Supplemental Material of Vivienne: Relational Verification of Cryptographic Implementations in WebAssembly

Rodothea Myrsini Tsoupidi
KTH, Royal Institute of Technology
Stockholm, Sweden
tsoupidi@kth.se

Musard Balliu
KTH, Royal Institute of Technology
Stockholm, Sweden
musard@kth.se

Benoit Baudry
KTH, Royal Institute of Technology
Stockholm, Sweden
baudry@kth.se

This document contains supplemental material for the paper "Vivienne: Relational Verification of Cryptographic Implementations in WebAssembly" [1].

References

[1] R. M. Tsoupidi, M. Balliu, and B. Baudry, "Vivienne: Relational Verification of Cryptographic Implementations in WebAssembly," in *2021 IEEE Secure Development (SecDev)*. IEEE, 2021, to appear.

Appendix A
Evaluation Results

Table III and Table IV show the complete results of the evaluation for Vivienne$_{\text{unroll}}$ and Vivienne$_{\text{inv}}$, respectively. The experiments for the two tables use a time limit of 90 minutes and the reported time values are in seconds and consist of the average and standard deviation after five runs. The first column shows the file name followed by the function that corresponds to the entry point for the analysis. Column `LoC` shows the number of WebAssembly instructions that the analysis accesses, column `AN time` is the analysis time in seconds. When `AN time` is -1, then Vivienne was not able to successfully analyze the respective implementations, whereas when `AN time` is * for Vivienne$_{\text{inv}}$, then this means that the invariant assertion failed for one of the loops. Column shows the number of discovered timing vulnerabilities. #FS is the number of formulas during the analysis and next column shows the time in seconds for the simplification step. #SS is the number of formulas that Vivienne forwards to the SMT solver, followed by the average number of expressions in each formula, #Exprs, and the solving time `SS time`. #Exprs is the value that decides selecting the *bindings* solver or the *portfolio* solver. In these experiments, for #Expr ≤ 1500, Vivienne uses the *bindings* solver, otherwise the portfolio solver.

For example, the third entry for WHACL* in Table IV shows the results for the analysis of function `Hacl_Poly1305_32_poly1305_mac` from WHACL* module `poly1305`. Vivienne goes through 1440 different WebAssembly instructions, not considering the multiple accesses for loops. The analysis time is 1.55 seconds and the analysis did not discover any timing vulnerabilities, generated 700 formulas that took less than 0.01 seconds to simplify. Of these 700 formulas, 69 where forwarded to the SMT solver, whereas the rest were simple enough for the analysis to infer their result. The average number of expressions in these 69 formulas is 22 expressions and the solving time was less than 0.01 seconds.

A. Vivienne$_{unroll}$ *and* Vivienne$_{inv}$ *comparison*

By comparing Tables III and IV, we notice that the number of queries, #FS, is, in general, larger for Vivienne$_{\text{unroll}}$ than Vivienne$_{\text{inv}}$. The reason for this is that Vivienne$_{\text{unroll}}$ needs to make queries for memory operations and control-flow instructions at every iteration. However, constant-time cryptographic implementations typically use constant memory indexes and often branch on constant values. This means that these queries are simple and in most cases do not require invoking the SMT solver (low #SS). On the other hand, Vivienne$_{\text{inv}}$ has lower #FS than Vivienne$_{\text{unroll}}$ (in most cases) because of the use of an invariant simplifies the analysis of loops. However, Vivienne$_{\text{inv}}$ has increased #SS because first the invariant analysis requires querying the policies of modified variables in the loop that might not be constant values and second, it replaces constant values in `if` statements or memory indexes with symbolic unbound values that increase search space of the formula. In some cases, Vivienne$_{\text{inv}}$ has larger #FS than Vivienne$_{\text{unroll}}$, like in `br_aes_ct_cbcenc_run` of BearSSL -O3, where #FS=157984 for Vivienne$_{\text{inv}}$ and #SS=2793. This is due to path explosion as a result of the invariant-induced overapproximation.

To summarize, we can see three types of complexity sources in our Relational Symbolic Execution (RelSE) analysis: 1) the number of loop iterations, 2) the number of execution paths, and 3) the formula complexity (depends often on the memory). Vivienne$_{\text{inv}}$ reduces 1) but may increases 2) and 3), whereas Vivienne$_{\text{unroll}}$ has higher 1) which may also increase 3), but typically lower 2). Depending on the combined effects of these three complexity sources, either of the two methods may perform better.

