# Cryptography with Formal Guarantees

Using HAX



### Outline

- Formal what and why?
- How?
- The hax toolchain
- Hands-on

### hax demo



### tutorial

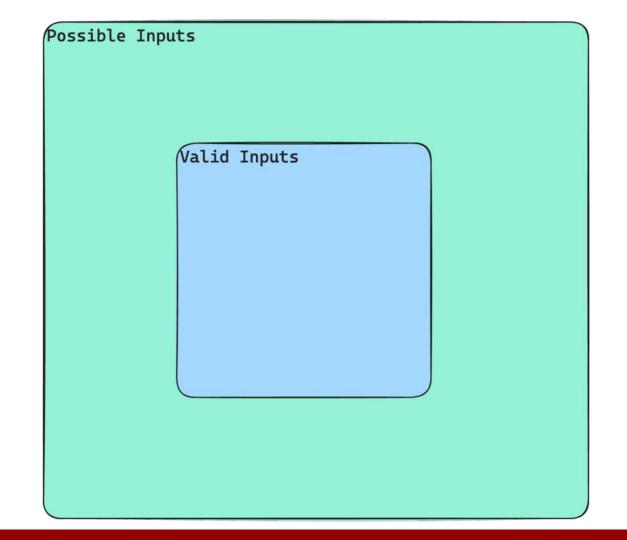


https://github.com/hacspec/hax/blob/franziskus/toronto-2024/examples/README.md

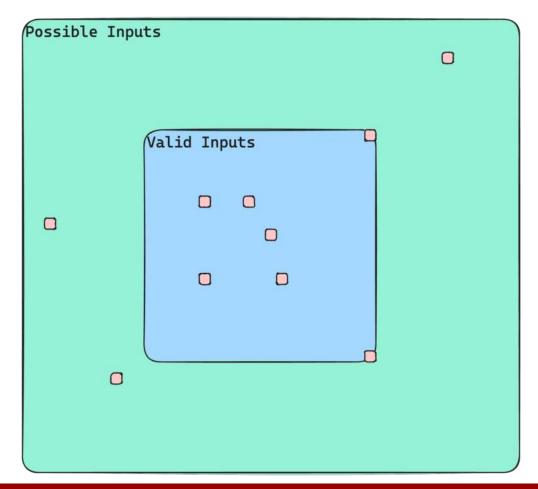
### Why Formal Verification?

### [...] testing is a necessary but insufficient step in the development process to fully reduce vulnerabilities at scale [...]

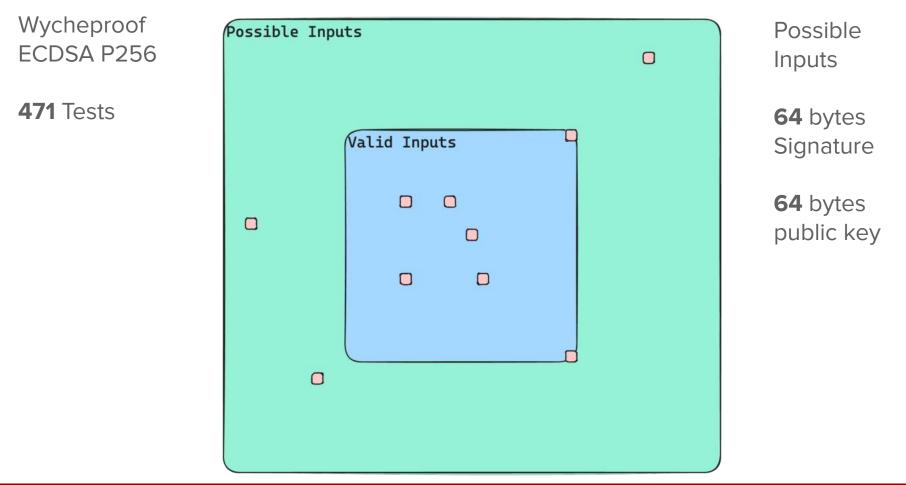
THE WHITE HOUSE WASHINGTON



#### Testing



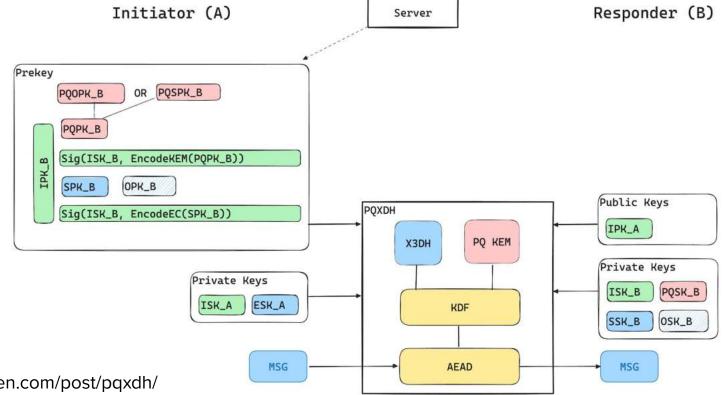
#### Testing



#### Verification

Possible Inp	uts
	Valid Inputs

### PQXDH



https://cryspen.com/post/pqxdh/

### TLS 1.3



#### let CompromiseServer() =

in(c, server\_name: bertie\_\_tls13utils\_\_t\_Bytes);
get server\_dbs(=server\_name, db) in
event CompromisedServer(server\_name);
out(c, db).

process

!CreateServer() | !Client() | !Server() | !CompromiseServer()

### Formal Methods for Performance

#### • • •

// Due to the small coefficient bound, we can skip the first round of
// Montgomery reductions.

```
let mut zeta_i = 1;
```

```
for j in 0..128 {
    // Multiply by the appropriate zeta in the normal domain.
    let t = re.coefficients[j + 128] * -1600;
```

```
re.coefficients[j + 128] = re.coefficients[j] - t;
re.coefficients[j] = re.coefficients[j] + t;
```

```
re = ntt_at_layer_3(&mut zeta_i, re, 6);
re = ntt_at_layer_3(&mut zeta_i, re, 5);
re = ntt_at_layer_3(&mut zeta_i, re, 4);
re = ntt_at_layer_3(&mut zeta_i, re, 3);
re = ntt_at_layer_3(&mut zeta_i, re, 2);
re = ntt_at_layer_3(&mut zeta_i, re, 1);
```

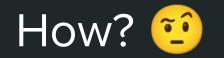
### Formal Methods for Performance

### •••

```
...
re = invert_ntt_at_layer(&mut zeta_i, re, 4);
re = invert_ntt_at_layer(&mut zeta_i, re, 5);
re = invert_ntt_at_layer(&mut zeta_i, re, 6);
re = invert_ntt_at_layer(&mut zeta_i, re, 7);
```

```
// Only reduce the first two entries
for i in 0..2 {
    re.coefficients[i] = barrett_reduce(re.coefficients[i]);
}
```

Verify when Review & Exhaustive Testing is impossible Verify to optimize with confidence



### "[...] correctness is defined as the ability of a piece of software to meet a specific [...] requirement"

THE WHITE HOUSE WASHINGTON

#### **Algorithm 8** NTT(f)

Computes the NTT representation  $\hat{f}$  of the given polynomial  $f \in R_a$ . **Input**: array  $f \in \mathbb{Z}_{q}^{256}$ . ▷ the coefficients of the input polynomial **Output**: array  $\hat{f} \in \mathbb{Z}_{q}^{256}$ . ▷ the coefficients of the NTT of the input polynomial 1:  $\hat{f} \leftarrow f$ ▷ will compute NTT in-place on a copy of input array 2:  $k \leftarrow 1$ 3: for  $(len \leftarrow 128; len \ge 2; len \leftarrow len/2)$ for (start  $\leftarrow 0$ ; start < 256; start  $\leftarrow$  start  $+ 2 \cdot len$ ) 4:  $zeta \leftarrow \zeta^{\mathsf{Bit}\mathsf{Rev}_7(k)} \mod q$ 5:  $k \leftarrow k+1$ 6: for  $(j \leftarrow start; j < start + len; j + +)$ 7:  $t \leftarrow zeta \cdot \hat{f}[j + len]$  $\triangleright$  steps 8-10 done modulo q 8:  $\hat{f}[j+len] \leftarrow \hat{f}[j]-t$ 9:  $\hat{f}[j] \leftarrow \hat{f}[j] + t$ 10: end for 11: end for 12: 13: end for 14: return  $\hat{f}$ 

