

Master-Key Recovery from Internal Activation Leakage in ReLU Realizations of Camellia

Kwangjo Kim

IRCS and KAIST(Korea), CSU(USA)
kkj@kaist.ac.kr

Abstract. This paper investigates functional correctness and master-key reconstruction in explicitly constructed rectified linear unit (ReLU) implementations of Camellia-128, Camellia-192, and Camellia-256. Exact neural realizations of Boolean gates, S-box selectors, and bit permutations provide a framework for reproducing Camellia on binary inputs. We present a component-wise correctness argument and report known-answer-test agreement for Camellia-128. We then analyze how selected whitening and round-key quantities determine the external master key through public key-schedule identities. Inverse rotations of suitable subkey pairs recover the required key components, whereas pre-whitening-key recovery alone is insufficient for complete reconstruction of the longer keys. Under explicitly constructed synthetic internal-oracle access, symmetric real-valued perturbations identify target subkey bits, requiring 128 and 256 query pairs for the coordinate-wise Camellia-128 and Camellia-256 procedures, respectively. A separate Camellia-192 experiment validates deterministic key-schedule inversion under direct access to DNN-generated round keys. Across 1,000 independently sampled master keys per variant, all three procedures achieved exact recovery in every trial, yielding 3,000 successful recoveries out of 3,000 trials, with zero recovery failures and no Camellia-192 complement-consistency failures. These results validate conditional master-key reconstruction under the specified internal-access models and highlight the distinction between binary functional correctness and security under stronger access assumptions. They do not establish plaintext/ciphertext-only attacks against standard Camellia.

Keywords: Camellia, DNN Leakage, Key Recovery, ReLU Activation, Neural Cryptanalysis, Master-Key Recovery

1 Introduction

Block ciphers are fundamental primitives for protecting data confidentiality and, through appropriate cryptographic constructions, supporting data integrity and authentication. Their practical security depends on both resistance to cryptanalytic attacks and protection of secret keys

within implementations. A mathematically secure cipher can nevertheless expose key-dependent information when its implementation provides capabilities beyond the conventional plaintext/ciphertext interface.

The development of modern block cipher standards reflects the need for adequate key sizes and efficient implementations. Following concerns about the 56-bit key of the Data Encryption Standard (DES) [5], NIST initiated the Advanced Encryption Standard (AES) selection process in 1997 and standardized AES [6] in 2001. AES uses 128-bit blocks and supports 128-, 192-, and 256-bit keys. Other prominent block ciphers include the Korean standard ARIA [4], the Japanese standard Camellia [7], and the Chinese standard SM4 [8], each of which features distinct internal structures and implementation characteristics.

Camellia, jointly developed by NTT and Mitsubishi Electric, is a 128-bit block cipher supporting the same three key lengths as AES. Its Feistel structure combines nonlinear S-boxes and linear diffusion with input/output whitening and key-dependent FL and FL^{-1} layers. Its key schedule derives subkeys from rotations of 128-bit internal values. These features make Camellia a useful case study for examining how internal key-dependent information can be related to the external master key.

1.1 Motivation and Research Objectives

Boolean cryptographic computations can be represented by explicitly constructed networks of rectified linear units (ReLU). Natural gate and corner-based constructions reproduce Boolean operations exactly on binary inputs, as demonstrated in [2] and its subsequent EUROCRYPT 2026 publication [3]. Their behavior on real-valued inputs, however, is an additional property of the chosen implementation. In particular, the ReLU realization of XOR computes an absolute difference, whose response to a real-valued perturbation can reveal a fixed binary operand when the relevant internal operation is accessible.

The objective of this paper is to examine this distinction for Camellia: functional correctness on the binary domain and master-key reconstruction under explicitly specified internal-access assumptions. We first consider how Boolean gates, S-boxes, and wire operations realize Camellia as a ReLU network. We then examine which whitening or round-key quantities suffice to reconstruct the external key for each key length. Finally, we evaluate the supplied recovery procedures while separating synthetic-oracle extraction from direct access to generated subkeys.

