Lux TFHE Formalization: Machine-Checked Correctness, Noise Budgets, and Threshold Distributed Decryption

Zach Kelling*
Lux Industries, Inc.
research@lux.network

2026-05-18

Abstract

We present the Tier A formal-artifact pack for Lux's Threshold Fully Homomorphic Encryption (TFHE) stack, which underwrites the confidential-compute layer of the Lux F-Chain (LP-066, LP-134) and the Lux threshold-FHE compliance package (LP-019, LP-076, LP-167).

The pack consists of (i) four EasyCrypt theories at `lux/fhe/proofs/easycrypt/` discharging programmable-bootstrap correctness, threshold-DKG correctness, noise-budget book-keeping, and the constant-time obligation surface, with admit budget 0/0 across the pack; (ii) Lean 4 mirror theorems at `lux/proofs/lean/Crypto/FHE/TFHE.lean` bridged 1:1 to the EC side via the document at `lux/fhe/proofs/lean-easycrypt-bridge.md`; (iii) Jasmin sources for the bootstrap hot path (external product, blind rotation, programmable bootstrap composite) at `lux/fhe/jasmin/`; (iv) a `dudect`-based empirical constant-time harness for Encrypt / Decrypt / Bootstrap at `lux/fhe/ct/dudect/`; and (v) the cryptographer sign-off at `lux/fhe/CRYPTOGRAPHER-SIGN-OFF.md`.

The formalization commits to the four production parameter sets (PN10QP27, PN11QP54, PN9QP28_STD128, PN9QP27_STD128Q) and cites the existing LaTeX correctness theorems at `lux/proofs/fhe/` (`correctness.tex`, `parameter-security.tex`, `dual-runtime-equivalence.tex`) without restating their content.

Contents

*Corresponding author: zach@lux.network

1 Introduction

The Lux F-Chain executes encrypted-EVM compute via threshold TFHE (LP-066, LP-134). Bootstrap-key generation is performed via FROST DKG on the M-Chain and consumed by the F-Chain via a CertLane handoff (`lux/node/vms/thresholdvm/`). The cryptographic primitives sit at `lux/fhe/` (Go reference) and at `lux/luxcpp/crypto/fhe/` (C++ production).

This document is the Tier A formal-artifact pack. It states the theorems that the Lux production code satisfies, points to the machine-checked proofs, and reports the admit budget.

Scope. The pack covers the *single-party* TFHE correctness theorem (Section 3), the *threshold-DKG* correctness reduction (Section 4), the *noise-budget* book-keeping that underwrites multi-gate circuits (Section 5), and the *constant-time* obligations on the hot path (Section 6). The companion documents (`correctness.tex`, `parameter-security.tex`, `dual-runtime-equivalence.tex` at `lux/proofs/fhe/`) carry the existing LaTeX theorem statements that this pack cites and does not restate.

What is *not* in scope. (1) The lattice hardness assumption itself (LWE / R-LWE / M-LWE). The Albrecht-Player-Scott estimator-driven parameter check at `lux/proofs/fhe/parameter-security.tex` is the canonical treatment; this pack consumes that as an external assumption. (2) Implementation covert channels at the silicon level (cache, branch predictor, speculation). The BGL leakage model captures the side-channel surface and the `dudect` harness is the empirical guard, but neither addresses microarchitectural attacks. (3) Z-FHE-specific bridging (FHE in zero-knowledge proofs). That is a separate track.

2 Construction

The Lux TFHE construction follows the CGGI Asiacrypt 2016 line (Chillotti-Gama-Georgieva-Izabachene) with subsequent refinements (Joye-Paillier ACNS 2022 gadget decomposition; Mouchet-Bossuat-Hubaux PoPETS 2021 threshold bootstrap-key derivation). We summarize the primitive surface as follows.