#### **Algorithm 8** NTT(f)

Computes the NTT representation $\hat{f}$ of	f the given polynomial $f \in R_q$ .
<b>Input</b> : array $f \in \mathbb{Z}_q^{256}$ .	▷ the coefficients of the input polynomial
<b>Output</b> : array $\hat{f} \in \mathbb{Z}_q^{256}$ .	▷ the coefficients of the NTT of the input polynomial
1: $\hat{f} \leftarrow f$	▷ will compute NTT in-place on a copy of input array
2: $k \leftarrow 1$	
3: for (len $\leftarrow$ 128; len $\ge$ 2; len $\leftarrow$ len	n/2)
4: <b>for</b> ( <i>start</i> $\leftarrow$ 0; <i>start</i> < 256; <i>sta</i>	$art \leftarrow start + 2 \cdot len)$
5: $zeta \leftarrow \zeta^{BitRev_7(k)} \mod q$	
6: $k \leftarrow k+1$	
7: <b>for</b> $(j \leftarrow start; j < start +$	<i>len</i> ; <i>j</i> ++) for round in 0(128 >> layer) {
8: $t \leftarrow zeta \cdot \hat{f}[j + len]$	*zeta_i += 1;
9: $\hat{f}[j+len] \leftarrow \hat{f}[j]-t$	
10: $\hat{f}[j] \leftarrow \hat{f}[j] + t$	<pre>let offset = round * step * 2;</pre>
11: end for	
12: end for	<pre>for j in offsetoffset + step {</pre>
13: end for	<pre>let t = montgomery_multiply_fe_by_f</pre>
14: return $\hat{f}$	re.coefficients[j + step], ZETAS_TIMES_MONTGOMERY_R[*zeta_
	);

}

}

```
re.coefficients[j + step] = re.coefficients[j] - t;
re.coefficients[j] = re.coefficients[j] + t;
```

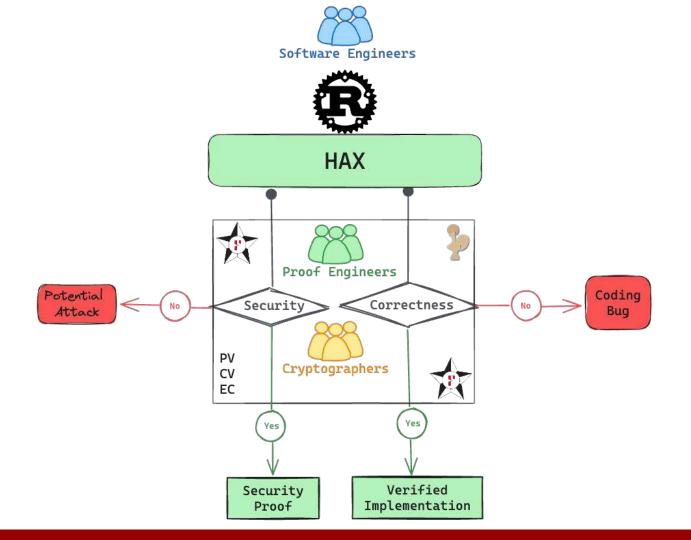


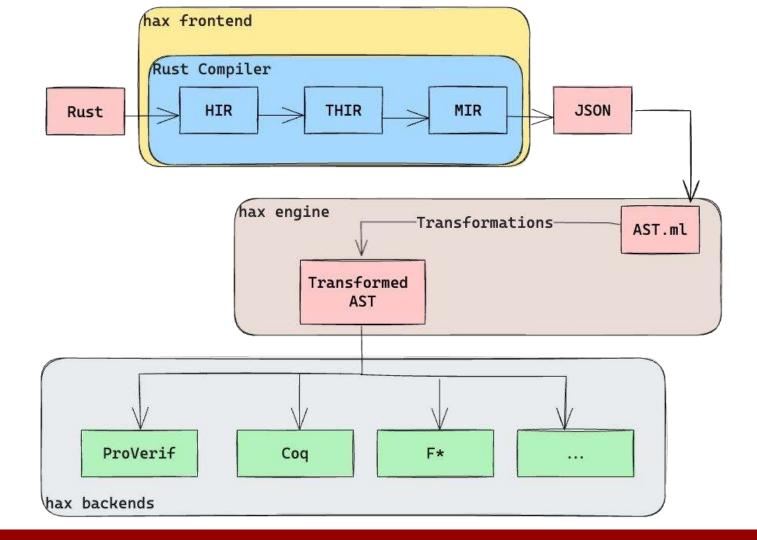


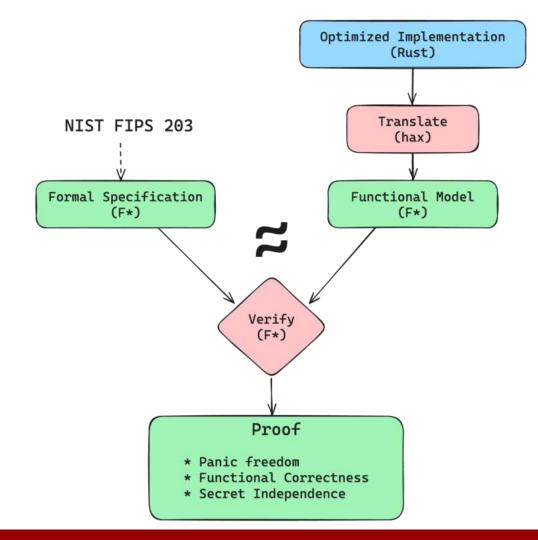
### hax

### High Assurance Translations









### hax: A Tool Framework for Rust Verification

- Accepts a largish (and expanding) subset of safe Rust
   Including hacspec, a purely functional spec language in Rust
- Translates it to formal models in F\* or Coq
   Opcoming backends for EasyCrypt, ProVerif, Lean, ...
- Usable and pragmatic design choices, not dogmatic
- Verify **panic-freedom**, **correctness**, **security**,... for the Rust code *you* care about, using the tool of *your choice*.

### hax: ongoing projects

- **Rust Core:** an annotated version of the Rust Core library
- **Backends:** new backends for Lean, EasyCrypt, ProVerif
- Verified
  - PQ Crypto: verified Rust code for Kyber/ML-KEM, ...
  - **OS Modules:** verified kernel code for RIOT-OS
  - **Protocols:** verified code for EDHOC, MLS, TLS 1.3, ...
  - **Contracts:** verified canisters for Internet Computer

### hax: Ongoing Challenges & Future Work

- Handle all the Rust code we want to handle
- Improve push-button verification
- Annotations in the proof assistant
- Proof assistant error messages
- IDE integration
- Correctness arguments of translations (research)

### Can I use hax?

YES

### Will there be bugs 🐞?

YES



(aedd173a2 2024-03-17)

Click or press 'S' to search, '?' for

### Crate core 🖻

### All Items

### □ The Rust Core Library

### hax: Ongoing Challenges & Future Work

- Handle all the Rust code we want to handle
- Improve push-button verification
- Annotations in the proof assistant
- Proof assistant error messages
- IDE integration
- Correctness arguments of translations (research)

```
// reduce_once reduces 0 \leq x < 2 \times Prime, mod kPrime.
                                    static uint16_t reduce_once(uint16_t x) {
                                      assert(x < 2 * kPrime);
                                      const uint16 t subtracted = x - kPrime;
                                      uint16 t mask = 0u - (subtracted \gg 15);
                                      // On Aarch64, omitting a |value_barrier_u16| results in a 2x speedup of Kyber
                                      // overall and Clang still produces constant-time code using `csel`. On other
                                      // platforms & compilers on godbolt that we care about, this code also
                                      // produces constant-time output.
                                      return (mask & x) | (~mask & subtracted);
// constant time reduce x mod kPrime using Barrett reduction. x must be less
// than kPrime + 2×kPrime<sup>2</sup>.
static uint16 t reduce(uint32 t x) {
  assert(x < kPrime + 2u * kPrime * kPrime);</pre>
  uint64_t product = (uint64_t)x * kBarrettMultiplier;
  uint32_t quotient = (uint32_t)(product >> kBarrettShift);
```

```
uint32_t remainder = x - quotient * kPrime;
```

```
return reduce_once(remainder);
```

```
// reduce_once reduces 0 ≤ x < 2*kPrime, mod kPrime.
                                     static uint16_t reduce_once(uint16_t x) {
                                      assert(x < 2 * kPrime);</pre>
                                      const uint16 t subtracted = x - kPrime;
                                      uint16 t mask = 0u - (subtracted \gg 15);
                                      // On Aarch64, omitting a |value_barrier_u16| results in a 2x speedup of Kyber
                                       // overall and Clang still produces constant-time code using `csel`. On other
                                       // platforms & compilers on godbolt that we care about, this code also
                                      // produces constant-time output.
                                      return (mask & x) | (~mask & subtracted);
// constant time reduce x mod kPrime using Barrett reduction. x must be less
// than kPrime + 2 \times kPrime^2.
static uint16_t reduce(uint32_t x) {
  assert(x < kPrime + 2u * kPrime * kPrime);</pre>
```

```
uint64_t product = (uint64_t)x * kBarrettMultiplier;
uint32_t quotient = (uint32_t)(product >> kBarrettShift);
uint32_t remainder = x - quotient * kPrime;
return reduce_once(remainder);
```

