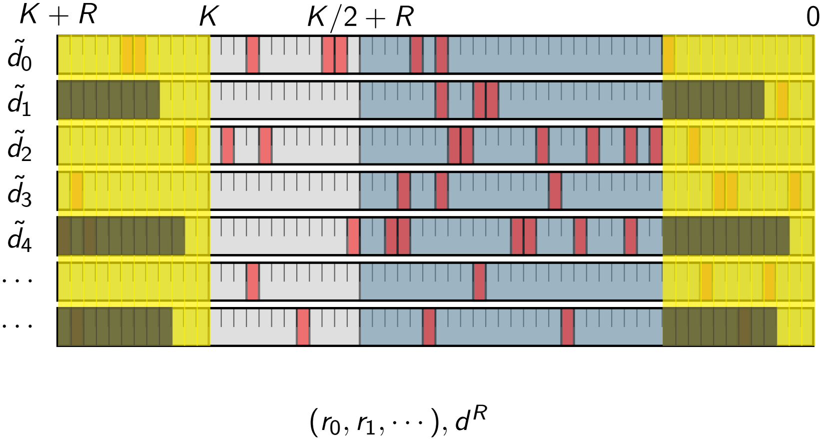
A divide and conquer algorithm [Schindler-Wiemers 2014]



$(r_0, r_1, \cdots), d^R$

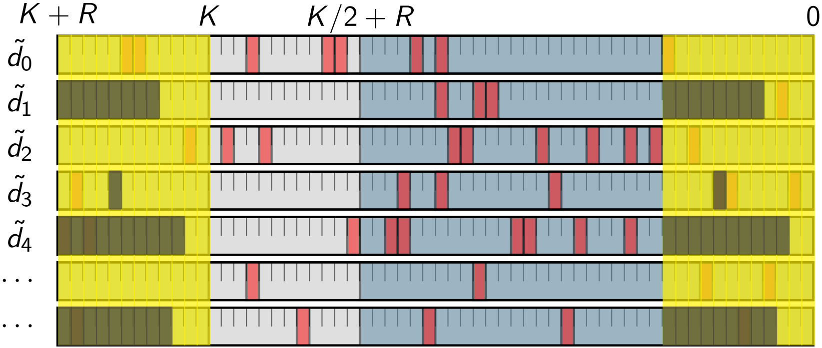
Complexity: $O(RN2^{2w})$

A divide and conquer algorithm [Schindler-Wiemers 2014]



Complexity: $O(RN2^{2w})$

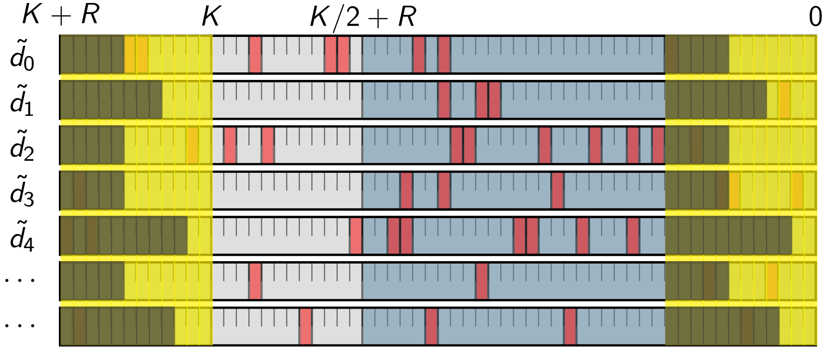
A divide and conquer algorithm [Schindler-Wiemers 2014]



$(r_0, r_1, \cdots), d^R$

Complexity: $O(RN2^{2w})$

A divide and conquer algorithm [Schindler-Wiemers 2014]



$(r_0, r_1, \cdots), d^R$

Complexity: $O(RN2^{2w})$

Simulation Results [Schindler-Wiemers 2014]