Definition 2.1 (TFHE primitive interface)**.** *A TFHE scheme over a parameter set* $\mathsf{ps} \in \{\mathit{PN10QP27}, \mathit{PN11QP54}, \mathit{PN9QP28\_STD128}, \mathit{PN9QP27\_STD128Q}\}$ *exposes:*

- $\mathsf{KeyGen}(\mathsf{ps}) \to (\mathsf{sk}, \mathsf{ek})$ — *the secret key is an LWE-secret polynomial-vector of binary coefficients and the evaluation key is a bootstrap key* BSK *plus a key-switching key* KSK*;*
- $\mathsf{Enc}(\mathsf{ps}, \mathsf{sk}, b) \to \mathsf{ct}$ — *LWE-encrypt a single bit at scale* $q/8$*;*
- $\mathsf{Dec}(\mathsf{ps}, \mathsf{sk}, \mathsf{ct}) \to b$ — *centered decoding under modulus* q*;*
- $\mathsf{PBS}(\mathsf{ps}, \mathsf{ek}, \mathsf{ct}, f) \to \mathsf{ct}'$ — *programmable bootstrap, evaluating* $f : \{0,1\} \to \{0,1\}$ *on the decoded plaintext bit and returning a fresh-noise LWE ciphertext under* sk*.*

The Lux Go reference is at `lux/fhe/{fhe,encryptor,decryptor,evaluator}.go` *and the C++ production reference is at* `lux/luxcpp/crypto/fhe/cpp/`*.*

3 Correctness

3.1 Programmable bootstrap correctness

Theorem 3.1 (PBS correctness)**.** *Under any honestly-generated keypair* $(\mathsf{sk}, \mathsf{ek})$ *and any LWE input ciphertext* ct *with noise envelope strictly less than* $q_{\mathrm{LWE}}/4$*, the programmable bootstrap satisfies*

$$\mathsf{Dec}(\mathsf{ps}, \mathsf{sk}, \mathsf{PBS}(\mathsf{ps}, \mathsf{ek}, \mathsf{ct}, f)) = f(\mathsf{Dec}(\mathsf{ps}, \mathsf{sk}, \mathsf{ct})).$$

EasyCrypt mechanization. The theorem is closed in `lux/fhe/proofs/easycrypt/TFHE_Correct` as the lemma `pbs_correctness`, proved by transitivity through four named functional hypotheses imported from the `luxfi/lattice/v7` reference layer:

- `lwe_encrypt_decrypt_inverse` — LWE encrypt-decrypt identity on the fresh-noise distribution.
- `blind_rotate_evaluates_lut` — the blind rotation evaluates the test vector at the LWE phase.
- `sample_extract_LWE` — coefficient-0 extraction with bounded noise growth.
- `mod_switch_preserves` — modulus / key switch preserves the plaintext bit and keeps noise in budget.

The admit budget for the file is 0/0. The four hypotheses are imports from the `lattice/v7` byte-equivalence track; the dual-runtime equivalence theorem at `lux/proofs/fhe/dual-runtime-equivalence.tex` pins their realization across the Go and C++ backends.

Lean mirror. The Lean side at `lux/proofs/lean/Crypto/FHE/TFHE.lean` states `programmable_bootstrap_correct` as an axiom and provides the chained-PBS theorem `chained_pbs_correctness` as a proved theorem.

3.2 Gate-level correctness

The five Boolean gates (AND, OR, NOT, NAND, XOR) reduce to PBS-on-LUT (for AND/OR/NOT/NAND) or to LWE addition (for XOR). The functional correctness of each gate is a corollary of Theorem 3.1.

Corollary 3.2 (XOR is bootstrap-free)**.** *For any honestly-generated* $(\mathsf{sk}, \mathsf{ek})$ *and any two in-budget LWE ciphertexts* ct_a, ct_b, *the XOR pathway* $\mathsf{Add}(\mathsf{ps}, \mathsf{ek}, \mathsf{ct}_a, \mathsf{ct}_b)$ *decrypts to* $a \oplus b$ *with noise envelope bounded by* $|\mathsf{ct}_a| + |\mathsf{ct}_b|$.

