Testing Side-Channel Security of Cryptographic Implementations against Future Microarchitectures

Chitchanok Chuengsatiansup and Marco Guarnieri

RWC 2025

Joint work with Gilles. Barthe, Marcel Böhme, Sunjay Cauligi,

Daniel Genkin, David Mateos Romero, Peter Schwabe, David Wu, and Yuval Yarom



Chitchanok Chuengsatiansup and Marco Guarnieri Testing Side-Channel Security of Cryptographic Implementations



• Microarchitectural optimization ightarrow speed up computation ightarrow

- Microarchitectural optimization ightarrow speed up computation ightarrow
- Sanchez Vicarte et al., ISCA 2021
 - identify many security-critical optimization proposals (***)

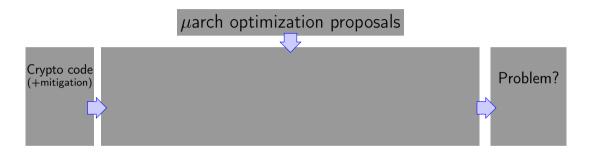
- Microarchitectural optimization ightarrow speed up computation ightarrow
- Sanchez Vicarte et al., ISCA 2021
 - identify many security-critical optimization proposals (>>)
- How to evaluate security impacts?

- Microarchitectural optimization ightarrow speed up computation ightarrow
- Sanchez Vicarte et al., ISCA 2021
 - identify many security-critical optimization proposals (***)
- How to evaluate security impacts?
 - cryptographic implementations

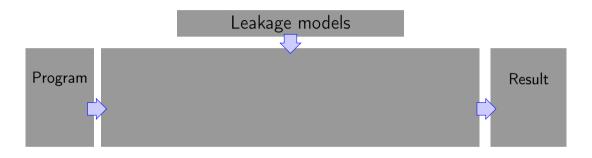
- Microarchitectural optimization ightarrow speed up computation ightarrow
- Sanchez Vicarte et al., ISCA 2021
 - identify many security-critical optimization proposals (***)
- How to evaluate security impacts?
 - cryptographic implementations
 - existing mitigations \rightarrow do they really help preventing leaks?

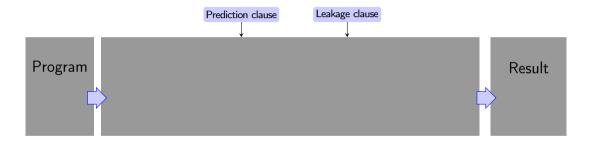
- Microarchitectural optimization ightarrow speed up computation ightarrow
- Sanchez Vicarte et al., ISCA 2021
 - identify many security-critical optimization proposals (***)
- How to evaluate security impacts?
 - cryptographic implementations
 - existing mitigations \rightarrow do they really help preventing leaks?
 - future microarchitectural optimizations

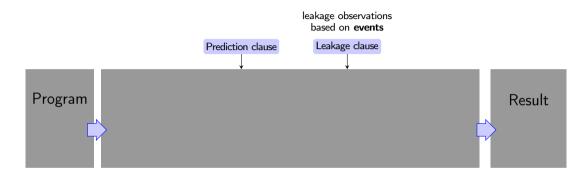
Our framework

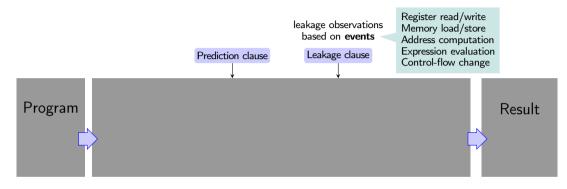


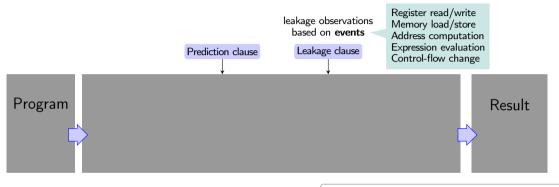




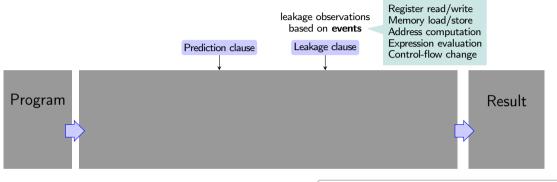




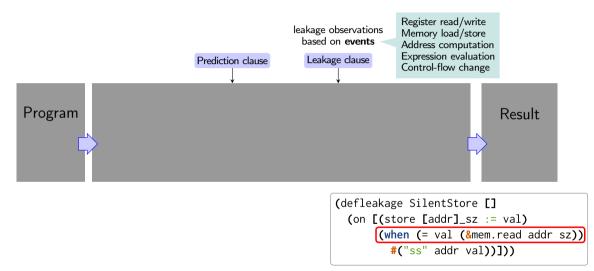


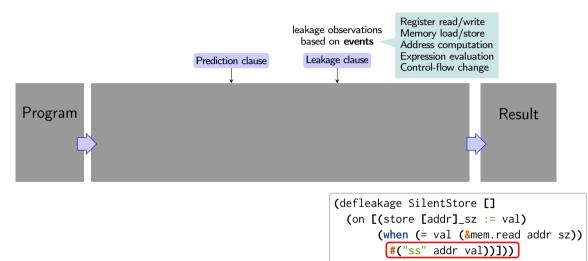


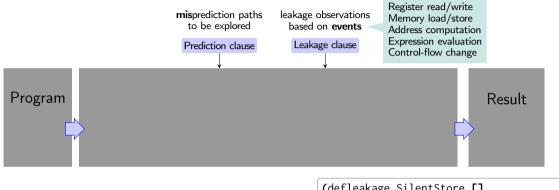
(defleakage SilentStore []
 (on [(store [addr]_sz := val)
 (when (= val (&mem.read addr sz))
 #("ss" addr val))]))



