# Online Template Attacks: Revisited

Tampere University

Alejandro Cabrera Aldaya and Billy Bob Brumley

Tampere University, Finland

0

works

Previous

### **Generic Scalar Multiplication**

Input: Integer k and generator G Output: P = kG K = Encode(k)  $S'_0 = \text{Init}(G)$ for  $\kappa_i$  in  $\kappa = {\kappa_1, \kappa_2, ..., \kappa_n}$  do  $S_j = \text{Select}(S'_{i-1}, \kappa_i)$   $S'_i = \text{Process}(S_i)$   $P = \text{Finalize}(S'_n)$ return P  $S'_i = \text{Process}(S_i)$   $P = \text{Finalize}(S'_n)$   $S'_i = \text{Process}(S_i)$   $S'_i = \text{Process}(S_i)$  $S'_i = \text{Process}(S_$ 

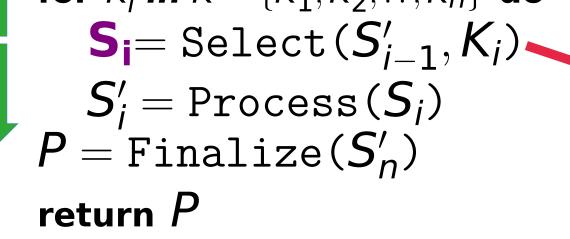
## **Attack Input and Direction**

K = Encode(k)  $S'_{0} = \text{Init}(G)$ for  $K_{i}$  in  $K = \{K_{1}, K_{2}, ..., K_{n}\}$  do  $S_{i} = \text{Select}(S'_{i-1}, K_{i})$   $S'_{i} = \text{Process}(S_{i})$   $P = \text{Finalize}(S'_{n})$ return P K = Encode(k)  $S'_{0} = \text{Init}(G)$ for  $K_{i}$  in  $K = \{K_{1}, K_{2}, ..., K_{n}\}$  do  $S_{i} = \text{Select}(S'_{i-1}, K_{i})$   $S'_{i} = \text{Process}(S_{i})$   $P = \text{Finalize}(S'_{n})$ return P

K = Encode(k) $S'_0 = \text{Init}(G)$ for  $K_i$  in  $K = \{K_1, K_2, \dots, K_n\}$  do Projective coordinates attack" [2]: ≤ 5 bits
Backward OTA (this): full scalar recovery.

#### **Generic trace**

An OTA assumes each execution of the Process operation leaks about the state  $S_i$ : Process $(S_i) \rightsquigarrow L_i(S_i)$ . A generic trace is composed by a sequence of  $L_i$ :  $\{L_1, L_2, ..., L_n\}$ .



**Open Question:** how to get an intermediate state?

### **Controlled Side-Channels**

Page	Instructions	Side-channels and their traces
<b>P1</b>	nop	PageTracer: Tracks the sequence of executed memory pages [4].
	nop	
	nop	
	• • •	
<b>P2</b>	nop	<b>CopyCat:</b> Counts # executed instructions at each tracked page [5].
	• • •	
<b>P3</b>	nop	3
	nop	

#### **Analyzed Open Source Libraries**

Comb method	Montgomery ladder
$elect(K_1, P)$ = K : i = [2, n] do = 2R	R = G, S = 2G for $i = \lfloor \log_2(k) \rfloor - 1$ downto 0 do if $k_i = 0$ then S = R + S, R = 2R else R = R + S, S = 2S return $R$
	Encode(k) recompute(G) $Elect(K_1, P)$ E K : i = [2, n] do $E Select(K_i, P)$ E R + T

return R

#### Leakage Analysis

For each library we enumerated the memory pages used by the selected Process operations highlighted above. We evaluated the difficulty of an OTA for each memory page combination using both PageTracer and CopyCat side-channels. The percentages below correspond to the ratio of combinations that fall into **Ideal** and **Insecure** settings.

**Attack Ideal Insecure Max bias** 

**Attack Ideal Insecure Max bias** 

**Attack Ideal Insecure Max bias** 

PageTracer	0	87%	50%	PageTracer	84%	99%	62%	PageTrace	er O	69%	52%
CopyCat 5	50%	98%	30%	CopyCat	99%	100%	24%	CopyCat	47%	94%	24%

#### References



Batina *et al.* Online Template Attacks. *INDOCRYPT*, 2014.
 Naccache *et al.* Projective Coordinates Leak. *EUROCRYPT*, 2004.
 Aldaya *et al.* From A to Z: Projective Coordinates Leakage in the Wild. *CHES*, 2020.
 Xu *et al.* Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems. *IEEE Symposium on Security and Privacy*, 2015.

- [5] Moghimi et al. CopyCat: Controlled Instruction-Level Attacks on Enclaves. USENIX Security Symposium, 2020.
- [6] Dugardin et al. Dismantling Real-World ECC with Horizontal and Vertical Template Attacks. COSADE, 2016.



IACR Conference on Cryptographic Hardware and Embedded Systems, CHES 2022