Solver	VIVIENNE$_{\text{unroll}}$	VIVIENNE$_{\text{inv}}$
Boolector	45.29%	91.94%
Yices 2	54.71%	6.11%
CVC4	0%	0.33%
Z3	0%	1.62%

TABLE I
PORTFOLIO SOLVER STATISTICS

Solver	VIVIENNE$_{\text{unroll}}$	VIVIENNE$_{\text{inv}}$
Z3 Bindings	6.0%	63.3%
Portfolio Solver	94.0%	36.7%

TABLE II
SOLVER STATISTICS

APPENDIX B
SMT SOLVER

Our approach uses an Satisfiability Modulo Theories (SMT) solver with two modes, the first uses the Z3 OCaml bindings for reduced communication overhead and the second uses a portfolio solver that runs four solvers in parallel. The analysis selects which SMT solver mode to use depending on the number of expressions in the formula. Table II shows the share of formulas that VIVIENNE passes to the bindings and the portfolio solver. The table shows that for VIVIENNE$_{\text{inv}}$, the analysis passes the majority of the queries (72.3%) to the `Z3 Bindings` solver, which means that the queries from VIVIENNE$_{\text{inv}}$ have relatively low number of expressions, an indication on the complexity and the size of the query. For VIVIENNE$_{\text{unroll}}$, the opposite is true, as the analysis passes the majority of the queries (90.4%) to the `Portfolio Solver`. This means that VIVIENNE$_{\text{unroll}}$ passes to the solver mostly queries that contain a large number of expressions, an indication of complexity.

The portfolio solver consists of four solvers, namely Boolector, Yices 2, CVC4, and Z3, that run in parallel and the first to finish reports the result to VIVIENNE. Table I shows the share of answers from each of the four solvers to the queries to the portfolio solver. In Table I, we see that `Z3` and `CVC4` are not able to answer to a large number of queries for either VIVIENNE$_{\text{inv}}$ (3.36% and 0.25%) or VIVIENNE$_{\text{unroll}}$ (0%). For VIVIENNE$_{\text{unroll}}$, `Yices 2` answers the majority, i.e. 79.88% of the queries and `Boolector` answers to 20.12%. For VIVIENNE$_{\text{inv}}$, the opposite is true, namely `Boolector` answers the majority of the queries, i.e. 86.83%, whereas `Yices 2` answers 9.56% of the queries. The difference in the efficiency of the solvers for VIVIENNE$_{\text{inv}}$ and VIVIENNE$_{\text{unroll}}$, depends primarily on the (default) heuristics that they use, which can be beneficial for specific queries. Another parameter that affects the performance of the solvers is the hardware that these solvers run on (see Section **??**) because the speed of the memory and the processor power may affect the performance of each solver. To summarize, our results show that `Boolector` and `Yices 2` are the best performing solvers in the portfolio, but there is no optimal solver for constant-time analysis of VIVIENNE.