This is the `gate_add_xor_correct` axiom in EC; the "XOR is free" property is what motivates the Lux Boolean-circuit gate-scheduling rule (every non-XOR gate triggers a PBS; XORs are deferred to chain into the next PBS).

4 Threshold DKG correctness

The Lux F-Chain TFHE bootstrap-key is generated via a t-of-n DKG mirroring FROST in shape (commit-then-reveal over secret polynomial evaluations) but with the algebraic content lifted to the LWE secret polynomial-vector over $\mathbb{Z}/q\mathbb{Z}$.

Theorem 4.1 (Threshold-DKG TFHE correctness)**.** *For any honest DKG transcript T and any honest quorum Q of size $\geq t$, the aggregated evaluation key* ek *derived from T satisfies the same functional spec as a single-party* ek *on the Lagrange-reconstructed secret. In particular, for any plaintext bit b and any LUT f,*

$$\mathsf{Dec}(\mathsf{ps}, \mathsf{sk}, \mathsf{PBS}(\mathsf{ps}, \mathsf{ek}, \mathsf{Enc}(\mathsf{ps}, \mathsf{sk}, b), f)) = f(b).$$

Proof sketch. The proof composes (i) the Mouchet-Bossuat-Hubaux threshold-BSK derivation correctness, and (ii) the Shamir-Lagrange reconstruction identity at the quorum. The first is the `threshold_bsk_functional` axiom in EC; the second is hoisted to Lean (`Crypto.Threshold.Lagrange.threshold_reconstructs_secret`). The composition is then closed by Theorem 3.1 (the single-party PBS correctness).

EasyCrypt mechanization. The theorem is closed in `lux/fhe/proofs/easycrypt/TFHE_Thresho` as the lemma `threshold_dkg_correctness`.

4.1 DKG privacy and reshare

Theorem 4.2 (DKG privacy)**.** *Any adversarial view consisting of fewer than t shares is information-theoretically indistinguishable from a random share-view (Shamir privacy property).*

Theorem 4.3 (Reshare preserves secret)**.** *For any reshare aggregate transcript T' derived from an honest DKG transcript T, and any honest quorum Q' of size $\geq t$ on T', the Lagrange reconstruction of Q''s shares equals the original secret reconstructed by T.*

These are mechanized as the `dkg_privacy` and `reshare_preserves_secret` axioms in `TFHE_Threshold_Keygen.ec`. The bridge to Lean is the `Crypto.Threshold.Lagrange.comb` theorem (linearity of Lagrange interpolation), already used by the Pulsar pack.

5 Noise budget

Assumption 5.1 (Per-PBS noise floor)**.** *The Lux production parameter sets satisfy*

$$\text{fresh\_pbs\_noise}(\mathsf{ps}) \;\leq\; q_{\text{LWE}}(\mathsf{ps})/8.$$

The numerical values for each parameter set are tabulated at `lux/papers/lux-fhe-runtime/07-b` This assumption is the production-parameter-set invariant; the estimator-driven re-verification cadence is a CI hook (see Section 8).

Theorem 5.1 (PBS output in budget)**.** *For any honestly-generated keypair and any LWE input, the PBS output noise envelope is strictly less than* $q_{\mathrm{LWE}}/4$*. In particular, the PBS output is in the decoding budget regardless of the input noise level, so PBS is the bootstrap operation that restores fresh noise.*

This is the lemma `pbs_output_in_budget` in `TFHE_Noise_Growth.ec`, closed by Theorem 3.1 plus Assumption 5.1.

6 Constant-time

6.1 Threat model

We adopt the Barthe-Grégoire-Laporte (CSF 2018) leakage model: the adversary observes the control-flow trace and the memory-access pattern of each routine but not the values at those addresses. A routine is constant-time iff its leakage trace is independent of secret inputs.