- 1. Asserts
- 2. Make the requirements formal
- 3. hax attributes for "design by contract"
- 4. F\* statically checks that the properties hold

```
#[requires(coefficient_bits ≤ 11 & i32::from(fe) ≤ FIELD_MODULUS)]
#[ensures(|result| result ≥ 0 & result ≤ (1 < coefficient_bits) - 1)]
pub(super) fn compress_q(coefficient_bits: usize, fe: u16) → KyberFieldElement {
    let mut compressed: u32 = (fe as u32) << (coefficient_bits + 1);
    compressed += FIELD_MODULUS as u32;
    compressed ≠= (FIELD_MODULUS << 1) as u32;
    (compressed & ((1u32 << coefficient_bits) - 1)) as KyberFieldElement</pre>
```

- 1. Asserts
- 2. Make the requirements formal
- 3. hax attributes for "design by contract"
- 4. F\* statically checks that the properties hold

## Verifying Libcrux's ML-KEM

Karthikeyan Bhargavan, Lucas Franceschino, Franziskus Kiefer, Goutam Tamvada

January 30, 2024



https://cryspen.com/post/ml-kem-verification/

### Ensure specification is correct

- Specification is ground truth
- Manual inspection
  - Spec must be succinct & easy to read
- Tests
  - Test vectors
  - Test against other implementations

## What do we prove?

- Panic freedom
- Secret independence
- Correctness (spec equivalence)
- Other properties when desirable
  - Examples

deserialize(serialize(x)) == x

decrypt(encrypt(y)) == y

## Demo & Hands-on

### Tasks

- Extract F\*
- Lax typecheck F\*
- Typecheck F\* (panic freedom)
- Modify pre-conditions in Rust (and try to typecheck again)

### Examples

- Chacha20
- Barrett reduction
- Compression in ML-KEM
- Write your own code!

Get together in small groups

Start docker and play with the examples

~/hax/examples

EXPLORER ····	📢 Welcome 🛛 🔞 lib.rs 🛛 🗙	<del></del>	> > (b) [[]		$\searrow$ zsh - chacha20 + $\checkmark$ 🗎 🛍 … $\times$
$\sim$ open editors	hax > examples > chacha20 > src > @ lib.rs > {} hacspec_	helper		~/repos/hax/examples/chacha20 / main ±	
刘 Welcome	You, 19 hours ago   2 authors (Lucas Franceschino and othe		W BURNE		
× 🖲 lib.rs examples/chacha20/			W BRANC		
$\sim$ UNTITLED (WORKSPACE)	<pre>2</pre>		TIMERINE		
$\sim$ hax	<pre>5 4 use hax_lib_macros as hax;</pre>		LELEVER WILLIAM		
> .cache	5				
> .docker	- 6 type State = [u32; 16];				
> .github	7 type Block = [u8; 64];				
> .hints	<pre>8 type ChaChaIV = [u8; 12];</pre>		Langester and the		
> .utils	9 type ChaChaKey = [u8; 32];		Live Safe		
> cli			THEFT		
> engine	11 #[hax::requires(a < 16 & b < 16 & d <		A DESCRIPTION OF A DESC		
✓ examples	12 fn chacha20_line(a: usize, b: usize, d:	usize, s: u32, m: State) → State {			
$\sim$ chacha20	13 let mut state = m; 14 state[a] = state[a].wrapping add(st				
✓ proofs	<pre>14 state[a] = state[a].wrapping_add(st 15 state[d] = state[d] ^ state[a];</pre>	ace[b]);			
> coq/ <u>extraction</u>	<pre>16 state[d] = state[d].rotate_left(s);</pre>				
✓ fstar/extraction	17 state				
hax.fst.config.json	18 }				
M Makefile					
✓ src	20 #[hax::requires(a < 16 & b < 16 & c <	16 <del>86</del> d < 16)]			
hacspec_helper.rs		b: usize, c: usize, d: usize, state: State) → Stat			
® lib.rs	<pre>22 let state = chacha20_line(a, b, d,</pre>				
> tests	23 let state = chacha20_line(c, d, b,				
Cargo.toml	<pre>24 let state = chacha20_line(a, b, d, 25 chacha20_line(c, d, b, 7, state)</pre>	8, state);			
M Makefile	26 }				
> limited-order-book					
> sha256	<pre>28 fn chacha20_double_round(state: State)</pre>	→ State {			
> target	<pre>29 let state = chacha20_quarter_round(</pre>	0, 4, 8, 12, state);			
≣ Cargo.lock	<pre>30 let state = chacha20_quarter_round(</pre>	1, 5, 9, 13, state);			
Cargo.toml	<pre>31 let state = chacha20_quarter_round(</pre>				
≣ default.nix	<pre>32 let state = chacha20_quarter_round(</pre>	3, 7, 11, 15, state);			
{} hax.fst.config.json	33				
M Makefile	34 let state = chacha20_quarter_round( 35 let state = chacha20 quarter round(				
③ README.md	35 let state = chacha20_quarter_round( 36 let state = chacha20_quarter_round(				
> frontend	37 chacha20_quarter_round(3, 4, 9, 14,				
> hax-lib	38 }				
> hax-lib-macros					
> proof-libs	40 pub fn chacha20_rounds(state: State) $\rightarrow$				
> target	41 let mut st = state;				
> test-harness	42 for _i in 010 {				
> tests	<pre>43 st = chacha20_double_round(st);</pre>				
> OUTLINE					
> TIMELINE	45 st				
> RUST DEPENDENCIES	46 <b>}</b> 47				
> TAG TREE					