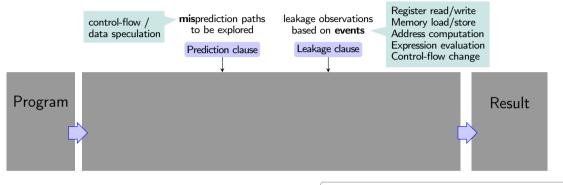
(defleakage SilentStore []
(on [(store [addr]_sz := val)
(when (= val (&mem.read addr sz))
#("ss" addr val))]))



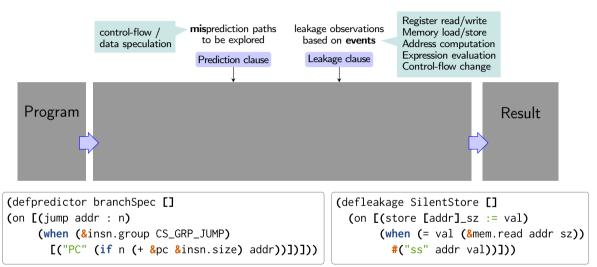


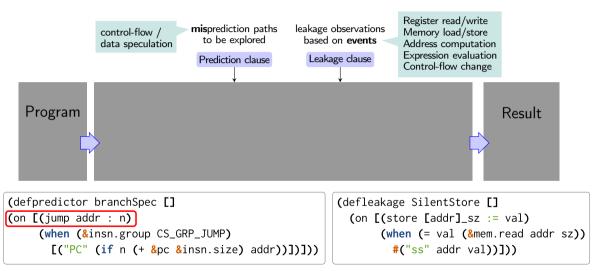


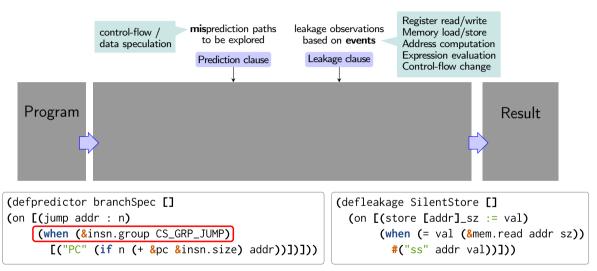
(defleakage SilentStore []
 (on [(store [addr]_sz := val)
 (when (= val (&mem.read addr sz))
 #("ss" addr val))]))

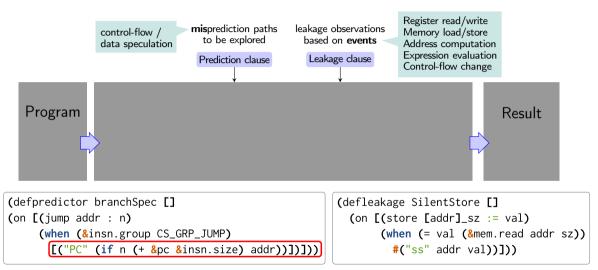


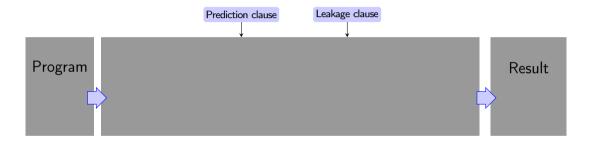
(defleakage SilentStore []
 (on [(store [addr]_sz := val)
 (when (= val (&mem.read addr sz))
 #("ss" addr val))]))

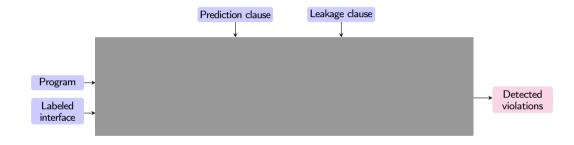




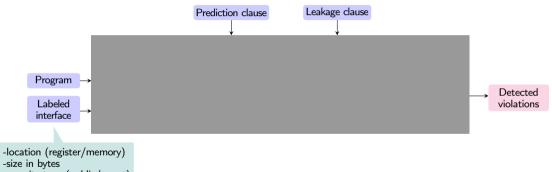


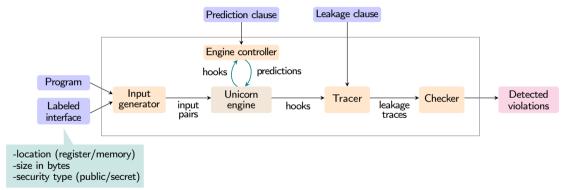


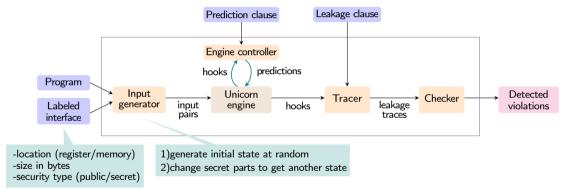


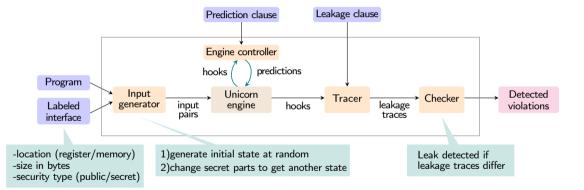












• 18 microarchitecture optimization (leakage clause)