The analysis does not assume that an ordinary encryption interface exposes the internal quantities used by these procedures. Access to ci-

phertext outputs, real-valued encryption evaluations, internal activations, synthetic subkey oracles, and directly readable round keys are distinct capabilities. The results are interpreted under the particular capability used in each experiment.

1.2 Main Contributions and Scope

The paper makes the following contributions:

1. **A correctness framework for ReLU Camellia.** We describe exact Boolean gate and S-box realizations and explain how component-wise exactness and an identical key schedule support an inductive equivalence argument. The Camellia-128 known-answer test supplies evidence for a particular tested input, with a clear distinction between experimental agreement and universal correctness.
2. **Variant-specific master-key reconstruction.** We identify the key material needed for Camellia-128, Camellia-192, and Camellia-256. Inverse rotations of selected round-key pairs recover the relevant 128-bit key components; concatenation and, for the 192-bit variant, the complement relation complete the reconstruction. We also explain why pre-whitening-key recovery alone is insufficient for full recovery of the longer keys.
3. **Explicitly scoped recovery experiments.** For the 128- and 256-bit variants, symmetric perturbations of synthetic internal XOR oracles recover the target subkeys using 128 and 256 query pairs, respectively. Across all three Camellia variants, 1,000 independently sampled master keys per variant were recovered exactly, yielding 3,000 successful recoveries out of 3,000 trials under the specified internal-access models.

These contributions characterize reconstruction from specified internal information. They do not establish plaintext/ciphertext-only key-recovery attacks against standard Camellia or demonstrate that the synthetic oracles arise naturally in a deployed implementation. The gap between an assumed internal interface and measurable implementation leakage remains a central question for further study.

1.3 Organization of the Paper

Section 2 introduces binary and word notation, ReLU networks, natural Boolean realizations, correctness, and the leakage-access assumptions. Section 3 describes Camellia’s encryption structure, round function,

FL/FL^{-1} layers, and the key-schedule identities relevant to reconstruction. Section 4 examines Boolean/ReLU equivalence for Camellia-128 and the reported known-answer test. Section 5 analyzes master-key recovery for all three key lengths and summarizes the recovery procedures and available experimental results. Section 6 concludes the paper and outlines future work on accessible leakage targets, finite precision, noisy observations, and reproducible validation.

2 Preliminaries

This section fixes the notation and neural constructions used in Sections 3, 4, and 5. We follow the Boolean-function and natural ReLU implementation terminology of [2].

2.1 Binary Vectors and Word Operations

Let $\mathbb{F}_2 = \{0, 1\}$ be the binary field and let $\mathbb{R}$ denote the real numbers. We identify a binary vector in $\mathbb{F}_2^n$ with its coordinate-wise embedding in $\mathbb{R}^n$. An n -bit word is represented by a vector $(x_1, \dots, x_n)$ in most-significant-bit-first order, with integer value $\sum_{i=1}^n x_i 2^{n-i}$. Hexadecimal strings use the corresponding big-endian byte order. The notation $A \parallel B$ denotes concatenation, with A as the higher-order component.

For words of equal length, $\oplus$, $\wedge$, $\vee$, and $\bar{A}$ denote bitwise XOR, AND, OR, and complement, respectively. Complementation is always restricted to the specified word length. The operations $\text{ROTL}_n(A, t)$ and $\text{ROTR}_n(A, t)$ rotate an n -bit word left and right by t positions, with t taken modulo n . They are mutual inverses:

$$\text{ROTR}_n(\text{ROTL}_n(A, t), t) = A. \quad (1)$$

The notation $A \lll t$ means a left rotation at the word width specified by the context; ROL_1 in the FL function means a one-bit left rotation of a 32-bit word.