		$\leftrightarrow$ $ ho$ Untitle	d (Workspace)		
EXPLORER ····	📢 Welcome 🛛 🔞 lib.rs 🛛 🗙		<> ○ ○ ③ ⑤ □ …		$\searrow$ zsh - chacha20 + $\vee$ $\square$ 🋍 $\cdots$ $\times$
$\sim$ open editors	hax > examples > chacha20 > src >	helper			
🗙 Welcome	You, 19 hours ago   2 authors (Lucas Franceschino and ot			/repos/hax/examples/chacha Compiling chacha20 v0.1.0	(/Users/franziskus/repos/hax/examples/chacha20)
×	1 mod hacspec_helper; Lucas Frances		: init w	Finished test [unoptimiz	ed + debuginfo] target(s) in 0.80s
<ul> <li>UNTITLED (WORKSPACE)</li> </ul>	2			Running unittests src/1 d2a88bf9d55)	<pre>ib.rs (/Users/franziskus/repos/hax/examples/target/debug/deps/chacha20-6312c</pre>
<ul> <li>v bitmeeb (workspace)</li> <li>v bax</li> </ul>				0280019033)	
<ul> <li>nax</li> <li>.cache</li> </ul>	4 use hax_lib_macros as hax;			running 0 tests	
			n and a state of the state of t	test result: ok 0 passed: 0	failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s
> .docker	<pre>6 type State = [u32; 16];</pre>				
> .github	<pre>7 type Block = [u8; 64];</pre>			Running tests/kat.rs (/	Users/franziskus/repos/hax/examples/target/debug/deps/kat-da1c246bb0221d83)
> .hints	<pre>8 type ChaChaIV = [u8; 12];</pre>		Transaction of the second	running 1 test	
> .utils	9 type ChaChaKey = [u8; 32];		The second s	test test_kat ok	
> cli					
> engine	11 #[hax::requires(a < 16 & b < 16 & d				
✓ examples	12 fn chacha20_line(a: usize, b: usize, d	: usize, s: usz, m: State) → State {	1202269.0	00500	hav into fator
✓ chacha20	<pre>13 let mut state = m; 14 state[a] = state[a].wrapping_add(s</pre>	tate[b])		caruo	hax into fstar
✓ proofs	15 state[d] = state[d] ^ state[a];	cace[0]);		3	
> coq	16 state[d] = state[d] state[a];				
✓ fstar/extraction	17 state	,		~/repos/hax/examples/chacha	20 > 🖉 main ± ) cargo hax into fstar
≣ Chacha20.fst M	18			Compiling proc-macro2 v1.	
≣ Chacha20.Hacspe M				Compiling unicode-ident v Compiling syn v1.0.109	1.0.12
{} hax.fst.config.json	20 #[hax::requires(a < 16 & b < 16 & c	< 16.6 <del>6</del> d < 16)]		Compiling version_check v	0.9.4
M Makefile		, b: usize, c: usize, d: usize, state: State) -	→ State	Compiling libc v0.2.150	
	<pre>22 let state = chacha20_line(a, b, d,</pre>			Compiling serde v1.0.192 Compiling serde_json v1.0	.108
	<pre>23 let state = chacha20_line(c, d, b,</pre>			Compiling ryu v1.0.15	
In the second	24 let state = chacha20_line(a, b, d,			Compiling itoa v1.0.9 Compiling schemars v0.8.1	6
Ib.rs	25 chacha20_line(c, d, b, 7, state)			Compiling cfg-if v1.0.0	
> tests				Compiling dyn-clone v1.0.	
Cargo.toml				Compiling proc-macro-erro	pre.1 (/Users/franziskus/repos/hax/hax-lib) r-attr v1.0.4
M Makefile	28 fn chacha20_double_round(state: State)	→ State {		Compiling proc-macro-erro	
> limited-order-book	<pre>29 let state = chacha20_quarter_round</pre>	(0, 4, 8, 12, state);		Compiling quote v1.0.33 Compiling syn v2.0.39	
> sha256	<pre>30 let state = chacha20_quarter_round</pre>	(1, 5, 9, 13, state);		Compiling getrandom v0.2.	11
> target	<pre>31 let state = chacha20_quarter_round</pre>			Compiling uuid v1.5.0	transla
≣ Cargo.lock	<pre>32 let state = chacha20_quarter_round</pre>	(3, 7, 11, 15, state);		Compiling serde_derive_ir Compiling schemars_derive	v0.8.16
Cargo.toml				Compiling serde_derive v1	.0.192
≣ default.nix	34 let state = chacha20_quarter_round				<pre>types v0.1.0-pre.1 (/Users/franziskus/repos/hax/hax-lib-macros/types) v0.1.0-pre.1 (/Users/franziskus/repos/hax/hax-lib-macros)</pre>
hax.fst.config.json	35 let state = chacha20_quarter_round			Compiling chacha20 v0.1.0	(/Users/franziskus/repos/hax/examples/chacha20)
M Makefile	36 let state = chacha20_quarter_round 37 chacha20 guarter round(3, 4, 9, 14			note: Writing file "/Users/1	ranziskus/repos/hax/examples/chacha20/proofs/fstar/extraction/Chacha20.fst"
① README.md		, state)		note: Writing file "/Users/f	ranziskus/repos/hax/examples/chacha20/proofs/fstar/extraction/Chacha20.Hacsp
> frontend	38 <b>}</b> 39			ec_helper.fst"	
> hax-lib	40 pub fn chacha20_rounds(state: State) -	A State /		Finished day [upontimize	d + debuginfo] target(s) in 7.79s
> hax-lib-macros	40 pub in chachazo_rounds(state: state) - 41 let mut st = state;			<ul> <li>~/repos/hax/examples/chacha</li> </ul>	
> proof-libs	42 for i in 010 {				
	43 st = chacha20 double round(st)				
> target					
	45 st				
> TIMELINE	46				
> RUST DEPENDENCIES					

$ \frac{1}{2} = 1$		$\leftarrow$ $\rightarrow$ $ ho$ Untitled	d (Workspace)		
<pre>ibr carefull is the intervention of a second s</pre>	EXPLORER	⊛ lib.rs ×		≣ Chacha20.fst M ×	ჯ. 🗢 ം 💿 🖽 …
bax-lib-macros     43     st = chacha20_double_round(st);     43     let state:t_Array u32 (sz 16) = chacha20_line c d b 12ul state in       > OUTLINE     44     }     let state:t_Array u32 (sz 16) = chacha20_line a b d 8ul state in       > TIMELINE     45     st     45       > RUST DEPENDENCIES     46     46	<pre>&gt; OPEN EDITORS GROUP 1</pre>	<pre>® lib.s × hax &gt; examples &gt; chacha20 &gt; src &gt; @ lib.rs &gt; () hacspec_helper You, 19 hours app (2 subhars (Locues Franceschino and others) 1 mod hacspec_helper::*; 3 4 use hax_lib_macros as hax; 5 5 type State = [u32; 16]; 7 type ChachaTV = [u8; 12]; 9 type ChachaTV = [u8; 12]; 11 #fhax::requires(a &lt; 16 66 b &lt; 16 66 d &lt; 16)] 12 fn chacha20_line(a: usize, b: usize, s: u32, m: State) → State { 13 let mut state = m; 14 state[a] = state[d].vrate_left(s); 15 state[d] = state[d].rotate_left(s); 16 state[d] = state[d].rotate_left(s); 17 state 18 } 19 19 fn chacha20_quarter_round(a: usize, b: usize, c: usize, d: usize, state: State) 11 let state = chacha20_line(a, b, d, 16, state); 12 let state = chacha20_line(a, b, d, 16, state); 13 let state = chacha20_line(a, d, b, 3, state); 14 let state = chacha20_line(a, d, b, 3, state); 15 let state = chacha20_line(a, d, b, 3, state); 16 let state = chacha20_quarter_round(0, 4, 8, 12, state); 17 let state = chacha20_quarter_round(0, 7, 11, 15, state); 18 let state = chacha20_quarter_round(2, 6, 10, 14, state); 19 let state = chacha20_quarter_round(2, 7, 11, 15, state); 19 let state = chacha20_quarter_round(2, 7, 11, 15, state); 19 let state = chacha20_quarter_round(2, 7, 11, 15, state); 19 let state = chacha20_quarter_round(2, 7, 11, 15, state); 19 let state = chacha20_quarter_round(2, 7, 11, 15, state); 19 let state = chacha20_quarter_round(2, 7, 11, 15, state); 19 let state = chacha20_quarter_round(2, 7, 11, 15, state); 19 let state = chacha20_quarter_round(2, 7, 11, 15, state); 19 let state = chacha20_quarter_round(2, 7, 18, 13, state); 19 let state = chacha20_quarter_round(2, 7, 14, 13, state); 19 let state = chacha20_quarter_round(2, 7, 18, 13, state); 10 let state = chacha20_quarter_round(2, 7, 18, 13, state); 10 let state = chacha20_quarter_round(2, 7, 18, 13, state); 10 let state = chacha20_quarter_round(2, 7, 18, 13, state); 10 let state = chacha20_quarter_round(2, 7, 18, 13, state); 10 let state = chacha20_quarter_round(2, 7, 18, 13, state); 10 let state = chacha20_q</pre>		<pre>hax &gt; examples &gt; chacha20 &gt; proofs &gt; fstar &gt; extract You, 11 seconds ago   2 authors (Lucas Franceschin module Chacha20 Lucas Franceschin g #set-options "fuel 0ifuel 1 3 open Core 4 open FStar.Mul 5 unfold 10 let t_Block = t_Array u8 (sz 64) 8 unfold 10 let t_ChaChaIV = t_Array u8 (sz 12 11 12 unfold 13 let t_ChaChaKey = t_Array u8 (sz 12 11 14 unfold 15 unfold 16 let t_State = t_Array u32 (sz 16) 17 18 let chacha20_line (a b d: usize) ( 19 : Prims.Pure (t_Array u32 (sz 16) 17 18 let chacha20_line (a b d: usize) ( 19 : Prims.Pure (t_Array u32 (sz 16) = 20   (requires a &lt;. sz 16 6fb &lt;. 21   (Core.Num.impl_u32_wrappin 22 let state:t_Array u32 (sz 16) = 23 let state:t_Array u32 (sz 16) = 24 Rust_primitives.Hax.update_at 30 in 31 let state:t_Array u32 (sz 16) = 32 Rust_primitives.Hax.update_at 33 in 34 let state: 34   (Core.Num.impl_u32_rotate_ 35 in 35 let chacha20_quarter_round (a b c 35 : Prims.Pure (t_Array u32 (sz 40   : Prims.Pure (</pre>	<pre>%1</pre>
> TAG TREE $(0 - c)$ and for checks 20 core(cfr: u22 cfd: State) $\rightarrow$ State $\int (0 - c)$	> hax-lib > hax-lib-macros > OUTLINE > TIMELINE > RUST DEPENDENCIES	<pre>42</pre>		42         let state:t_Array u32 (sz 16) =           43         let state:t_Array u32 (sz 16) =           44         let state:t_Array u32 (sz 16) =           45         chacha20_line c d b 7ul state           46         let chacha20_double_round (state:	chacha20_line c d b 12ul state in chacha20_line a b d 8ul state in t_Array u32 (sz 16)) : t_Array u32 (sz 16) =