- 18 microarchitecture optimization (leakage clause)
 - CT constant time,

- 18 microarchitecture optimization (leakage clause)
 - CT constant time, SS silent store,

- 18 microarchitecture optimization (leakage clause)
 - CT constant time, SS silent store, RFC register file compression,

- 18 microarchitecture optimization (leakage clause)
 - CT constant time, SS silent store, RFC register file compression, CS computation simplification,

- 18 microarchitecture optimization (leakage clause)
 - CT constant time, SS silent store, RFC register file compression, CS computation simplification, OP operand packing,

- 18 microarchitecture optimization (leakage clause)
 - CT constant time, SS silent store, RFC register file compression, CS computation simplification, OP operand packing, CR computation reuse,

- 18 microarchitecture optimization (leakage clause)
 - CT constant time, SS silent store, RFC register file compression, CS computation simplification, OP operand packing, CR computation reuse, CC cacheline compression,

- 18 microarchitecture optimization (leakage clause)
 - CT constant time, SS silent store, RFC register file compression, CS computation simplification,OP operand packing,CR computation reuse, CC cacheline compression,PF prefetching,

- 18 microarchitecture optimization (leakage clause)
 - CT constant time, SS silent store, RFC register file compression,
 CS computation simplification, OP operand packing, CR computation reuse,
 CC cacheline compression, PF prefetching, + their variations

- 18 microarchitecture optimization (leakage clause)
 - CT constant time, SS silent store, RFC register file compression, CS computation simplification,OP operand packing,CR computation reuse, CC cacheline compression,PF prefetching,+ their variations
- 6 execution models (prediction clause)

- 18 microarchitecture optimization (leakage clause)
 - CT constant time, SS silent store, RFC register file compression, CS computation simplification,OP operand packing,CR computation reuse, CC cacheline compression,PF prefetching,+ their variations
- 6 execution models (prediction clause)
 - sequential,

- 18 microarchitecture optimization (leakage clause)
 - CT constant time, SS silent store, RFC register file compression, CS computation simplification,OP operand packing,CR computation reuse, CC cacheline compression,PF prefetching,+ their variations
- 6 execution models (prediction clause)
 - sequential, conditional branch,

- 18 microarchitecture optimization (leakage clause)
 - CT constant time, SS silent store, RFC register file compression, CS computation simplification,OP operand packing,CR computation reuse, CC cacheline compression,PF prefetching,+ their variations
- 6 execution models (prediction clause)
 - sequential, conditional branch, straight-line,

- 18 microarchitecture optimization (leakage clause)
 - CT constant time, SS silent store, RFC register file compression, CS computation simplification,OP operand packing,CR computation reuse, CC cacheline compression,PF prefetching,+ their variations
- 6 execution models (prediction clause)
 - sequential, conditional branch, straight-line, store bypass,

- 18 microarchitecture optimization (leakage clause)
 - CT constant time, SS silent store, RFC register file compression, CS computation simplification, OP operand packing, CR computation reuse, CC cacheline compression, PF prefetching, + their variations
- 6 execution models (prediction clause)
 - sequential, conditional branch, straight-line, store bypass, return address

- 18 microarchitecture optimization (leakage clause)
 - CT constant time, SS silent store, RFC register file compression, CS computation simplification,OP operand packing,CR computation reuse, CC cacheline compression,PF prefetching,+ their variations
- 6 execution models (prediction clause)
 - sequential, conditional branch, straight-line, store bypass, return address (x2)

- 18 microarchitecture optimization (leakage clause)
 - CT constant time, SS silent store, RFC register file compression, CS computation simplification, OP operand packing, CR computation reuse, CC cacheline compression, PF prefetching, + their variations
- 6 execution models (prediction clause)
 sequential, conditional branch, straight-line, store bypass, return address (x2)

- 18 microarchitecture optimization (leakage clause)
 - CT constant time, SS silent store, RFC register file compression, CS computation simplification, OP operand packing, CR computation reuse, CC cacheline compression, PF prefetching, + their variations
- 6 execution models (prediction clause)
 sequential, conditional branch, straight-line, store bypass, return address (x2)
- 8 cryptographic primitives

- 18 microarchitecture optimization (leakage clause)
 - CT constant time, SS silent store, RFC register file compression, CS computation simplification, OP operand packing, CR computation reuse, CC cacheline compression, PF prefetching, + their variations
- 6 execution models (prediction clause)
 sequential, conditional branch, straight-line, store bypass, return address (x2)
- 8 cryptographic primitives
 - AES, SHA512, HMAC Stream-XOR, Salsa20, Poly1305, Ed25519, X25519

- 18 microarchitecture optimization (leakage clause)
 - CT constant time, SS silent store, RFC register file compression, CS computation simplification, OP operand packing, CR computation reuse, CC cacheline compression, PF prefetching, + their variations
- 6 execution models (prediction clause)
 sequential, conditional branch, straight-line, store bypass, return address (x2)
- 8 cryptographic primitives
 - AES, SHA512, HMAC Stream-XOR, Salsa20, Poly1305, Ed25519, X25519
- 5 popular libraries

- 18 microarchitecture optimization (leakage clause)
 - CT constant time, SS silent store, RFC register file compression, CS computation simplification, OP operand packing, CR computation reuse, CC cacheline compression, PF prefetching, + their variations
- 6 execution models (prediction clause)
 sequential, conditional branch, straight-line, store bypass, return address (x2)
- 8 cryptographic primitives
 - AES, SHA512, HMAC Stream-XOR, Salsa20, Poly1305, Ed25519, X25519
- 5 popular libraries
 - libsodium, libjade, libnettle, cryptlib, rust-crypto