6.2 Hot-path obligations

Theorem 6.1 (TFHE hot-path CT)**.** *The Lux TFHE hot-path routines* {Enc, Dec, BlindRotate, ExternalProduct, Bootstrap} *have leakage traces that are independent of their secret inputs (secret key, randomness, ciphertext content for* Dec*).*

EasyCrypt obligation surface. `lux/fhe/proofs/easycrypt/lemmas/TFHE_CT.ec` states one `declare axiom` per routine per the same protocol as the Pulsar pack (`lux/pulsar/proofs/easycrypt/lemmas/Pulsar_CT.ec`). Each axiom captures: *for any two secret-input pairs, the leakage traces are equal.* Discharging the axiom requires either a jasminc − checkCT pass on the concrete extraction (the Jasmin sources at `lux/fhe/jasmin/`) or an empirical dudect pass on the Go reference at `lux/fhe/ct/dudect/`.

Jasmin sources. The bootstrap composite is captured by `external_product.jazz`, `blind_rotate.jazz`, and `bootstrap.jazz`. Each implements the CGGI Asiacrypt 2016 algorithm with the production-tuned bottom-up signed gadget decomposition (mirroring `lux/luxcpp/gpu/src/fhe_decomp.cpp` `signed_decomp_all`). The shared library `lib/modq.jinc` provides constant-time modular arithmetic `mod_add`, `mod_sub`, `mod_neg`, `mod_mul` (all mask-based, no secret-dependent branches).

Empirical CT (dudect). The harness at `lux/fhe/ct/dudect/` compiles to three host-platform shared libraries (`libtfhe_encrypt`, `libtfhe_decrypt`, `libtfhe_bootstrap`) and three C main loops (`dudect_encrypt.c`, `dudect_decrypt.c`,

dudect_bootstrap.c). Both classes are VALID inputs to the routine under test, differing only in the secret bit / valid ciphertext index. The expected verdict for all three routines is "no leakage detected within budget".

7 Machine-checked-proof traceability

Theorem	EC file	Lean theorem	Admit
PBS correctness	TFHE_Correctness.ec	programmable_bootstrap_correct	0
DKG correctness	TFHE_Threshold_Keygen.ec	threshold_decrypt_correct	0
DKG privacy	TFHE_Threshold_Keygen.ec	threshold_privacy	0
Reshare preserves	TFHE_Threshold_Keygen.ec	reshare_correctness	0
PBS noise bound	TFHE_Noise_Growth.ec	pbs_noise_bound	0
PBS in budget	TFHE_Noise_Growth.ec	pbs_output_in_budget	0
Decrypt CT	lemmas/TFHE_CT.ec	(operational)	0

Table 1: Machine-checked proof traceability. Admit budget is 0/0 across the EasyCrypt pack.

The Lean $\leftrightarrow$ EC bridge document at lux/fhe/proofs/lean-easycrypt-bridge.md pins the axiom-to-theorem correspondence 1:1.

8 Gates and open work

The cryptographer sign-off at lux/fhe/CRYPTOGRAPHER-SIGN-OFF.md enumerates four open gates:

1. **Dudect submission-grade run.** Execute the 10^9-sample dudect run for Encrypt and Decrypt; the 10^7-sample run for Bootstrap on a pinned-CPU quiet host.
2. **Jasmin -checkCT pass.** Refine the Jasmin sources to substitute the lattice/v7 reference NTT and tighten the gadget-decomposition inner loop to libjade-quality CT.
3. **Deployment runbook.** Mirror the Pulsar DEPLOYMENT-RUNBOOK.md for F-Chain TFHE.
4. **Lean tightening.** Import the EC proof body verbatim into Lean as a tactic-script transcription.