	$\leftarrow$ $\rightarrow$ $\bigcirc$ Untit	tled (Workspace)		
EXPLORER ···	® lib.rs $ imes$		≣ Chacha20.fst M ×	ሮ 🗠 🖓 🗓
<ul> <li>&gt; OPEN EDITORS GROUP 1</li> <li>Ib.rs examples/chacha20/ cROUP 2</li> <li>&gt; E Chacha20.fst exam M</li> <li>&gt; UNTITLED (WORKSPACE)</li> <li>&gt; honts</li> <li>&gt; .docker</li> <li>&gt; .docker</li> <li>&gt; .dithub</li> <li>&gt; .hints</li> <li>&gt; .utils</li> <li>&gt; cli</li> <li>&gt; engine</li> </ul>	<pre>     W lib.rs ×     hax &gt; examples &gt; chacha20 &gt; src &gt;          W lib.rs &gt; {} hacspec_helper         You, 19 hours ago 12 authors (Lucas Franceschino and others)         mod hacspec_helper;:         use hacspec_helper::+;         use hax_lib_macros as hax;         f         type State = [u32; 16];         type Block = [u8; 64];         type ChaChaIV = [u8; 12];         type ChaChaKey = [u8; 32];         #         #[hax::requires(a &lt; 16 &amp; b &lt; 16 &amp; 6 d &lt; 16)]         tn chacha20_Line(a: usize, b: usize, d: usize, s: u32, m: State) → State {             let mut state = m;             state[a] = state[a] = state[a].wrapping_add(state[b]);         }         }         }</pre>		hax > examples > chacha20 > proofs > fstar > extraction > ≣ Chacha2 You, 11 seconds ago   2 authors (Lucas Franceschino and others)	Ofst ago via PR #295 • add Coq and F* sr
<pre>     vexamples     vexacha20     vproofs     coq     fstar/extraction     E Chacha20.fst     M     E Chacha20.fst     M </pre>	<pre>14 State[a] = State[a].wrapping_add(state[b]); 15 state[d] = state[d] ^ state[a]; 16 state[d] = state[d].rotate_left(s); 17 state 18 } 19 20 #[hax::requires(a &lt; 16 &amp;&amp; b &lt; 16 &amp;c &lt; 16 &amp; d &lt; 16)] 21 pub fn chacha20_quarter_round(a: usize, b: usize, c: usize, d: usize, state: State)</pre>		14 15 unfold 16 let t_State = t_Array u32 (sz 16) 17 18 let chacha20_line (a b d: usize) (s: u32) (m: t_ 19 : Prims.Pure (f Array u32 (sz 16)) 20 (requires a <. sz 16 & b d <. sz 16 & b d <. 21 (fun _ → Prims.L_Irue) =	
<ul> <li>() hax.fst.config.json</li> <li>M Makefile</li> <li>✓ src</li> <li>⊗ hacspec_helper.rs</li> <li>⊗ lib.rs</li> </ul>	<pre>22</pre>		22       let state:t_Array u32 (sz 16) = m in         23       let state:t_Array u32 (sz 16) =         24       Rust_primitives.Hax.update_at state         25       a         26       (Core.Num.impl_u32_wrapping_add (state.[         27       in	a]<:u32)(state.[b]<:u32)<:
<ul> <li>&gt; tests</li> <li>Cargo.toml</li> <li>M Makefile</li> <li>&gt; limited-order-book</li> <li>&gt; sha256</li> <li>&gt; target</li> <li>≅ Cargo.lock</li> </ul>	<pre>28 fn chacha20_double_round(state: State) → State { 29     let state = chacha20_quarter_round(0, 4, 8, 12, state); 30     let state = chacha20_quarter_round(1, 5, 9, 13, state); 31     let state = chacha20_quarter_round(2, 6, 10, 14, state); 32     let state = chacha20_quarter_round(3, 7, 11, 15, state); 33 34     let state = chacha20_quarter_round(0, 5, 10, 15, state); </pre>		28       let state:t_Array u32 (sz 16) =         29       Rust_primitives.Hax.update_at state d ((stat 30         31       let state:t_Array u32 (sz 16) =         32       Rust_primitives.Hax.update_at state         33       d         34       (Core.Num.impl_u32_rotate_left (state.[	
<ul> <li>Cargo.toml</li> <li>default.nix</li> <li>hax.fst.config.json</li> <li>Makefile</li> <li>README.md</li> </ul>	<pre>35 let state = chacha20_quarter_round(1, 6, 11, 12, state); 36 let state = chacha20_quarter_round(2, 7, 8, 13, state); 37 chacha20_quarter_round(3, 4, 9, 14, state) 38 } 39 pub fn chacha20_rounds(state: State) → State { 40 let mut st = state; 41 let mut st = state; 42 let mut st = state; 43 let mut st = state; 44 let mut st = state; 45 let mut st = state; 46 let mut st = state; 47 let mut st = state; 48 let mut st = state; 48 let mut st = state; 48 let mut st = state; 49 let mut st = state; 40 let mut st = state; 40 let mut st = state; 40 let mut st = state; 41 let mut st = state; 42 let mut st = state; 43 let mut st = state; 44 let mut st = state; 44 let mut st = state; 45 let mut st = state; 46 let mut st = state; 47 let mut st = state; 48 let mut st = state; 48</pre>		35       in         36       state         37	
> frontend > hax-lib > hax-lib-macros > OUTLINE > TIMELINE > RUST DEPENDENCIES	<pre>41</pre>		<ul> <li>41 (Tun _ → Prims.L_irue) =</li> <li>42 let state:t_Array u32 (sz 16) = chacha20_line</li> <li>43 let state:t_Array u32 (sz 16) = chacha20_line</li> <li>44 let state:t_Array u32 (sz 16) = chacha20_line</li> <li>45 chacha20_line c d b 7ul state</li> <li>46</li> <li>47 let chacha20_double_round (state: t_Array u32 (s</li> </ul>	c d b 12ul state in a b d 8ul state in
> TAG TREE	10 muh fa chachaga caralata: 1022 eta: Ctata) - Ctata J		47 Let chachazo_uouble_round (state. t_Array usz (s	

¢ Lucas Franceschino, 4 weeks ago via PR #295 Ln 1, Col 1 Spaces: 2 UTF-8 LF F\* இ Spell ⊘ Prettier ...

