Online Template Attacks: Revisited

Alejandro Cabrera Aldaya and Billy Bob Brumley

Tampere University, Finland

Generic Scalar Multiplication

Input: Integer k and generator G
Output: $P = kG$
$K = \texttt{Encode}(k)$
$S'_0 = \texttt{Init}(G)$
for K_i **in** $K = \{K_1, K_2, \ldots, K_n\}$ **do**
  $S_i = \texttt{Select}(S'_{i-1}, K_i)$
  $S'_i = \texttt{Process}(S_i)$
$P = \texttt{Finalize}(S'_n)$
return P

- Secure Branches
- Table lookups
- Leakage free

- Point operations
- OTA target

Generic trace

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

Attack Input and Direction

Previous works [1,6]...

$K = \texttt{Encode}(k)$
$S'_0 = \texttt{Init}(G)$
for K_i **in** $K = \{K_1, K_2, \ldots, K_n\}$ **do**
  $S_i = \texttt{Select}(S'_{i-1}, K_i)$
  $S'_i = \texttt{Process}(S_i)$
$P = \texttt{Finalize}(S'_n)$
return P

$K = \texttt{Encode}(k)$
$S'_0 = \texttt{Init}(G)$
for K_i **in** $K = \{K_1, K_2, \ldots, K_n\}$ **do**
  $S_i = \texttt{Select}(S'_{i-1}, K_i)$
  $S'_i = \texttt{Process}(S_i)$
$P = \texttt{Finalize}(S'_n)$
return P

- "Projective coordinates attack" **[2]**: ≤ 5 bits
- **Backward OTA** (this): full scalar recovery.

$K = \texttt{Encode}(k)$
$S'_0 = \texttt{Init}(G)$
for K_i **in** $K = \{K_1, K_2, \ldots, K_n\}$ **do**
  $S_i = \texttt{Select}(S'_{i-1}, K_i)$
  $S'_i = \texttt{Process}(S_i)$
$P = \texttt{Finalize}(S'_n)$
return P

Open Question:
how to get an intermediate state?

Controlled Side-Channels

Page	Instructions
	nop
P1	nop
	nop
	...
P2	nop
	...
P3	nop
	nop

Side-channels and their traces

PageTracer: Tracks the sequence of executed memory pages **[4]**.

CopyCat: Counts # executed instructions at each tracked page **[5]**.

3 1 2

Analyzed Open Source Libraries

libgcrypt

Double-and-add always

$R = \mathcal{O}$
for $i = \lfloor \log_2 k \rfloor$ ***downto*** 0 **do**
  $R = 2R$
  $T = R + G$
  $R = \texttt{cond\_assign}(T, R, k_i)$
return R

mbedTLS

Comb method

$K = Encode(k)$
$P = Precompute(G)$
$R = Select(K_1, P)$
for $K_i \in K : i = [2, n]$ **do**
  $R = 2R$
  $T = Select(K_i, P)$
  $R = R + T$
return R

wolfSSL

Montgomery ladder

$R = G, S = 2G$
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

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.

libgcrypt:

Attack	Ideal	Insecure	Max bias
PageTracer	0	87%	50%
CopyCat	50%	98%	30%

mbedTLS:

Attack	Ideal	Insecure	Max bias
PageTracer	84%	99%	62%
CopyCat	99%	100%	24%

wolfSSL:

Attack	Ideal	Insecure	Max bias
PageTracer	0	69%	52%
CopyCat	47%	94%	24%

References

Paper

[1] Batina *et al.* **Online Template Attacks**. *INDOCRYPT*, 2014.
[2] Naccache *et al.* **Projective Coordinates Leak**. *EUROCRYPT*, 2004.
[3] Aldaya *et al.* **From A to Z: Projective Coordinates Leakage in the Wild**. *CHES*, 2020.
[4] 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.

Demo