- 18 microarchitecture optimization (leakage clause)
 - CT constant time, SS silent store, RFC register file compression, CS computation simplification, OP operand packing, CR computation reuse, CC cacheline compression, PF prefetching, + their variations
- 6 execution models (prediction clause)
 sequential, conditional branch, straight-line, store bypass, return address (x2)
- 8 cryptographic primitives
 - AES, SHA512, HMAC Stream-XOR, Salsa20, Poly1305, Ed25519, X25519
- 5 popular libraries
 - libsodium, libjade, libnettle, cryptlib, rust-crypto

25 implementations

	СТ		SS			RFO	2		CS		OP	C	R	C	C		PF	
libsodium			٠ı	$\cdot I_0$	•	•0	٠N	·	·т	٠N	•	· ·	٠A	FPC	BDI	·NL	۰s	·DD
salsa20	***	٠	٠	***	٠	٠	٠	٠	***	٥	•	•	٠	•	+	***	***	***
poly1305	***	•	•	•	•	•	+	•	+	۵	•	•	+	•	•	***	***	٥
sha512	***	+	•	•	•	+	+	•	+	٥.	•	•	+	•	•	- 🐝	٥.	٥
hmac	***	٠	٠.	٠.	٠	٠.	٠.	•	٥.	٥.	•	•	٠.	•	٠.	٥.	٥.	٥,
ed25519	- 🐝	٠	٠.	٠.	٠.	٠.	٠.	٠	٠.	***	٠.	•	٠.	٠.	٠.	- 🐝	***	- 🐝
x25519	***	+	•	•	٠	+	+	•	+	۵	•	•	+	•	•	***	***	•
stream-xor	***	•	•	***	•	•	+	•	***	٥	•	•	•	•	+	***	***	٥
cryptlib																		
aes-cbc	•	٠	٠	٠	٠	٠	:	:	٠	٥	•	+	٠	•	•	•	•	٠
sha512	***	•	•	٠	•	•	•	•	•	٥	•	•	•	•	•	***	***	***
libnettle																		
aes-cbc	•	٠	٠	٠	٠	٠	٠	•	٠	٥	•	•	٠	•	+	•	٠	***
salsa20	***	•	***	***	•	•	•	•	****	٥	•	***	***	•	•	•	٥	٥
sha512	***	•	•	•	•	•	•	•	•	۰.	•	•	•	•	•	- 🐝	***	* 0 ×
hmac	***	+	•	•	•	•	•	•	**	٥	•	***	***	•	•	۰	٥	٥
ed25519	٥.	•	٠.	٠.	٠,	٠.	•	•	٠.	٥.	•	•	٠.	•	٠.	۰.	٥,	*. &
x25519	246	•	٠.	•,	٠.	•,	•,	•	٠.	٥,	•	•	•	•	246	246	٥,	٥,
rust-crypto	,																	
salsa20	♦	٠	٠	***	٠	٠	٠	•	٥,	٥	•	***	***	•	•	♦	٥,	\$
poly1305	- 🐝	•	٠.	٠.	٠	٠.	٠.	•	٠.	- 🐝	•	•	٠.	•	٠.	- 🐝	***	٥,
sha512	***	•	•	•	•	•	+	•	•	٥	•	•	•	•	•	- 🐝	***	٥
x25519	٥,	٠,	٠.	٠.	٠,	٠.	٠.	•	٠.	٥,	•	•	٠.	•	٠.	٥,	٥,	٥,
stream-xor	***	•	•	***	•	٠	+	•	٥	٥	•	***	***	•	+	- 🐝	٥	٥
libjade																		
salsa20	***	٠	***	***	٠	٠	٠	+	***	٥	+	***	***	•	+	***	***	***
poly1305	***	+	***	***	٠	+	+	•	•	•	•	•	•	•	•	•	٥	+
sha512	***	+	•	•	+	٠	+	+	+	٥	•	***	***	•	•	- 🐝	***	- 888
x25519	٥,	٠,	٠.	٥,	٠	•	•	•	٠.	٥.	•	•	٠.	•	٥,	۵,	٥,	۰.
stream-xor	- 🐝	+	***	***	•	+	+	•	٥	٥	•	***	- 🐝	•	+	- 🐝	***	٥

Chitchanok Chuengsatiansup and Marco Guarnieri Testing Side-Channel Security of Cryptographic Implementations