A Boolean function is a map $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$, and a vectorial Boolean function is a map $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$. The set of scalar Boolean functions is denoted by $\mathcal{B}_n$. A fixed ordering of the input vectors gives a truth table that uniquely specifies f . Equivalently, f has a unique algebraic normal form (ANF), a multilinear polynomial

$$f(x) = \bigoplus_{v \in \mathbb{F}_2^n} a_v \prod_{i=1}^n x_i^{v_i}, \quad a_v \in \mathbb{F}_2, \quad (2)$$

where arithmetic is over $\mathbb{F}_2$. The fast Möbius transform converts between the truth table and ANF using $O(n2^n)$ bit additions and $O(2^n)$ bits of storage [1]. The S-box construction below uses the truth table directly.

2.2 ReLU Networks and Exact Boolean Gates

The rectified linear unit is $\text{ReLU}(z) = \max\{0, z\}$. A ReLU neuron with weight vector w and bias b computes $\eta(x) = \text{ReLU}(w^\top x + b)$. A feed-forward ReLU network consists of affine maps followed by coordinate-wise ReLU activations, with an affine output layer:

$$\begin{aligned} h^{(0)} &= x, \\ h^{(\ell)} &= \text{ReLU}(W_\ell h^{(\ell-1)} + b_\ell), \quad 1 \leq \ell \leq L, \\ D(x) &= W_{L+1} h^{(L)} + b_{L+1}. \end{aligned} \tag{3}$$

Here $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ and $b_\ell \in \mathbb{R}^{d_\ell}$. The term DNN refers to this explicitly constructed network; training is not required for the gate identities used in this paper.

Define the real-valued gate extensions

$$\begin{aligned} \text{XOR}_R(x, y) &= \text{ReLU}(x - y) + \text{ReLU}(y - x), \\ \text{AND}_R(x, y) &= \text{ReLU}(x + y - 1), \\ \text{OR}_R(x, y) &= x + y - \text{AND}_R(x, y), \\ \text{NOT}_R(x) &= 1 - x. \end{aligned} \tag{4}$$

On binary inputs these equal the corresponding Boolean gates. In particular, $\text{XOR}_R(x, y) = |x - y|$ for real inputs. Vector gates act coordinate-wise, and multi-input XORs follow the composition order of the implemented circuit. Bit permutations, concatenations, and rotations are exact wire rearrangements. These primitives realize the Boolean operations appearing in Camellia's key schedule, diffusion layer, and FL/FL^{-1} functions.

2.3 Natural Implementations and S-box Selectors

For a binary corner $b \in \{0, 1\}^n$ and $0 < c \leq 1$, define

$$\text{corner}_{c,b}(x) = \text{ReLU}\left(\sum_{i:b_i=1} x_i + \sum_{i:b_i=0} (1 - x_i) - n + c\right). \tag{5}$$

On binary inputs this function equals c at $x = b$ and zero elsewhere. Consequently, for the active-corner set $\mathcal{C} = \{b : f(b) = 1\}$, an exact natural realization is

$$\text{Impl}_{f,c}(x) = \frac{1}{c} \sum_{b \in \mathcal{C}} \text{corner}_{c,b}(x). \quad (6)$$

The factor $1/c$ is necessary for binary output one when $c \neq 1$. The implementations considered here use $c = 1$, so that the unnormalized sum already agrees with f on every binary input.

For an 8-bit S-box $s : \{0, 1\}^8 \rightarrow \{0, 1\}^8$, set $m_b(x) = \text{corner}_{1,b}(x)$ for all 256 byte values. Its j -th output coordinate is realized by

$$\text{Impl}_s(x)_j = \sum_{b \in \{0,1\}^8} s(b)_j m_b(x), \quad 1 \leq j \leq 8. \quad (7)$$

Exactly one selector is active on a binary byte, giving the correct S-box output. The eight coordinates can share the same 256 hidden ReLU selectors and use separate affine output sums. This construction applies to each of Camellia’s four related S-boxes. Its behavior on nonbinary inputs is determined by the chosen real extension and is not specified by the Boolean truth table alone.