	$\leftarrow$ $ ightarrow$ [ $ ho$ Unti	ed (Workspace)	
EXPLORER	😢 lib.rs 🛛 🗙	··· ≡ Chacha20.fst M ×	뱣 🗢 🗠 🕥 🖽 …
$\sim$ OPEN EDITORS	hax > examples > chacha20 > src > <a> lib.rs &gt; {} hacspec_helper</a>	hax > examples > chacha20 > proofs > fstar > extrac	tion > ≣ Chacha20.fst
GROUP 1	You, 19 hours ago   2 authors (Lucas Franceschino and others)	18 let chacha20_line (a b d: usize) (	(s: u32) (m: t_Array u32 (sz 16))
B lib.rs examples/chacha20/	1 mod hacspec_helper;	THE WITCH A CONTRACT OF A CONT	
GROUP 2	2 🔓e hacspec_helper::*;		d: usize) (state: t_Array u32 (sz 16))
× ≣ Chacha20.fst exam M		39         : Prims.Pure (t_Array u32 (sz           1782	
V UNTITLED (WORKSPACE)	4 use hax_lib_macros as hax;	40 (requires a <. sz 10 00 b <. 41 (fun _ → Prims.l_True) =	. sz 16 86 c <. sz 16 86 d <. sz 16)
<ul> <li>v hax</li> </ul>	5	41 (101 _ → Prims.t_1rue) = 42 let state:t Array u32 (sz 16) =	chacha20_line a b d 16ul state in
> .cache	6 type State = [u32; 16];		chacha20_line c d b 12ul state in
	7 type Block = [u8; 64];		chacha20_line a b d 8ul state in
> .docker	8 type ChaChaIV = [u8; 12];	45 chacha20_line c d b 7ul state	
> .github	9 type ChaChaKey = [u8; 32]; 10	The second	Light ( data data part are a set
> .hints	10 #[hax::requires(a < 16 66 b < 16 66 d < 16)]		t_Array u32 (sz 16)) : t_Array u32 (sz 16) =
> .utils	11 mainax::requires(a < 16 do b < 16 do a < 16)] 12 fn chacha20_line(a: usize, b: usize, d: usize, s: u32, m: State) → State {		chacha20_quarter_round (sz 0) (sz 4) (sz 8) (sz 12
> cli	13 let mut state = m;		chacha20_quarter_round (sz 1) (sz 5) (sz 9) (sz 13 tereszteretetetetetetetetetetetetetetetetetet
> engine	14 state[a] = state[a].wrapping_add(state[b]);	🕱 50 let state:t_Array u32 (sz 16) =	chacha20_quarter_round (sz 2) (sz 6) (sz 10) (sz 1
✓ examples	15 state[d] = state[d] ^ state[a];	🕱 51 let state:t_Array u32 (sz 16) =	chacha20_quarter_round (sz 3) (sz 7) (sz 11) (sz 1,
∨ chacha20     ●	16 state[d] = state[d].rotate_left(s);	🕱 52 let state:t_Array u32 (sz 16) =	chacha20_quarter_round (sz 0) (sz 5) (sz 10) (sz 1
✓ proofs	17 state	🕈 53 let state:t_Array u32 (sz 16) =	chacha20_quarter_round (sz 1) (sz 6) (sz 11) (sz 1
> coq	18	🛣 54 let state:t_Array u32 (sz 16) =	chacha20_quarter_round (sz 2) (sz 7) (sz 8) (sz 13
✓ fstar/extraction		🕈 55 chacha20_quarter_round (sz 3) (s	sz 4) (sz 9) (sz 14) state
E Chacha20.fst M	20 #[hax::requires(a < 16 & b < 16 & c < 16 & d < 16)]		
E Chacha20.Hacspe M	21 pub fn chacha20_quarter_round(a: usize, b: usize, c: usize, d: usize, state: State)		ay u32 (sz 16)) : t_Array u32 (sz 16) =
<pre>{} hax.fst.config.json</pre>	<pre>22 let state = chacha20_line(a, b, d, 16, state);</pre>	58 let st:t_Array u32 (sz 16) = sta	ite in
M Makefile	<pre>23 let state = chacha20_line(c, d, b, 12, state);</pre>	59 let st:t_Array u32 (sz 16) =	
✓ src	<pre>24 let state = chacha20_line(a, b, d, 8, state);</pre>		old (Core.Iter.Traits.Collect.f_into_iter ({
6 hacspec_helper.rs	25 chacha20_line(c, d, b, 7, state)	··· 61 Core.Ops.Range.f_sta ··· 62 Core.Ops.Range.f end	
Bib.rs	26 }	<pre> 62 Core.Ops.Range.f_end 63</pre>	1 = 101
> tests		··· 65 //	
	<pre>28 fn chacha20_double_round(state: State) → State {</pre>	65 Core.Ops.Range.t_Range	132)
<ul> <li>Cargo.toml</li> <li>Makefile</li> </ul>	<pre>29 let state = chacha20_quarter_round(0, 4, 8, 12, state);</pre>	66 <:	. 1527
	<pre>30 let state = chacha20_quarter_round(1, 5, 9, 13, state);</pre>	67 Core.Ops.Range.t_Range i32	
> limited-order-book	<pre>31 let state = chacha20_quarter_round(2, 6, 10, 14, state); 32 let state = chacha20 quarter round(3, 7, 11, 15, state);</pre>		
> sha256	<pre>32 let state = chacha20_quarter_round(3, 7, 11, 15, state); 33</pre>	69 (fun st v_i →	
> target	<pre>33 34 let state = chacha20_quarter_round(0, 5, 10, 15, state);</pre>	70 let st:t_Array u32 (sz 1	L6) = st in
≣ Cargo.lock	35 let state = chacha20_quarter_round(1, 6, 11, 12, state);		
Cargo.toml	36 let state = chacha20_quarter_round(2, 7, 8, 13, state);		: <: t_Array u32 (sz 16))
≣ default.nix	37 chacha20_quarter_round(3, 4, 9, 14, state)		
{} hax.fst.config.json	38		
M Makefile			
③ README.md	40 pub fn chacha20_rounds(state: State) → State {		: t_Array u32 (sz 16)) : t_Array u32 (sz 16) =
> frontend	41 let mut st = state;	<pre> 77 let state:t_Array u32 (sz 16) =</pre>	st0 in
> hax-lib	42 for _i in 010 {	<pre> 78 let state:t_Array u32 (sz 16) =</pre>	
> hax-lib-macros	<pre>43 st = chacha20_double_round(st);</pre>	… 79 Rust_primitives.Hax.update_at	state
> OUTLINE		80 (sz 12)	
> TIMELINE	45 <b>st</b>		ng_add (state.[ sz 12 ] <: u32) ctr <: u32)
> RUST DEPENDENCIES			
> TAG TREE		83 let k:t_Array u32 (sz 16) = chac	
	$2 \otimes 0 \otimes 1$ Live Share $\otimes$ rust-analyzer $\oplus$ opam(4.14.1)	84 Chacha20.Hacspec helper.add stat	Ln 213. Col 1 Spaces: 2 UTF-8 LF F* @ Spell ⊘ Prettier □

🄀 🌮 main\* ↔ 🐉 ⊗ 0 🛆 0 🛈 2 👾 0 🔗 Live Share 🛛 ⊗ rust-analyzer 🗇 opam(4.14.1)