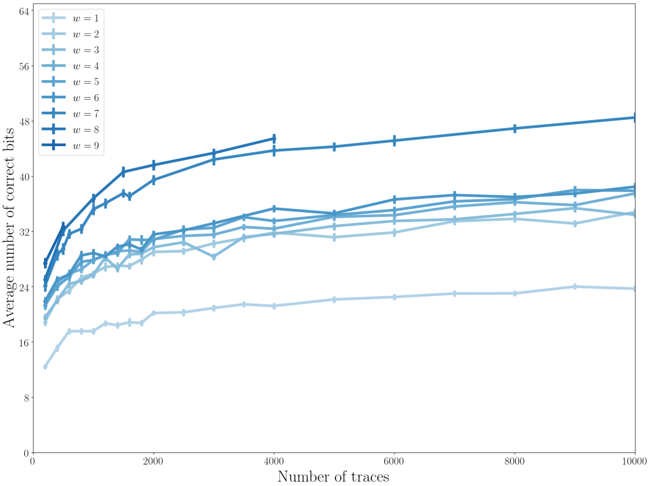
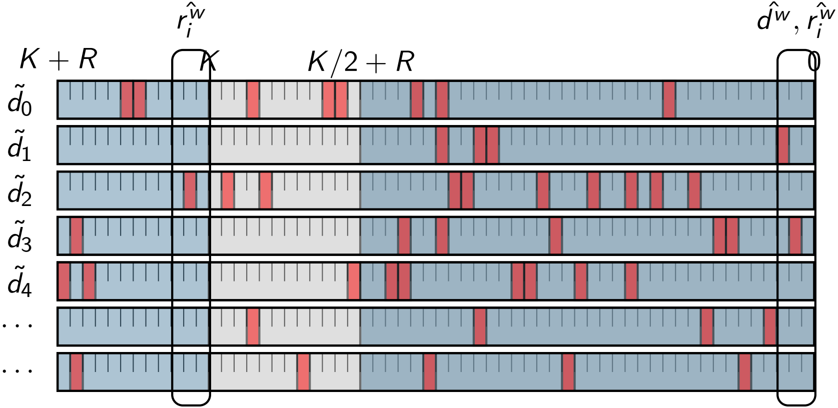
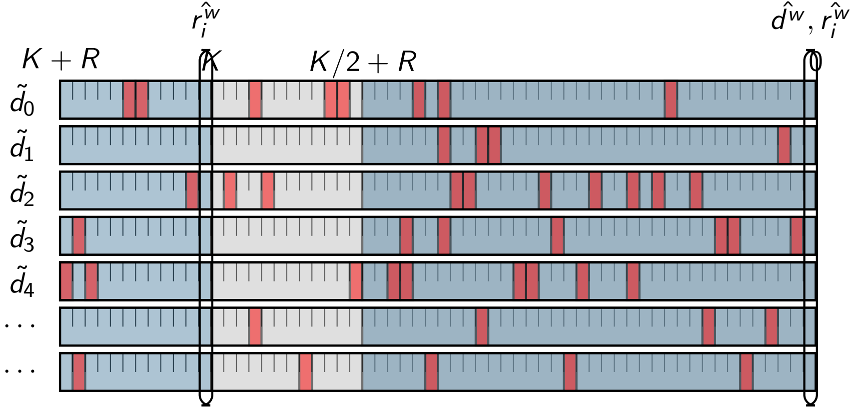


Figure: $K = 256, R = 64, \epsilon_b = 0.15$ on curve `secp-256-k1`.

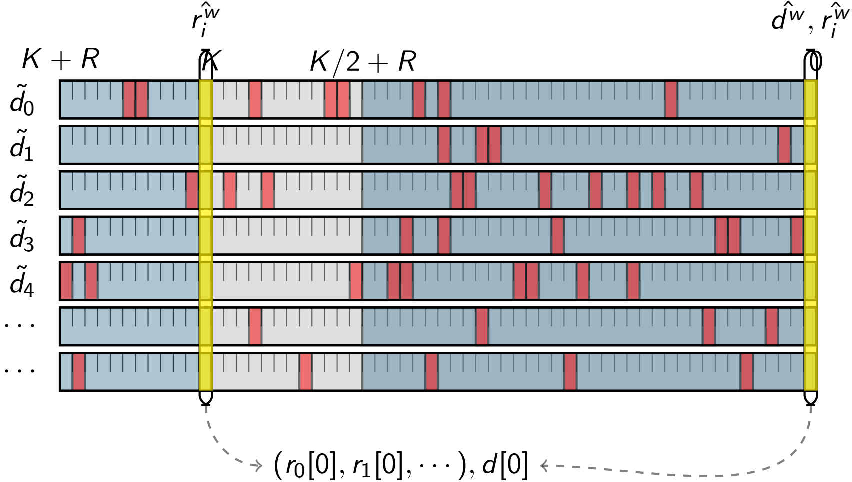
Proposed Improvements on Schindler-Wiemers' Algorithm