2.4 Keyed Implementations and Correctness

For $\lambda \in \{128, 192, 256\}$, let $\mathcal{K}_\lambda = \{0, 1\}^\lambda$ be the master-key space. Write $E_B(K, P)$ for conventional Boolean Camellia encryption of $P \in \{0, 1\}^{128}$ and $E_{\text{DNN}}(K, x)$ for its specified ReLU realization on $x \in \mathbb{R}^{128}$. The key remains binary in both cases. Correctness means agreement on the binary domain:

$$E_{\text{DNN}}(K, P) = E_B(K, P) \quad \text{for every } K \in \mathcal{K}_\lambda, P \in \{0, 1\}^{128}. \quad (8)$$

Exact component realizations and identical subkeys allow this property to be established by induction over the encryption stages. A known-answer test checks a particular key/plaintext/ciphertext triple and provides implementation evidence, rather than a proof of universal equivalence. Correctness on binary inputs places no security requirement on evaluations outside that domain.

We use K for the external master key; K_L, K_R, K_A, K_B for 128-bit key-schedule values; kw_i for 64-bit whitening keys; k_r for 64-bit round keys; and kl_i for the keys used by FL/FL^{-1} . For Camellia-192, R denotes

the additional 64-bit master-key component, so $K_R = R \parallel \overline{R}$. These symbols are distinct from the Feistel state halves L_r, R_r . Their variant-specific relations are given in Section 3. In the recovery experiments, the local symbol S denotes a target subkey pair; in the cipher specification, S denotes the substitution layer.

2.5 Leakage Access and Symmetric Perturbations

We distinguish access to encryption outputs from access to internal key-dependent quantities. A conventional encryption oracle accepts binary plaintexts and returns ciphertexts. A real-input encryption oracle additionally evaluates the specified ReLU circuit on nonbinary inputs. Neither interface, by itself, grants observations of internal activations or round keys.

The experiments in Section 5 assume stronger, explicitly stated access. The Camellia-128 and Camellia-256 experiments construct a synthetic internal oracle for selected subkey bits. The Camellia-192 experiment directly reads generated round keys and tests deterministic key-schedule inversion. Such access must be distinguished from leakage actually available in a deployed cipher. An attacker may choose queries adaptively, although the symmetric queries below do not require adaptation.

For a fixed binary target $S = (S_1, \dots, S_n)$, the synthetic oracle’s scalar response is

$$O_i(x) = \text{XOR}_R(x, S_i) = |x - S_i|. \quad (9)$$

A soft query is a real-valued probe. For $0 < \epsilon < 1/2$, the symmetric probes $x^- = 1/2 - \epsilon$ and $x^+ = 1/2 + \epsilon$ give

$$\Delta_i = O_i(x^+) - O_i(x^-) = 2\epsilon(1 - 2S_i). \quad (10)$$

Thus $\Delta_i > 0$ identifies $S_i = 0$, and $\Delta_i < 0$ identifies $S_i = 1$. A coordinate-wise procedure uses n query pairs, or $2n$ evaluations of the stated oracle, to recover n target bits. Scalar and vector oracle calls must be counted according to the interface actually used. The relation above assumes exact responses; finite precision or measurement error must preserve the sign of Δ_i for recovery to remain valid.

Recovered subkeys are subsequently mapped to the external key by public inverse rotations and concatenations. Complete master-key recovery means exact equality $\hat{K} = K$, not merely recovery of a single round key or an effective XOR mask. In repeated trials, each master key is sampled independently, and the empirical success rate is the number of exact recoveries divided by the number of trials. For Camellia-192, the

additional consistency condition $K_{R,R} = \overline{K_{R,L}}$ checks the required complement relation.

3 Specification of Camellia Block Cipher

Camellia is a symmetric-key block cipher with a 128-bit block size and supports secret keys of 128, 192, and 256 bits. Its overall structure is based on a Feistel network augmented with input/output whitening and additional FL and FL^{-1} transformations. For a 128-bit key, Camellia employs 18 Feistel rounds, whereas the 192- and 256-bit variants employ 24 rounds.