Ln 213, Col 1 Spaces: 2 UTF-8 LF F\* @ Spell ⊘ Prettier 🔒

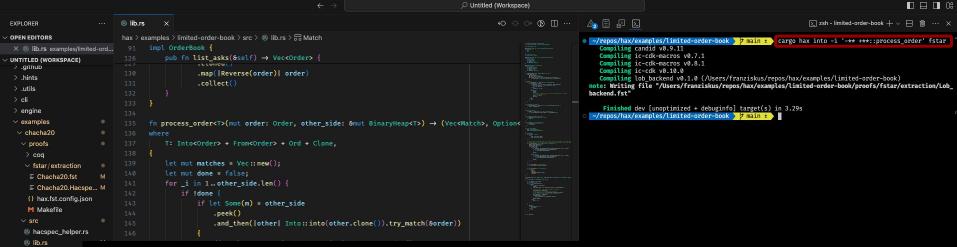
	,⊃ Unti	itled (Workspace)		
(i) lib.rs M ×		≣ Chacha20.fst 3, M ×		ფ⇔⇔⊛Ш…
<pre>hac&gt; examples &gt; chacha20 &gt; src &gt; @ lib.rs &gt; @ chacha20_line You, teacond appl 2 withouts (Lucas Frameworknow and others) 1 mod/ hacspec_helper; 2 use hacspec_helper; 3 use hax_lib_macros as hax; 5 type State = [u32; 16]; 1 type Btack = [u32; 16]; 1 type ChachaVe = [u43; 12]; 1 type ChachaVe = [u43; u32]; 1 f c_hacha20_line(rs: usize, b: usize, s: u32, m: State) → State { 1 tate[d] = state[d] * state[a]; 1 state[d] = state[d] * state[a]; 1 state[d] = state[d] * state[a]; 1 tate = chacha20_line(c, d, b, 12, state); 1 let state = chacha20_line(c, d, b, 12, state); 1 let state = chacha20_line(c, d, b, 12, state); 1 let state = chacha20_line(c, d, b, 13, state); 1 let state = chacha20_line(c, d, b, 13, state); 1 let state = chacha20_line(c, d, b, 13, state); 1 let state = chacha20_line(c, d, b, 14, 13, state); 1 let state = chacha20_quarter_round(f, 5, 10, 11, state); 1 let state = chacha20_quarter_round(f, 5, 10, 15, state); 1 let state = chacha20_quarter_round(f, 5, 11, 12, state); 1 let state = chacha20_quarter_round(f, f, f, 11, 12, state); 1 let state = chacha20_quarter_round(f, f, f, 11, 12, state); 1 let state = chacha20_quarter_round(f, f, f, 11, 12, state); 1 let state = chacha20_quarter_round(f, f, f</pre>		14       state i}; got type usize         15       unfold       - See also /Users/franziskus/repos//         16       let t       libs/fstar/rust_primitives/Rust_primit         17       Rust_primitives.Hax.fst(16, 33): related to the state i}; got type usize         20       - Subtyping check failed; expected to state i}; got type usize         21       - Subtyping check failed; expected to state i}; got type usize         22       unknown because (incomplete qual Note:         23       let         24       Ru         25       a         26       (Core.Num.impl_u32_wrapping_add (state 17)); in         28       let state:t_Array u32 (sz 16) =	<pre>15" type i: usize{Core.Ops.Index.in_range hax/proof- tives.Hax.fst(16,32-16,44) ted location type i: usize{Core.Ops.Index.in_range ntifiers) (fuel=0; ifuel=1) s reached' means the SMT query timed te.[ a ] &lt;: u32) (state.[ b ] &lt;: u32) &lt;: u3 state.[ d ] &lt;: u32) ^. (state.[ a ] &lt;: u32) e.[ d ] &lt;: u32) s &lt;: u32) e.[ d ] &lt;: u32) s &lt;: u32) (state: t_Array u32 (sz 16)) c &lt;. sz 16 &amp; d d &lt;. sz 16) ine a b d 16ul state in ine a b d 8ul state in ine a b d 8ul state in</pre>	
ארא		م ( Avray 22 (حبر 16 - chacha20 مر مربع) م ( م	seconds ago Ln 20, Col 16 Spaces: 2 UTF-8 LF	

```
\mathfrak{U} \leftrightarrow \mathfrak{O} \oplus \mathfrak{O} \square \cdots
🛞 lib.rs M 🛛
hax > examples > chacha20 > src > <a>[b]</a> lib.rs > <a>[c]</a> chacha20_quarter_round
                                                                                                                          hax > examples > chacha20 > proofs > fstar > extraction > E Chacha20.fst
      mod hacspec_helper;
       use hacspec_helper::*;
                                                                                                                                let t Block = t Arrav u8 (sz 64)
       use hax_lib_macros as hax;
                                                                                                                                let t_ChaChaIV = t_Array u8 (sz 12)
      type State = [u32; 16];
       type Block = [u8: 64]:
       type ChaChaIV = [u8; 12];
                                                                                                                                let t_ChaChaKey = t_Array u8 (sz 32)
       type ChaChaKey = [u8; 32];
       #[hax::requires a < 12 56 b < 16 86 d < 16)]
                                                                                                                                 let t_State = t_Array u32 (sz 16)
       fn chacha20_line(a: usize, b: usize, d: usize, s: u32, m: State) → State {
           let mut state = m:
                                                                                                                                let chacha20_line (a b d: usize) (s: u32) (m: t_Array u32 (sz 16))
           state[a] = state[a].wrapping_add(state[b]);
                                                                                                                                     : Prims.Pure (t_Array u32 (sz 16))
           state[d] = state[d] ^ state[a];
                                                                                                                                      (requires a <. sz 12 & b <. sz 16 & d <. sz 16)
           state[d] = state[d].rotate left(s):
                                                                                                                                      (fun _ → Prims.l_True) =
           state
                                                                                                                                  let state:t_Array u32 (sz 16) = m in
                                                                                                                                   let state:t_Array u32 (sz 16) =
                                                                                                                                    Rust_primitives.Hax.update_at state
 20
       #[hax::requires(a < 12 & b < 16 & c < 12 & d < 16)]
       pub fn chacha20_quarter_round(a: usize, b: usize, c: usize, d: usize, state: State) → State {
                                                                                                                                      (Core.Num.impl_u32_wrapping_add (state.[ a ] <: u32) (state.[ b ] <: u32) <: u32)
           let state = chacha20_line(a, b, d, 16, state);
           let state = chacha20_line(c, d, b, 12, state);
                                                                                                                                  let state:t_Array u32 (sz 16) =
           let state = chacha20_line(a, b, d, 8, state);
                                                                                                                                    Rust_primitives.Hax.update_at state d ((state.[ d ] <: u32) ^. (state.[ a ] <: u32) <: u32)
           chacha20_line(c, d, b, 7, state)
                                                                                                                                   let state:t_Array u32 (sz 16) =
                                                                                                                                    Rust_primitives.Hax.update_at state
       fn chacha20_double_round(state: State) → State {
           let state = chacha20_quarter_round(0, 4, 8, 12, state);
                                                                                                                                      (Core.Num.impl_u32_rotate_left (state.[ d ] <: u32) s <: u32)</pre>
           let state = chacha20_quarter_round(1, 5, 9, 13, state);
           let state = chacha20_quarter_round(2, 6, 10, 14, state);
                                                                                                                                  state
           let state = chacha20_quarter_round(3, 7, 11, 15, state);
                                                                                                                                 let chacha20_quarter_round (a b c d: usize) (state: t_Array u32 (sz 16))
           let state = chacha20 guarter round(0, 5, 10, 15, state):
                                                                                                                                    : Prims.Pure (t Arrav u32 (sz 16))
           let state = chacha20_quarter_round(1, 6, 11, 12, state);
                                                                                                                                      (requires a <, sz 12 & b <, sz 16 & c <, sz 12 & d <, sz 16)
           let state = chacha20_quarter_round(2, 7, 8, 13, state);
                                                                                                                                      (fun _ → Prims.l_True) =
           chacha20_quarter_round(3, 4, 9, 14, state)
                                                                                                                                  let state:t_Array u32 (sz 16) = chacha20_line a b d 16ul state in
                                                                                                                                  let state:t_Array u32 (sz 16) = chacha20_line c d b 12ul state in
                                                                                                                                  let state:t_Array u32 (sz 16) = chacha20_line a b d 8ul state in
       pub fn chacha20_rounds(state: State) → State {
                                                                                                                                   chacha20 line c d b 7ul state
           let mut st = state:
                                                                                                                           46
           for _i in 0..10 {
                                                                                                                                 let chacha20_double_round (state: t_Array u32 (sz 16)) : t_Array u32 (sz 16) =
               st = chacha20 double round(st):
                                                                                                                                  let state:t_Array u32 (sz 16) = chacha20_quarter_round (sz 0) (sz 4) (sz 8) (sz 12) state in
                                                                                                                                  let state:t_Array u32 (sz 16) = chacha20_quarter_round (sz 1) (sz 5) (sz 9) (sz 13) state in
                                                                                                                                  let state:t_Array u32 (sz 16) = chacha20_quarter_round (sz 2) (sz 6) (sz 10) (sz 14) state in
                                                                                                                                  let state:t_Array u32 (sz 16) = chacha20_quarter_round (sz 3) (sz 7) (sz 11) (sz 15) state in
                                                                                                                                  let state:t_Array u32 (sz 16) = chacha20_quarter_round (sz 0) (sz 5) (sz 10) (sz 15) state in
       nuh fo chachaja consister uzz eta: Stato) - Stato
```