bench/function	LoC	AN time	🐞	#FS	FS time	#SS	#Exprs	SS time
CT-wasm								
salsa20/decrypt	515	0.09 ± 0.00	0	602	< 0.01	0		
salsa20/encrypt	512	0.10 ± 0.01	0	602	< 0.01	0		
sha256/transform	372	0.05 ± 0.01	0	926	< 0.01	0		
sha256/update	409	0.18 ± 0.01	0	1312	< 0.01	0		
tea/decrypt	80	< 0.01	0	72	< 0.01	0		
tea/encrypt	80	0.01 ± 0.00	0	72	< 0.01	0		
TweetNaCl								
core_hsalsa20/core_hsalsa20	356	< 0.01	0	46	< 0.01	0		
core_salsa20/core_salsa20	412	0.01 ± 0.00	0	54	< 0.01	0		
poly1305/crypto_onetimeauth	787	0.11 ± 0.00	0	81	< 0.01	0		
WHACL*								
chacha20/Hacl_Chacha20_chacha20_encrypt	1777	669.91 ± 3.53	0	9665	0.07 ± 2.77	0		
curve25519_51/Hacl_Curve25519_51_scalarmult	-1	-1	0	80896	0.07 ± 0.26	0		
poly1305/Hacl_Poly1305_32_poly1305_mac	1440	1.34 ± 0.01	0	829	< 0.01	0		
salsa20/Hacl_Salsa20_salsa20_encrypt	1887	162.86 ± 1.56	0	8596	0.02 ± 0.71	0		
sha256/Hacl_Hash_SHA2_hash_256	1147	1323.51 ± 7.13	0	14512	0.09 ± 4.56	0		
sha512/Hacl_Hash_SHA2_hash_512	1550	456.20 ± 4.14	0	12287	0.04 ± 1.62	0		
BearSSL -O0								
aes_big/br_aes_big_cbcenc_run	2089	13.04 ± 0.11	32	1111	< 0.01	32	3711	0.36 ± 0.37
aes_ct/br_aes_ct_cbcenc_run	4857	46.54 ± 0.76	0	4233	0.01 ± 0.13	0		
des_ct/br_des_ct_cbcenc_run	3841	1560.52 ± 6.80	0	23463	0.07 ± 1.23	0		
des_tab/br_des_tab_cbcenc_run	1920	24.94 ± 0.16	8	3301	0.01 ± 0.05	8	262	< 0.01
BearSSL -O3								
aes_big/br_aes_big_cbcenc_run	791	7.89 ± 0.09	32	218	< 0.01	32	3327	0.22 ± 0.22
aes_ct/br_aes_ct_cbcenc_run	1717	1.69 ± 0.01	0	493	< 0.01	0		
des_ct/br_des_ct_cbcenc_run	993	6.49 ± 0.03	0	952	0.01 ± 0.19	0		
des_tab/br_des_tab_cbcenc_run	581	3.20 ± 0.03	8	381	0.01 ± 0.15	8	262	< 0.01
Libsodium -O0								
aead/crypto_aead_chacha20poly1305_encrypt	7720	369.83 ± 1.33	0	11507	0.03 ± 0.48	16	4	0.04 ± 0.00
auth/crypto_auth_hmacsha256	13913	4856.64 ± 27.94	0	47679	0.10 ± 0.52	0		
chacha20/crypto_stream_chacha20	3313	228.04 ± 1.61	0	8756	0.03 ± 0.51	2	4	0.04 ± 0.00
poly1305/crypto_onetimeauth_poly1305_donna	3685	20.78 ± 0.09	0	1671	0.01 ± 0.07	0		