• All analyzed implementations leak

	СТ		SS			RFC	2		CS		OP	C	R	C	С		PF	
libsodium		·	٠ı	$\cdot I_0$		•0	·N	· ·	·т	·N			·A	FPC	BDI	·NL	۰s	·DD
salsa20	***	•	•	***	٠	+	+	٠	***	٥	•	•	٠	•	•	***	***	***
poly1305	-	•		•				•	•	\$	•	•		•	•	-	- ***	\$
sha512	***	•			•		•	•	•	۵.	•	•		•	•	-	۵.	۵.
hmac	***	•	•	•	•	•	•	•	۵.	۸.	•	•	•	•	•	۵.	•	۵.
ed25519	-		÷.		•		•	•		-	•	1 🗼	•		•	1	-	-
x25519	***	•		•	•			•	÷	\$	•	•	•	•	•	•	***	•
stream-xor	***	•	•	***	٠	+	+	•	***	۰	•	•	+	•	+	-	***	۰
cryptlib																		
aes-cbc	٠	•	٠	٠	٠	:	•	٠	٠	٥	•	•	٠	•	٠	•	٠	٠
sha512	***	•	•	•	•	•	•	•	•	٥	•	•	•	:	+	***	***	***
libnettle																		
aes-cbc	٠	•	٠	٠	٠	٠	٠	٠	•	٥	+	•	٠	•	•	•	•	***
salsa20	***	•	***	***	•	+	•	•	****	٥	•	288	***	•	•	۰	٥	٥
sha512	***	•	•	•	•	•	•	•	•	۰.	•	•	•	•	•	- 🐝	- 🐝	* * *
hmac	***	•	•	•	•	+	•	•	***	٥	•	288	***	•	•	۰	٥	٥
ed25519	٥,	•	٠.	•	٠,	٠.	٠.	•	٠.	٥,	•	•	٠.	•	٠.	٥,	۰.	٠.
x25519	246	•	٠.	•,	•	٠.	♠,	۰.	٠.	٥,	•	•	٠.	•	246	246	٥,	٥,
rust-crypto																		
salsa20	♦	•	+	***	٠	+	+	•	۰.	٥	+	***	***	+	+	♦	۰.	٥
poly1305	- 🐝	•	♦,	•	٠,	•	•	•	٠.	- 🐝	•	•	•	•	♠,	- 🐝	***	٥,
sha512	***	•	•	•	•	•	•	•	•	٥	•	•	•	•	•	- 🐝	***	٥
x25519	٥,	•	٠.	•	٠,	•	•	•	٠.	٥,	•	•	٠.	•	٠.	٥,	٥,	٥,
stream-xor	***	•	•	***	•	+	+	•	٥	٥	•	***	***	•	•	- 🐝	٥	٥
libjade																		
salsa20	***	+	***	***	٠	+	+	+	***	٥	+	***	***	+	+	***	***	***
poly1305	***	•	***	***	+	+	+	•	+	•	•	•	+	•	•	♦	٥	+
sha512	***	•	•	•	+	•	•	•	•	٥	•	88	***	•	•	***	***	***
x25519	۰.	•	٠.	٥,	٠,	•	•	٠	٠.	۰.	•	•	•	•	٥,	٥,	٥,	٥,
stream-xor	***	+	***	***	+	+	+	٠	٥	۰	+	- 🐝	***	•	•	***	***	٥

- All analyzed implementations leak
- Memory-safe languages (e.g., Rust) not significantly mitigate leaks

	СТ		SS		RFC				CS		OP	C	R	C	С			
libsodium	•	•	٠ı	·I0	•	•0	٠N	· ·	·т	٠N	•		٠A	FPC	BDI	·NL	۰s	·DE
salsa20	***	٠	٠	***	٠	٠	٠	٠	***	٥	٠	٠	٠	٠	٠	***	***	***
poly1305	***	•	+	+	•	+	+	•	+	٥	•	•	•	•	•	***	***	۵
sha512	***	•	•	+	•	٠	+	•	+	٥.	•	•	•	•	•	-	٥.	۵
hmac	***	٠.	٠.	•	٠	٠.	٠.	٠.	٥.	٥.	•	•	٠.	٠.	٠.	٥.	٥.	۰.
ed25519	***	٠.	٠.	٠.	٠.	٠.	٠.	٠	٠.	***	•	٠.	٠.	•	٠.	***	***	- 988
x25519	***	٠	•	+	٠	٠	+	٠	•	٥	•	•	+	•	•	***	***	+
stream-xor	***	•	٠	***	•	٠	•	•	****	٥	•	•	•	•	•	***	****	
cryptlib																		
aes-cbc	٠	٠	٠	٠	٠	٠	٠	٠	٠	٥	•	٠	٠	•	٠	•	٠	٠
sha512	***	:	•	•	•	٠	:	•	•	\$	•	:	•	:	•	***	***	***
libnettle																		
aes-cbc	٠	٠	٠	٠	٠	٠	٠	٠	٠	٥	•	٠	٠	٠	٠	٠	٠	***
salsa20	***	•	***	***	•	٠	•	•	***	٥	•	***	***	•	•	•	٥	٥
sha512	***	•	•	•	•	•	•	•	•	٥.	•	•	•	•	•	- 🐝	***	•0×
hmac	***	•	•	•	•	٠	•	•	***	٥	•	***	***	•	•	٥	٥	٥
ed25519	٥,	٠.	٠.	•	٠.	٠.	•	٠.	٠.	٥,	•	•	٠.	•	٠.	٥,	٥,	•
x25519	246	•	٠.	•	٠.	•,	•,	•	♠,	٥,	•	•	٠.	•	246	246	٥,	٥,
rust-crypto																		
salsa20	♦	٠	٠	***	٠	٠	٠	٠	۰,	٥	+	***	***	•	٠	♦	\$,	\$
poly1305	- 🐝	•	٠.	•	٠,	•	•	٠	٠.	₩.	•	•	•	•	٠.	- 🐝	- 🐝	٥,
sha512	***	•	•	•	•	•	•	•	•	٥	•	•	•	•	•	- 🐝	***	♦
x25519	٥,	٠,	٠.	٠.	٠.	٠.	٠.	٠	٠.	٥,	•	•	٠.	٠.	٠.	٥,	٥,	۰.
stream-xor	- 🐝	+	•	***	٠	٠	+	•	٥	٥	+	***	***	+	+	***	٥	\$
libjade																		
salsa20	***	٠	***	***	٠	٠	٠	٠	***	٥	+	***	***	٠	٠	***	***	***
poly1305	***	+	***	***	+	٠	+	•	+	•	•	•	•	•	•	۰	٥	•
sha512	***	+	+	+	+	٠	+	•	•	٥	•	***	***	•	•	***	***	***
x25519	۵.	٠	٠.	٥,	٠	٠.	•	•	٠.	٥,	•	•	٠.	•	٥,	۵.	٥,	۰.
stream-xor	***	•	***	***	•		•	•	0	0	•	***	***	•	•	***	***	•