3.1 Encryption Structure

Let the 128-bit plaintext be denoted by

$$M = M_L \parallel M_R,$$

where each component is 64 bits. Before entering the Feistel network, Camellia performs key whitening using two 64-bit whitening subkeys kw_1 and kw_2 :

$$L_0 \parallel R_0 = M \oplus (kw_1 \parallel kw_2).$$

For an ordinary Feistel round r , the state is updated as

$$L_r = R_{r-1} \oplus F(L_{r-1}, k_r), \quad (11)$$

$$R_r = L_{r-1}, \quad (12)$$

where k_r denotes the 64-bit round subkey.

For Camellia-128, the cipher consists of 18 rounds and introduces FL and FL^{-1} layers after rounds 6 and 12. The final ciphertext is obtained by swapping the two Feistel halves and applying post-whitening:

$$C = (R_{18} \parallel L_{18}) \oplus (kw_3 \parallel kw_4).$$

Camellia-192 and Camellia-256 extend the Feistel network to 24 rounds. In these variants, FL and FL^{-1} layers are inserted after rounds 6, 12, and 18, and the ciphertext is computed as

$$C = (R_{24} \parallel L_{24}) \oplus (kw_3 \parallel kw_4).$$

Decryption uses essentially the same Feistel structure with the subkeys applied in reverse order.

3.2 Round Function

The nonlinear round function F operates on a 64-bit input X and a 64-bit round key k and is defined as

$$F(X, k) = P(S(X \oplus k)),$$

where S denotes the nonlinear substitution layer and P denotes a linear diffusion transformation.

Writing

$$X \oplus k = x_1 \parallel x_2 \parallel \cdots \parallel x_8,$$

with $x_i \in \{0, 1\}^8$, the substitution layer independently processes the eight bytes using four related 8-bit S-boxes s_1, s_2, s_3, s_4 . In particular,

$$\begin{aligned} S(X) = & s_1(x_1) \parallel s_2(x_2) \parallel s_3(x_3) \parallel s_4(x_4) \\ & \parallel s_2(x_5) \parallel s_3(x_6) \parallel s_4(x_7) \parallel s_1(x_8). \end{aligned}$$

The resulting eight bytes are subsequently combined by the linear P transformation using XOR operations. Thus, the key-dependent input $X \oplus k$ immediately precedes the nonlinear S-box layer, which makes this intermediate state particularly relevant when considering implementations that expose internal activations.

3.3 FL and FL^{-1} Functions

In addition to the Feistel round function, Camellia introduces the key-dependent FL and FL^{-1} functions. For

$$X = X_L \parallel X_R, \quad kl = kl_L \parallel kl_R,$$

where all components are 32 bits, the FL transformation is defined by

$$Y_R = \text{ROL}_1(X_L \wedge kl_L) \oplus X_R, \tag{13}$$

$$Y_L = (Y_R \vee kl_R) \oplus X_L. \tag{14}$$

The FL^{-1} function performs the corresponding inverse transformation. These layers introduce additional key-dependent Boolean operations between groups of Feistel rounds.

3.4 Key Schedule

The Camellia key schedule is based on four 128-bit quantities K_L , K_R , K_A , and K_B . The relationship between the external key and (K_L, K_R) depends on the key size.

For Camellia-128,

$$K_L = K, \quad K_R = 0.$$

For Camellia-192, if

$$K = K_L \parallel K_{R,L},$$

where $K_{R,L}$ is the additional 64-bit part of the key, then

$$K_R = K_{R,L} \parallel \overline{K_{R,L}}.$$

For Camellia-256,

$$K = K_L \parallel K_R,$$

where both K_L and K_R are 128-bit quantities.

The auxiliary values K_A and K_B are generated from K_L and K_R through several applications of the F function using six fixed 64-bit constants $\Sigma_1, \dots, \Sigma_6$. The value K_B is required only for the 192- and 256-bit variants.