salsa20/crypto_core_salsa20	1628	11.99 ± 0.04	0	3513	< 0.01	0		
sha256/SHA256_Transform	11692	136.11 ± 0.95	0	8299	0.02 ± 0.06	0		
sha256/crypto_hash_sha256	13225	536.25 ± 3.84	0	18712	0.03 ± 0.11	0		
sha512/crypto_hash_sha512	13351	295.80 ± 3.18	0	12993	0.02 ± 0.08	0		
Libsodium -O3								
aead/crypto_aead_chacha20poly1305_encrypt	1971	45.06 ± 0.29	0	896	0.05 ± 0.67	16	4	0.04 ± 0.00
auth/crypto_auth_hmacsha256	3256	562.00 ± 4.19	0	4559	0.12 ± 5.32	0		
chacha20/crypto_stream_chacha20	956	0.29 ± 0.01	0	253	< 0.01	2	4	0.04 ± 0.00
poly1305/crypto_onetimeauth_poly1305_donna	940	11.20 ± 0.07	0	223	0.05 ± 0.58	0		
salsa20/crypto_core_salsa20	483	0.01 ± 0.00	0	52	< 0.01	0		
sha256/SHA256_Transform	2171	0.01 ± 0.00	0	479	< 0.01	0		
sha256/crypto_hash_sha256	2980	28.06 ± 0.66	0	1643	0.02 ± 0.66	0		
sha512/crypto_hash_sha512	2844	6.20 ± 0.06	0	1344	< 0.01	0		
Almeida -O0								
naive_select/ct_select_u32_naive	49	0.03 ± 0.00	1	9	< 0.01	3	15	< 0.01
select_v1/ct_select_u32_v1	149	< 0.01	0	14	< 0.01	0		
select_v2/ct_select_u32_v2	93	< 0.01	0	10	< 0.01	0		
select_v3/ct_select_u32_v3	70	< 0.01	0	9	< 0.01	0		
select_v4/ct_select_u32_v4	70	< 0.01	0	9	< 0.01	0		
sort/sort3	254	0.18 ± 0.00	1	298	< 0.01	14	68	< 0.01
sort_multiplex/sort3_multiplex	276	0.02 ± 0.00	0	89	< 0.01	0		
sort_negative/sort3_negative	209	0.16 ± 0.01	1	245	< 0.01	14	68	< 0.01
Almeida -O3								
naive_select/ct_select_u32_naive	5	< 0.01	0	0		0		
select_v1/ct_select_u32_v1	5	< 0.01	0	0		0		
select_v2/ct_select_u32_v2	5	< 0.01	0	0		0		
select_v3/ct_select_u32_v3	5	< 0.01	0	0		0		
select_v4/ct_select_u32_v4	5	< 0.01	0	0		0		
sort/sort3	84	0.07 ± 0.01	3	21	< 0.01	3	229	0.02 ± 0.01
sort_multiplex/sort3_multiplex	74	0.10 ± 0.00	3	17	< 0.01	3	229	0.02 ± 0.02
sort_negative/sort3_negative	74	0.09 ± 0.01	3	17	< 0.01	3	229	0.02 ± 0.02
lucky13 -O0								
tls1_cbc_remove_padding_lucky13/tls1_..._lucky13	-1	-1	5	24978	0.01 ± 0.06	4027	35698	0.87 ± 0.60
lucky13 -O3								
tls1_cbc_remove_padding_lucky13/tls1_..._lucky13	133	960.17 ± 15.52	5	3144	< 0.01	3106	3080	0.25 ± 1.03