P main\* ↔ 1 × 0 △ 0 0 2 × 0 
P main\* ↔ 1 × 0 △ 0 0 2 × 0 
P main\* ↔ 1 × 0 △ 0 0 2 
P main\* ↔ 1 × 0 △ 0 0 0 2

o Lucas Franceschino, 4 weeks ago via PR #295 Ln 46, Col 1 Spaces: 2 UTF-8 LF F\* 🎲 Spell ⊘ Prettier 🗋

			$\leftrightarrow$ $\rightarrow$ $\bigcirc$ Untitled (Wo	rkspace)		
EXPLORER ····	🖲 lib.r	rs ×	÷C	) · · · · • • • • • • • • • • • • • • •	4 <sub>2</sub> E 🗗 D	$\Sigma$ zsh - limited-order-book + $_{\vee}$ 🖯 🏛 $\cdots$ $\times$
$\sim$ OPEN EDITORS	hax >	examples > limited-order-book > src > 🐵 lib.rs > 🗟 Ma	tch		··/monos/bax/oxamplos/li	mited-order-book ) / main ± ) cargo hax into -i '-** +**::process_order' fstar
× 🐵 lib.rs examples/limited-ord		impl OrderBook <mark>{</mark>			Compiling candid v0.9	.11
✓ UNTITLED (WORKSPACE)		pub fn list_asks(&self) $\rightarrow$ Vec <order></order>			Compiling ic-cdk-macr Compiling ic-cdk-macr	os v0.7.1
> .github	129			TRA	Compiling ic-cdk v0.1	0.0
> .hints	131	.collect()		WERMannenover.		v0.1.0 (/Users/franziskus/repos/hax/examples/limited-order-book)
> .utils					backend.fst"	rs/franziskus/repos/hax/examples/limited-order-book/proofs/fstar/extraction/Lob_
> cli						and the second second of the second
> engine				""SDawronafenia.	Finished dev [unopti //repos/hax/examples/li	mized + debuginfo] target(s) in 3.29s mited-order-book / p main ± /
✓ examples		fn process_order <t>(mut order: Order, oth</t>	er_side: &mut BinaryHeap <t>) <math>\rightarrow</math> (Vec<match>, Opt</match></t>	ion<		
∽ chacha20    ●				THE COMPANY OF THE PARTY OF THE		
∽ proofs ●		T: Into <order> + From<order> + Ord +</order></order>	Clone,	A State of the second sec		
> coq						
✓ fstar/extraction		<pre>let mut matches = Vec::new();</pre>		- server mar soul		
≣ Chacha20.fst M		<pre>let mut done = false; for _i in 1other_side.len() {</pre>				
E Chacha20.Hacspe M		if !done {				
{} hax.fst.config.json		if let Some(m) = other_side				
M Makefile		.peek()				
∨ src 🄍			nto(other.clone()).try_match(ℴ))			
hacspec_helper.rs						
🐵 lib.rs 🛛 M						
> tests		order.quantity -= m.quant	ity;			
Cargo.toml						
M Makefile			<pre>to::into(other_side.pop().unwrap());</pre>			
$\checkmark$ limited-order-book						
$\sim$ proofs		other.quantity -= m.quant if other.quantity > 0 {	1ty;			
> coq			:from(other.clone()));			
$\sim$ fstar/ <u>extraction</u>		}				
≣ Lob_backend.fst		<pre>matches.push(m);</pre>				
M Makefile		} else {				
∽ src		done = true;				
eanister.rs						
® lib.rs						
Cargo.toml						
≣ lob_backend.did						
M Makefile		matches,				
③ README.md		<pre>if order.quantity &gt; 0 {     Some(order)</pre>				
≻ sha256		} else {				
> target		None				
≣ Cargo.lock		}.				
🌣 Cargo.toml						
≣ default.nix						
> OUTLINE						
> TIMELINE		pub mod canister;				
> RUST DEPENDENCIES						
> TAG TREE						



#### > tests M Makefile

#### cargo hax into -i '-\*\* +\*\*::process\_order' fstar Cargo.toml

✓ limited-order-book	
✓ proofs	122 Other.quantity m.quantity;
> coq	153 if other.quantity > 0 {
	154 other_side.push(From::from(other.clone()));
$\sim$ fstar/ <u>extraction</u>	
≣ Lob_backend.fst	156 matches.push(m);
M Makefile	
∽ src	158 done = true;
canister.rs	
🐵 lib.rs	
Cargo.toml	
≣ lob_backend.did	
M Makefile	163 matches,
README.md	164 if order.quantity > 0 {
> sha256	165 Some(order)
	166 } else {
> target	167 None
≣ Cargo.lock	
🌣 Cargo.toml	
≣ default.nix	
OUTLINE	171 172 pub mod canister;
> TIMELINE	172 pub mod canister; 173
> RUST DEPENDENCIES	
> TAG TREE	
<mark>۶ Բ</mark> main* -⊖ ֆԲ⊗0_∆0.@	) 2 👷 0 🤗 Live Share 🛛 🛞 rust-analyzer 🗇 opam(4.14.1)

	$\leftarrow$ $\rightarrow$ $\bigcirc$ Untit	tled (Workspace)	
® lib.rs M ×		≣ Lob_backend.fst 2, M X ≣ Core.Cmp.fsti 1, M	ι <>  ↔  ⊕  Ш ···
<pre>w Tubes M × hat &gt; examples &gt; limited-order-book &gt; src &gt; @ lib.rs &gt; () impl Order &gt; @ try_match Lucas Franceschino, 4 weeks ago [ 1 author (Lucas Franceschino) 4 Lucas Franceschino, 4 weeks ago [ 1 author (Lucas Franceschino) 3 # fiderive(CandidType, Deserialize)] 4 pub bid_id: OrderId, 2 pub bid_id: OrderId, 2 pub quantity: Quantity, 2 } 3 3 1 fn is_match(order: 60rder, other: 60rder) → bool { 3</pre>	<pre>print and a second s second second seco</pre>	<pre>be Long_Dackendist 2, M × = Core.Chp.ist 1, M // = Core.Chp.ist 2, M × = Core.C</pre>	al 8 British annuar in the second of the s
<pre>53 price: self.price, 54 quantity, 55 }) 56 } else { 57 None 58 } 59 } 60 } 60 } 61 62 impl PartialOrd for Order { 63 fn partial_cmp(6self, other: 6Self) → Option<std::cmp::ordering> 64 Some(self.cmp(other)) 65 } 66 } 77 68 impl Ord for Order { 69 fn cmp(6self, other: 6Self) → std::cmp::Ordering { 77 78 impl Ord for Order { 79 fn cmp(6self, other: 6Self) → std::cmp::Ordering { 70 fn cmp(6self, other: 6Self) → std::cmp::Ordering { 71 fn cmp(6self, other: 6Self) → std::cmp::Ordering { 72 fn cmp(6self, other: 6Self) → fn Stare for the for the form form form for the form form for the form for the form form for the form form for the form form form for the form form for the form form for the form form form form form for the form form form form for the form form form form form form form form</std::cmp::ordering></pre>		108       let hoist2:v_T = Core.Option.impl_unwrap hoist1 in         109       let obist2:v_T = Core.Option.impl_unwrap hoist1 in         100       let (other: t_Order):t_Order = Core.Convert.f_into hoist2 in         110       let other:t_Order =         111         { other with f_quantity = other.f_quantity - m.f_quantity <: t_Order         112       in         113       let other.side:Alloc.Collections.Binary_heap.t_BinaryHeap v_T =         114       if other.f_quantity >. 0uL         115       then         116       let other_side:Alloc.Collections.Binary_heap.t_BinaryHeap v_T =         117       Alloc.Collections.Binary_heap.impl_9_push other_side         118         (Core.Convert.f_from (Core.Clone.f_clone other <: t_Order)         120       in         121       else other_side         122       in         123       let matches:Alloc.Vec.t_Vec t_Match Alloc.Alloc.t_Global =         124       Allor Vec imal 1 _nush matches m	

🂉 🌮 main\* ↔ 🕼 ⊗ 0 🛆 3 🛈 6 👷 0 🔗 Live Share 🛛 🛞 rust-analyzer 🗇 opam(4.14.1)

Ln 154, Col 1 Spaces: 2 UTF-8 LF F\* 🍓 Spell ⊘ Prettier 🚨



# Thanks



https://github.com/hacspec/hax