- All analyzed implementations leak
- Memory-safe languages (e.g., Rust) not significantly mitigate leaks
- CT programming (e.g., libsodium, libjade) does not fully prevent leaks

	СТ		SS			RFC	2		CS		OP	C	R	C	С		PF			
libsodium	•	÷.	٠ı	$\cdot I_0$		•0	٠N	•	·т	٠N	•	· ·	٠A	FPC	BDI	·NL	٠s	·DD		
salsa20	***	٠	٠	***	٠	٠	٠	٠	***	٥	٠	+	٠	٠	٠	***	***	***		
poly1305	***	•	•	•	٠	+	•	•	•	٥	•	•	•	•	•	***	***	۰		
sha512	***	+	•	+	٠	+	•	•	+	٥.	•	+	•	•	•	- 🐝	٥.	٥		
hmac	***	٠.	٠.	•	٠	٠.	٠.	•	٥.	۵.	•	•	٠.	•	٠.	٥.	٥.	٥.		
ed25519	***	٠	٠.	٠.	۰.	٠.	٠.	٠.	٠.	****	•	٠.	٠.	•	٠.	***	***	- 000		
x25519	***	+	•	•	٠	+	+	٠	•	٥	•	•	+	•	•	***	***	+		
stream-xor	***	+	•	***	٠	+	+	•	***	٥	+	+	+	+	+	***	***	٥		
cryptlib																				
aes-cbc	٠	٠	٠	•	٠	٠	٠	٠	٠	٥	+	+	٠	•	٠	•	٠	٠		
sha512	***	•	•	•	٠	٠	٠	:	٠	٥	•	•	+	•	•	***	***	***		
libnettle																				
aes-cbc	٠	٠	•	•	٠	٠	٠	٠	٠	٥	+	+	•	•	•	•	٠	***		
salsa20	***	•	***	***	٠	•	•	•	***	٥	•	***	***	•	•	•	٥	٥		
sha512	***	•	•	•	٠	•	•	•	•	۰.	•	•	•	•	•	- 🐝	***	•0×		
hmac	***	•	•	•	٠	•	•	•	***	٥	•	***	***	•	•	٥	٥	٥		
ed25519	٥,	•	٠.	•	٠.	٠.	٠.	•	٠.	٥,	•	•	٠.	•	٠.	٥,	۰.	•		
x25519	246	٠.	♠,	•	۰.	٠.	•,	•	♠,	٥,	•	•	٠.	•	246	246	٥,	٥,		
rust-crypto																				
salsa20	♦	٠	٠	***	٠	٠	٠	٠	٥,	\$	+	***	***	•	٠	♦	\$,	\$		
poly1305	- 🐝	٠	♦,	•	٠.	٠.	•	•	•	- 🐝	•	•	٠.	•	٠.	- 🐝	***	٥,		
sha512	***	•	•	•	٠	•	•	•	•	۰	•	•	•	•	•	- 🐝	***	♦		
x25519	٥,	٠.	٠.	•	٠.	٠.	•	٠	٠.	٥,	•	٠	٠.	٠.	٠.	٥,	٥,	٥,		
stream-xor	***	+	•	***	٠	+	+	•	٥	٥	•	***	***	•	•	- 🐝	٥			
libjade																				
salsa20	***	٠	***	***	٠	٠	٠	٠	***	٥	+	***	***	+	•	***	***	***		
poly1305	***	+	***	***	٠	٠	+	+	•	•	+	+	•	•	•	♦	٥	•		
sha512	***	+	•	•	٠	+	+	+	+	٥	+	- 🐝	***	•	•	***	***	***		
x25519	٥,	٠.	٠.	۰.	٠.	٠.	•	٠	٠.	٥,	•	•	♦.	٠.	٥,	۵.	۰.	۰.		
stream-xor	***	+	***	***	٠	+	+	•	٥	٥	•	- 🐝	- 🐝	•	•	***	***	٥		

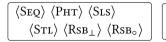
Chitchanok Chuengsatiansup and Marco Guarnieri Testing Side-Channel Security of Cryptographic Implementations

- All analyzed implementations leak
- Memory-safe languages (e.g., Rust) not significantly mitigate leaks
- CT programming (e.g., libsodium, libjade) does not fully prevent leaks
- Majority of leaks are present even without prediction clause