TABLE III

EVALUATION RESULTS WITH VIVIENNE$_{\text{UNROLL}}$

bench/function	LoC	AN time	🐞	#FS	FS time	#SS	#Exprs	SS time
CT-wasm								
salsa20/decrypt	515	38.99 ± 7.31	0	272	< 0.01	160	426	0.23 ± 0.90
salsa20/encrypt	512	57.78 ± 17.51	0	272	< 0.01	160	426	0.35 ± 1.55
sha256/transform	372	1.06 ± 0.03	0	97	< 0.01	36	323	0.02 ± 0.07
sha256/update	409	3.47 ± 0.03	0	123	< 0.01	44	2469	0.06 ± 0.07
tea/decrypt	80	0.17 ± 0.00	0	25	< 0.01	6	99	0.02 ± 0.03
tea/encrypt	80	0.17 ± 0.01	0	25	< 0.01	6	99	0.02 ± 0.03
TweetNaCl								
core_hsalsa20/core_hsalsa20	356	17.28 ± 0.18	0	98	< 0.01	66	291	0.25 ± 0.86
core_salsa20/core_salsa20	412	27.11 ± 5.04	0	106	< 0.01	66	291	0.40 ± 1.71
poly1305/crypto_onetimeauth	787	145.44 ± 0.38	0	116	< 0.01	32	221	4.54 ± 4.55
WHACL*								
chacha20/Hacl_Chacha20_chacha20_encrypt	1777	101.19 ± 0.88	0	2029	0.01 ± 0.46	100	95241	0.73 ± 2.36
curve25519_51/Hacl_Curve25519_51_scalarmult	44234	2007.77 ± 9.08	0	59780	0.03 ± 0.09	5676	80	0.01 ± 0.04
poly1305/Hacl_Poly1305_32_poly1305_mac	1440	1.55 ± 0.01	0	700	< 0.01	69	22	< 0.01
salsa20/Hacl_Salsa20_salsa20_encrypt	1887	230.22 ± 2.83	0	6449	0.03 ± 1.12	311	75631	0.11 ± 0.38
sha256/Hacl_Hash_SHA2_hash_256	1147	4.67 ± 0.05	0	720	< 0.01	197	257	0.01 ± 0.05
sha512/Hacl_Hash_SHA2_hash_512	1550	6.88 ± 0.07	0	832	< 0.01	211	244	0.01 ± 0.07
BearSSL -O0								
aes_big/br_aes_big_cbcenc_run	-1	-1	39	766	< 0.01	146	7296	8.15 ± 10.06
aes_ct/br_aes_ct_cbcenc_run	4857	19.51 ± 0.18	0	4337	< 0.01	50	10	< 0.01
des_ct/br_des_ct_cbcenc_run	-1	-1	14	3630	< 0.01	337	19742	9.07 ± 8.59
des_tab/br_des_tab_cbcenc_run	-1	-1	13	1563	< 0.01	340	12436	6.59 ± 7.45
BearSSL -O3								
aes_big/br_aes_big_cbcenc_run	791	45.45 ± 0.42	32	270	< 0.01	70	3684	0.63 ± 0.94
aes_ct/br_aes_ct_cbcenc_run	-1	-1	9	157984	0.02 ± 0.73	2793	4189	0.32 ± 0.06
des_ct/br_des_ct_cbcenc_run	-1	*	*	256	0.03 ± 0.34	129	3291	0.34 ± 0.27
des_tab/br_des_tab_cbcenc_run	-1	*	*	180	< 0.01	83	7209	0.83 ± 1.75
Libsodium -O0								
aead/crypto_aead_chacha20poly1305_encrypt	-1	-1	3	5207	0.02 ± 0.28	13	207810	5.11 ± 9.05
auth/crypto_auth_hmacsha256	-1	-1	74	473	0.01 ± 0.02	133	113452	13.01 ± 16.88
chacha20/crypto_stream_chacha20	3313	231.17 ± 3.25	0	8756	0.03 ± 0.51	2	4	0.04 ± 0.00
poly1305/crypto_onetimeauth_poly1305_donna	-1	-1	52	1623	0.01 ± 0.08	17	90626	39.19 ± 112.16
salsa20/crypto_core_salsa20	1628	13.58 ± 0.12	0	3513	< 0.01	0		
sha256/SHA256_Transform	-1	-1	102	410	0.01 ± 0.03	29	100360	9.32 ± 11.75
sha256/crypto_hash_sha256	-1	-1	110	541	0.01 ± 0.03	67	110077	14.73 ± 15.69
sha512/crypto_hash_sha512	-1	-1	6	270	0.01 ± 0.02	86	109908	0.99 ± 0.41
Libsodium -O3								
aead/crypto_aead_chacha20poly1305_encrypt	-1	*	*	376	0.04 ± 0.45	48	330717	6.63 ± 28.03
auth/crypto_auth_hmacsha256	-1	-1	3	669	0.03 ± 0.75	52	107103	1.49 ± 1.77
chacha20/crypto_stream_chacha20	956	0.31 ± 0.01	0	253	< 0.01	2	4	0.05 ± 0.00
poly1305/crypto_onetimeauth_poly1305_donna	-1	-1	6	326	0.03 ± 0.17	87	59566	16.92 ± 36.65
salsa20/crypto_core_salsa20	483	15.87 ± 0.06	0	106	< 0.01	66	291	0.23 ± 0.46
sha256/SHA256_Transform	2171	0.01 ± 0.00	0	479	< 0.01	0		
sha256/crypto_hash_sha256	-1	-1	0	632	0.04 ± 0.78	34	34559	0.27 ± 0.84
sha512/crypto_hash_sha512	-1	-1	4	68	0.01 ± 0.04	20	90629	0.62 ± 0.41
Almeida -O0								
naive_select/ct_select_u32_naive								
select_v1/ct_select_u32_v1								
select_v2/ct_select_u32_v2								
select_v3/ct_select_u32_v3					No loops			
select_v4/ct_select_u32_v4								
sort/sort3								
sort_multiplex/sort3_multiplex								
sort_negative/sort3_negative								
Almeida -O3								
naive_select/ct_select_u32_naive								
select_v1/ct_select_u32_v1								
select_v2/ct_select_u32_v2								
select_v3/ct_select_u32_v3					No loops			
select_v4/ct_select_u32_v4								
sort/sort3								
sort_multiplex/sort3_multiplex								
sort_negative/sort3_negative								
lucky13 -O0								
tls1_cbc_remove_padding_lucky13/tls1_..._lucky13	575	9.83 ± 0.04	5	539	< 0.01	217	701	0.03 ± 0.02
lucky13 -O3								
tls1_cbc_remove_padding_lucky13/tls1_..._lucky13	-1	*	*	94	< 0.01	63	472	0.03 ± 0.03

TABLE IV

Evaluation results with Vivienne$_{\mathrm{INV}}$