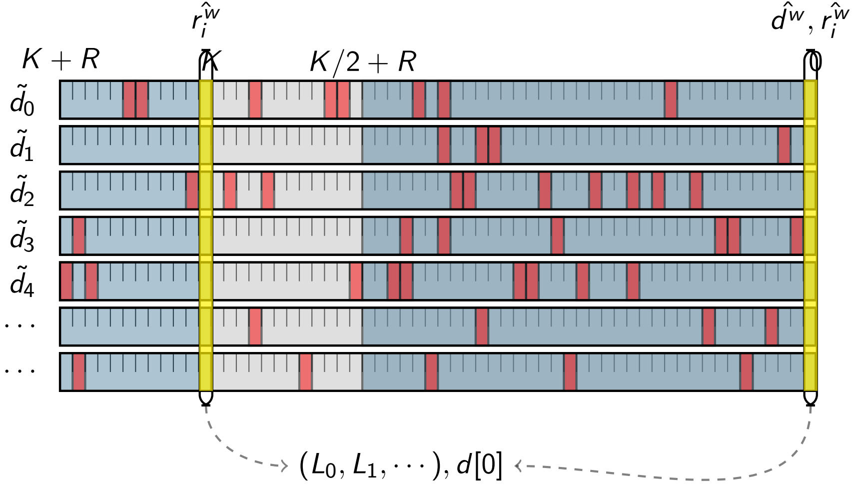
Proposed Improvements on Schindler-Wiemers' Algorithm



Proposed Improvements on Schindler-Wiemers' Algorithm

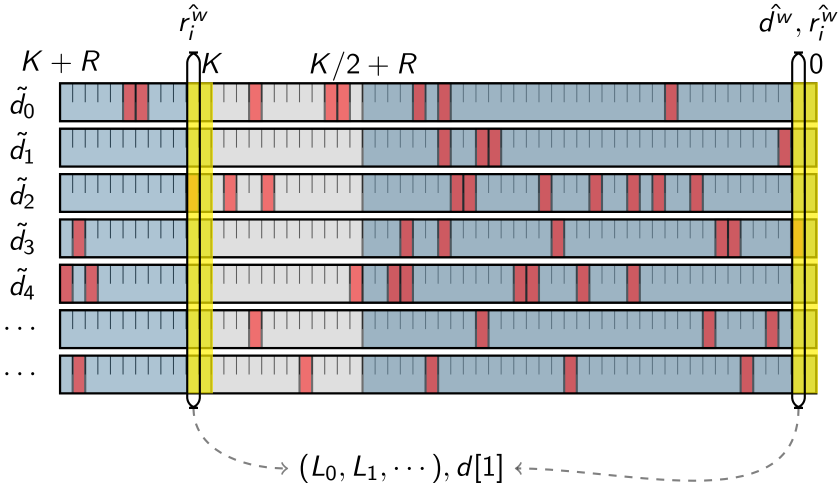


Proposed Improvements on Schindler-Wiemers' Algorithm



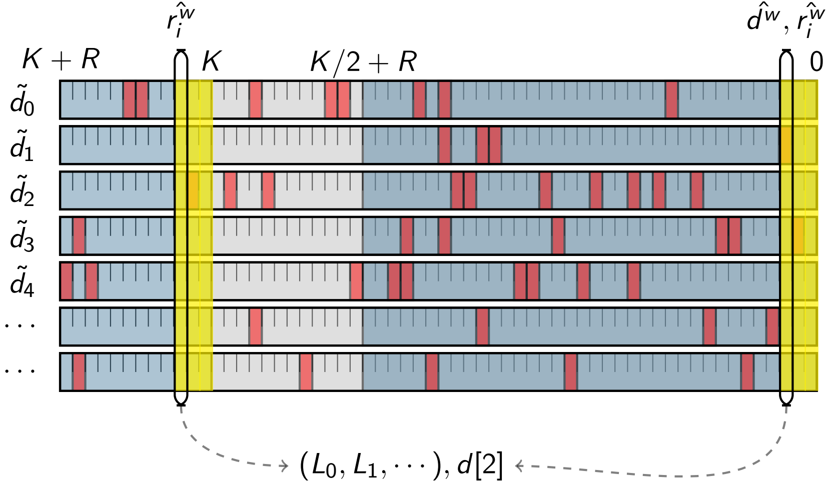
L_ℓ stores the most probable values for r_ℓ, $\#L_\ell \leq L$