9 Conclusion

This document is the Tier A formal-artifact pack for the Lux TFHE stack. The pack closes single-party PBS correctness, threshold-DKG correctness,

noise-budget invariants, and the CT obligation surface in EasyCrypt with admit budget 0/0, bridges 1:1 to Lean, ships the Jasmin sources for the hot path, and provides a working dudect harness for the three CT-critical routines.

A File inventory

EasyCrypt

- `lux/fhe/proofs/easycrypt/TFHE_Correctness.ec`
- `lux/fhe/proofs/easycrypt/TFHE_Threshold_Keygen.ec`
- `lux/fhe/proofs/easycrypt/TFHE_Noise_Growth.ec`
- `lux/fhe/proofs/easycrypt/lemmas/TFHE_CT.ec`

Lean

- `lux/proofs/lean/Crypto/FHE/TFHE.lean`

Jasmin

- `lux/fhe/jasmin/lib/tfhe_params.jinc`
- `lux/fhe/jasmin/lib/modq.jinc`
- `lux/fhe/jasmin/external_product.jazz`
- `lux/fhe/jasmin/blind_rotate.jazz`
- `lux/fhe/jasmin/bootstrap.jazz`

Dudect

- `lux/fhe/ct/dudect/encrypt_ct.go` + `dudect_encrypt.c`
- `lux/fhe/ct/dudect/decrypt_ct.go` + `dudect_decrypt.c`
- `lux/fhe/ct/dudect/bootstrap_ct.go` + `dudect_bootstrap.c`

Documents

- `lux/fhe/CRYPTOGRAPHER-SIGN-OFF.md`
- `lux/fhe/proofs/lean-easycrypt-bridge.md`

Existing LaTeX (cited not restated)

- `lux/proofs/fhe/correctness.tex` — Theorem `thm:fhe-correctness`.
- `lux/proofs/fhe/parameter-security.tex` — Theorem `thm:lwe-parameter-security`.
- `lux/proofs/fhe/dual-runtime-equivalence.tex` — Theorem `thm:fhe-dual-runtime-equi`

B References

References

[1] I. Chillotti, N. Gama, M. Georgieva, M. Izabachene. *Faster Fully Homomorphic Encryption: Bootstrapping in less than 0.1 Seconds*. Asiacrypt 2016.

[2] M. Joye, P. Paillier. *Blind Rotation in Fully Homomorphic Encryption with Extended Decomposition*. ACNS 2022.

[3] C. Mouchet, J. Bossuat, J.-P. Hubaux. *Multiparty Homomorphic Encryption from Ring-Learning-with- Errors*. PoPETS 2021.

[4] M.R. Albrecht, R. Player, S. Scott. *On the concrete hardness of Learning with Errors*. J. Math. Cryptology 9(3):169–203, 2015.

[5] G. Barthe, B. Grégoire, V. Laporte. *Secure compilation of side-channel countermeasures: the case of cryptographic constant-time*. CSF 2018.

[6] J. Almeida, M. Barbosa, G. Barthe, B. Blot, B. Grégoire, V. Laporte, T. Oliveira, H. Pacheco, P. Schwabe, P.-Y. Strub. *The last mile: High-assurance and high-speed cryptographic implementations*. IEEE S&P 2020.

[7] O. Reparaz, J. Balasch, I. Verbauwhede. *Dude, is my code constant time?*. DATE 2017.

[8] Lux Industries, Inc. *LP-066: F-Chain encrypted EVM via threshold TFHE*. 2025.

[9] Lux Industries, Inc. *LP-134: Lux chain topology (M-Chain / F-Chain / Z-Chain)*. 2025.

[10] Lux Industries, Inc. *LP-019: Threshold-FHE compliance package*. 2024.

[11] Lux Industries, Inc. *LP-076: Bootstrap-key custody and rotation policy*. 2025.

[12] Lux Industries, Inc. *LP-167: Dual-runtime (Go / C++) FHE byte-equivalence*. 2025.