	СТ		SS			RFO	2		CS		OP	C	R	C	C			
libsodium		•	٠ı	$\cdot I_0$	•	•0	٠N	· ·	·т	٠N	•	· ·	٠A	FPC	BDI	·NL	۰s	٠DI
salsa20	***	٠	٠	***	٠	٠	٠	٠	***	٥	٠	+	٠	٠	٠	***	***	***
poly1305	***	•	•	•	•	•	+	•	+	٥	•	+	+	•	•	***	***	۵
sha512	***	•	•	•	•	+	+	•	•	۵.	•	•	+	•	•	-	۵.	•
hmac	***	٠	٠.	٠.	٠	٠.	•	•	٥.	۵.	•	•	٠.	•	٠.	۵.	۵.	۰.
ed25519	***	•	•	•	٠	•	•	•	•	-	•	•	•	•	•	***	***	*
x25519	***	٠	•	•	٠	٠	+	+	+	۵	+	+	+	•	+	***	***	+
stream-xor	***	•	٠	***	•	+	+	•	***	٥	•	•	•	•	+	***	****	\$
cryptlib																		
aes-cbc	•	٠	٠	٠	٠	:	٠	٠	:	٥	•	+	٠	•	٠	•	٠	٠
sha512	***	•	•	•	•	•	+	+	•	٥	•	•	•	•	+	***	***	**
libnettle																		
aes-cbc	•	٠	٠	٠	٠	٠	٠	•	•	٥	+	+	٠	•	•	•	٠	**
salsa20	***	•	***	***	٠	•	•	•	***	٥	•	***	***	•	•	♦	٥	•
sha512	***	•	•	•	•	•	+	•	•	۰.	•	+	•	•	•	- 🐝	***	• •
hmac	***	•	٠	•	٠	٠	•	•	***	٥	•	***	***	•	•	♦	٥	
ed25519	٥.	٠.	٠.	•	٠,	٠.	•	•	٠.	٥,	•	•	٠.	•	٠.	٥,	٥.	•
x25519	246	٠,	٠.	٠.	•	•	. ♠,	٠	♠,	٥,	•	•	•	•	246	246	٥,	٥,
rust-crypto	,																	
salsa20	♦	٠	٠	***	٠	٠	٠	+	۰,	٥	+	***	***	•	•	♦	۰.	\$
poly1305	- 🐝	•	٠.	•	٠	•	•	•	٠.	* **	•	•	•	•	٠.	- 🐝	***	٥,
sha512	***	•	•	•	•	•	•	•	•	٥	•	•	•	•	+	- 🐝	***	
x25519	٥,	٠,	٠.	•	٠	٠.	٠.	•	٠.	٥,	•	٠	٠.	•	٠.	٥,	٥,	۵,
stream-xor	***	•	•	***	•	•	+	•	٥	٥	•	***	***	•	+	***	٥	
libjade																		
salsa20	***	٠	***	***	٠	٠	٠	+	***	٥	+	***	***	•	•	***	***	- 00
poly1305	***	+	***	***	+	٠	+	•	•	•	+	+	•	•	•	♦	٥	•
sha512	***	+	•	•	+	٠	+	•	•	٥	+	- 🐝	***	•	•	***	***	**
x25519	٥,	٠	٠.	۰.	٠	٠.	•	•	٠.	٥,	•	٠	٠.	•	٥,	۵.	٥,	۵,
stream-xor	***	•	***	- 🐝	٠	+	+	•	۵	۵	•	***	***	•	+	***	***	•

Chitchanok Chuengsatiansup and Marco Guarnieri Testing Side-Channel Security of Cryptographic Implementations

- All analyzed implementations leak
- Memory-safe languages (e.g., Rust) not significantly mitigate leaks
- CT programming (e.g., libsodium, libjade) does not fully prevent leaks
- Majority of leaks are present even without prediction clause





	СТ		SS			RFC	2		CS		OP	C	R	C	С		PF	
libsodium	•	· ·	٠ı	$\cdot I_0$	•	•0	٠N	· ·	·т	٠N	· ·	· ·	٠A	FPC	BDI	·NL	۰s	·DD
salsa20	***	٠	٠	***	٠	٠	٠	٠	***	٥	٠	•	٠	٠	٠	***	***	***
poly1305	***	•	•	•	•	+	•	•	•	٥	•	•	•	•	•	***	***	۰
sha512	***	•	•	+	•	+	•	•	+	٥.	•	•	•	•	•	- 🐝	٥.	٥
hmac	***	•	٠.	•	٠	٠.	٠	•	۰.	۵.	•	•	٠.	•	٠.	۵.	۵.	۵.
ed25519	***	•	٠.	٠.	٠.	٠.	٠.	٠.	٠.	***	•	•	٠.	•	٠.	***	***	- 🐝
x25519	***	•	•	+	٠	+	+	٠	•	٥	•	•	+	•	•	***	***	+
stream-xor	***	•	•	***	•	+	•	•	***	٥	•	•	•	•	•	***	***	٥
cryptlib																		
aes-cbc	٠	•	٠	٠	٠	٠	٠	٠	٠	٥	•	+	٠	•	٠	•	٠	٠
sha512	***	•	•	+	•	+	+	•	•	٥	•	•	•	•	+	***	***	***
libnettle																		
aes-cbc	٠	•	٠	٠	٠	٠	٠	٠	٠	٥	•	+	٠	•	•	•	٠	***
salsa20	***	•	***	***	•	•	•	•	***	٥	•	***	***	•	•	•	٥	٥
sha512	***	•	•	•	•	•	•	•	•	٥.	•	•	•	•	•	- 🐝	***	• 0 ×
hmac	***	•	•	•	•	•	•	•	***	٥	•	***	***	•	•	٥	٥	٥
ed25519	٥.	•	٠.	•	٠	•	٠.	•	٠.	٥.	•	•	•	•	•	٥.	٥.	•,
x25519	246	•	٠.	•	۰,	•	•,	•	٠.	٥,	•	•	٠.	•	246	246	٥,	٥,
rust-crypto																		
salsa20	♦	•	٠	***	٠	٠	٠	٠	٥,	٥	+	***	***	•	٠	♦	\$,	\$
poly1305	- 🐝	•	٠.	٠.	٠,	•	•	•	٠.	***	•	•	٠.	•	٠.	- 🐝	* ***	۰.
sha512	***	•	•	•	•	•	•	•	•	٥	•	•	•	•	•	- 🐝	***	♦
x25819	٥,	•	٠.	•	٠,	•	•	٠	٠.	٥,	•	•	٠.	٠.	٠.	٥,	٥,	٥,
stream-xor	***	•	•	***	+	+	•	•	٥	٥	•	***	***	•	+	- 🐝	٥	٥
libjade																		
salsa20	***	•	***	***	٠	+	٠	٠	***	٥	+	***	***	+	+	***	***	***
poly1305	***	•	***	***	+	•	+	+	•	+	•	•	•	•	•	♦	٥	•
sha512	***	•	•	•	+	•	+	•	•	٥	•	***	***	•	•	***	***	***
x25519	۰.	•	•	٥,	٠,	•	•	٠	•	٥,	•	٠	•	•	٥,	۵,	٥,	٥,
stream-xor	***	•	***	***	٠	•	•	+	٥	٥	•	***	***	•	•	- 🐝	***	٥

```
fe25519_cswap(fe25519_limb f[5], fe25519_limb g[5], bool b)
Ł
    mask = (-(int64_t) b);
   x[0..5] = f[0..5] \cap g[0..5];
    x[0..5] \&= mask;
    f[0..5] = f[0..5] ^ x[0..5];
   g[0..5] = f[0..5] ^ x[0..5];
}
```

```
secret
fe25519_cswap(fe25519_limb f[5], fe25519_limb g[5], bool b)
Ł
    mask = (-(int64_t) b);
    x[0..5] = f[0..5] \cap g[0..5];
    x[0..5] \&= mask:
    f[0..5] = f[0..5] ^ x[0..5]:
    g[0..5] = f[0..5] ^ x[0..5];
}
```