Proposed Improvements on Schindler-Wiemers' Algorithm



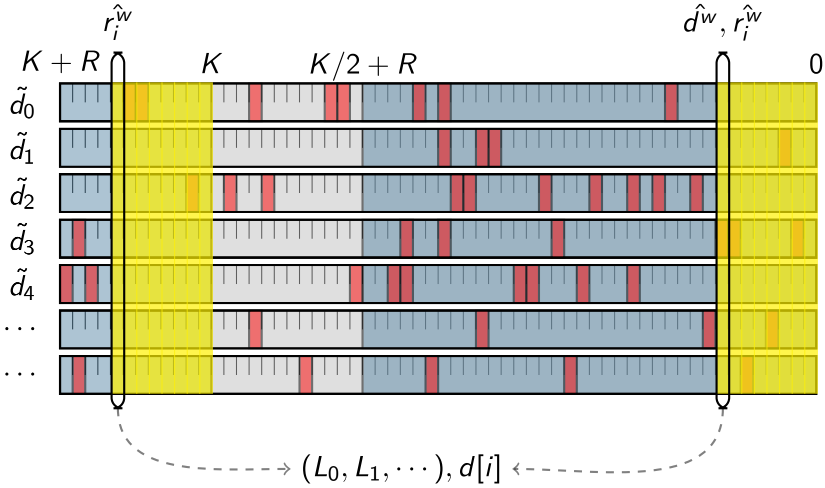
L_ℓ stores the most probable values for r_ℓ, $\#L_\ell \leq L$

Proposed Improvements on Schindler-Wiemers' Algorithm



L_ℓ stores the most probable values for r_ℓ, $\#L_\ell \leq L$

Proposed Improvements on Schindler-Wiemers' Algorithm



Complexity: $O(RNL)$ - L_ℓ stores the most probable values for r_ℓ, $\#L_\ell \leq L$

Simulation Results

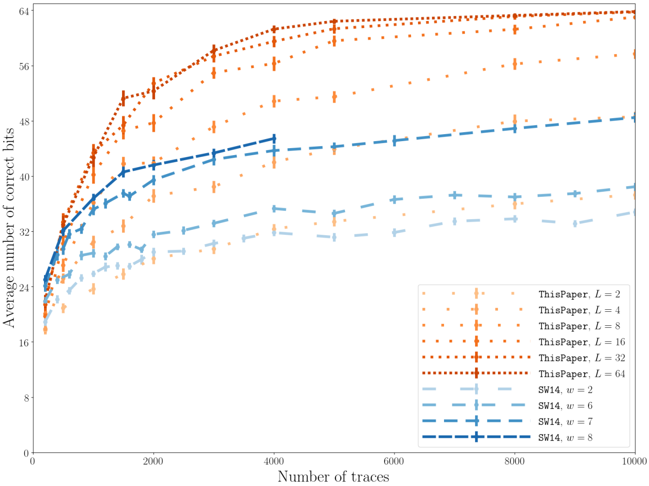


Figure: $K = 256, R = 64, \epsilon_b = 0.15$ on curve secp-256-k1.

Simulation Results

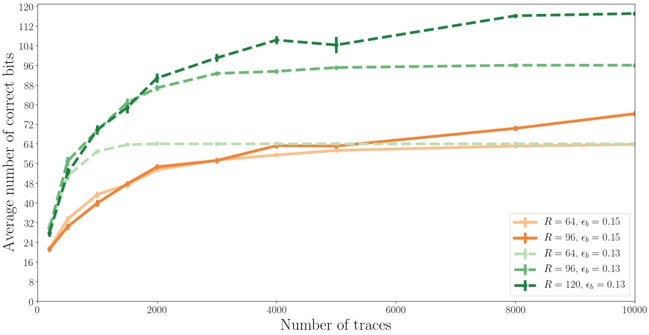
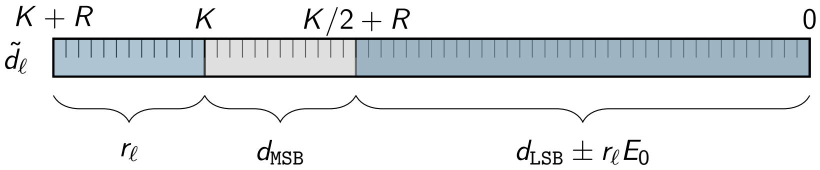


Figure: $K = 256, L = 32, t = 16.$

Structured-Order Effect on Scalar Randomization

- (d, E) of length K, r_ℓ of length R (with $R < K/2$)
- $d_\ell = d + r_\ell \times E$
- $E = 2^K \pm E_0$

$$d_\ell = r_\ell \times 2^K + d \pm r_\ell \times E_0$$



Outline

Conclusion And Future Directions

What we have seen

- The problem of blinded scalar correction is a critical problem in real-wolrd side-channel attacks.
- For structued-order Elliptic Curves taking $R < K/2$ is a clearly a bad idea.

Next Steps

- Find theoretic bounds $B(N)$ on the bit-error probability s.t. if $\epsilon_b < B(N)$ then correction is possible with N observations.
- Correct algorithm for random-order Elliptic Curves...