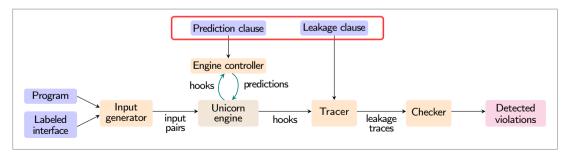
```
secret
fe25519_cswap(fe25519_limb f[5], fe25519_limb g[5], bool b)
ſ
                                              mask = 0 \rightarrow result = 0
    mask = (-(int64_t) b);
                                           RFC: compress to zero register
    x[0..5] = f[0..5] \cap g[0..5];
    x[0..5] \&= mask:
    f[0..5] = f[0..5] ^ x[0..5]:
    g[0..5] = f[0..5] ^ x[0..5];
}
```

```
secret
fe25519_cswap(fe25519_limb f[5], fe25519_limb g[5], bool b)
ſ
                                                 mask = 0 \rightarrow result = 0
    mask = (-(int64_t) b);
                                             RFC: compress to zero register
    x[0..5] = f[0..5] \cap g[0..5];
                                        mask = 0 \rightarrow x = 0 \rightarrow xor 0 not changed
    x[0..5] \&= mask:
                                                  CS: xor is simplified
    f[0..5] = f[0..5] ^ x[0..5]:
    g[0..5] = f[0..5] ^ x[0..5];
}
```

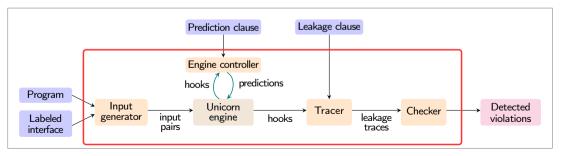
```
secret
fe25519_cswap(fe25519_limb f[5], fe25519_limb g[5], bool b)
ſ
                                                  mask = 0 \rightarrow result = 0
    mask = (-(int64_t) b);
                                              RFC: compress to zero register
    x[0..5] = f[0..5] \cap g[0..5];
                                         mask = 0 \rightarrow x = 0 \rightarrow xor 0 not changed
    x[0..5] \&= mask:
                                                   CS: xor is simplified
    f[0..5] = f[0..5] ^ x[0..5];
                                        values not swapped \rightarrow memory not modified
    g[0..5] = f[0..5] ^ x[0..5];
                                                 SS: stores are suppressed
}
```



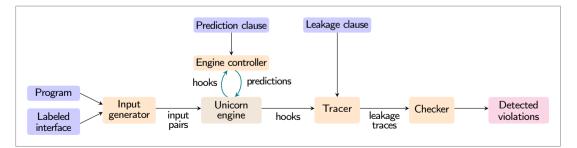
• LMSPEC: Language for specifying leakage models at the ISA level



- $\bullet~\mathrm{LmSPEC}:$ Language for specifying leakage models at the ISA level
- LMTEST: Testing framework for automatically detecting leaks



- $\bullet~\mathrm{LmSPEC}:$ Language for specifying leakage models at the ISA level
- LMTEST: Testing framework for automatically detecting leaks
- Extensive evaluation on cryptographic algorithms \rightarrow all leak!



- $\bullet~\mathrm{LmSPEC}:$ Language for specifying leakage models at the ISA level
- LMTEST: Testing framework for automatically detecting leaks
- Extensive evaluation on cryptographic algorithms $\rightarrow all \ leak!$
- For more details, see https://arxiv.org/pdf/2402.00641