The whitening keys, round keys, and FL/FL^{-1} keys are then obtained primarily as selected 64-bit halves of fixed rotations of K_L , K_R , K_A , and K_B . Consequently, several pairs of consecutive 64-bit subkeys constitute an entire rotated 128-bit internal key quantity.

For example, for Camellia-128, the specification gives

$$k_3 \parallel k_4 = K_L \lll 15.$$

Since $K_L = K$ for the 128-bit variant, knowledge of both k_3 and k_4 directly determines the external key:

$$\boxed{K = \text{ROTR}_{128}(k_3 \parallel k_4, 15)}. \quad (15)$$

For the 192- and 256-bit variants, the specification gives

$$k_3 \parallel k_4 = K_R \lll 15$$

and

$$k_9 \parallel k_{10} = K_L \lll 45.$$

Hence the corresponding 128-bit internal quantities can be recovered as

$$K_R = \text{ROTR}_{128}(k_3 \parallel k_4, 15), \quad (16)$$

$$K_L = \text{ROTR}_{128}(k_9 \parallel k_{10}, 45). \quad (17)$$

In particular, for Camellia-256, these two relations recover the complete 256-bit external key:

$$\boxed{K = \text{ROTR}_{128}(k_9 \parallel k_{10}, 45) \parallel \text{ROTR}_{128}(k_3 \parallel k_4, 15)}. \quad (18)$$

For Camellia-192, recovery of K_L together with the appropriate 64-bit component of K_R determines the 192-bit external key, because the remaining half of K_R is defined as the bitwise complement of that component.

3.5 Relevance to Internal-State Key Recovery

Two properties of Camellia are particularly important for internal-state leakage analysis. First, each round evaluates the nonlinear transformation

$$S(X \oplus k),$$

so that internal values immediately before or after the S-box layer depend directly on the corresponding 64-bit round subkey. Leakage from such an intermediate representation can therefore potentially be exploited to infer round-key material.

Second, the key schedule does not merely produce unrelated round keys: many subkey pairs are direct rotated representations of K_L or K_R . Consequently, recovery of suitably selected consecutive round keys can be converted algebraically into recovery of the external secret key without exhaustively reversing the entire key schedule. This property provides a useful bridge between internal round-key leakage and full master-key recovery in Camellia-128, Camellia-192, and Camellia-256.

4 Correctness of the Boolean and ReLU-DNN Implementations

We assess the functional correctness of the ReLU-DNN realization of Camellia-128 by comparing it with a conventional Boolean implementation. Let $E_B(K, P)$ and $E_{\text{DNN}}(K, P)$ denote the ciphertexts produced

by the two implementations for a 128-bit key K and plaintext P , respectively. The intended correctness property is

$$E_{\text{DNN}}(K, P) = E_{\text{B}}(K, P) \quad \text{for all } K, P \in \{0, 1\}^{128}. \quad (19)$$

The two implementations follow the same Camellia-128 encryption schedule: input whitening, 18 Feistel rounds, insertion of FL and FL^{-1} after rounds 6 and 12, and final swapping and output whitening. The Boolean version operates on 64-bit words, whereas the ReLU-DNN version represents these words as bit vectors and uses neural realizations of the corresponding operations.

For binary inputs $a, b \in \{0, 1\}$, XOR and AND admit the exact ReLU identities

$$a \oplus b = \text{ReLU}(a - b) + \text{ReLU}(b - a), \quad (20)$$

$$a \wedge b = \text{ReLU}(a + b - 1). \quad (21)$$

Together with exact bit permutations, rotations, and a correctly realized Camellia S-box, these identities provide a basis for bit-exact evaluation of the F , FL , and FL^{-1} functions. If the ReLU-DNN key schedule also generates subkeys identical to the Boolean key schedule, equality of the initial states and induction over the 18 rounds imply Eq. (19). This argument is conditional on bit-exact implementation of every component, including the S-box.

We additionally checked both implementations against the standard Camellia-128 known-answer test vector:

$$K = 0123456789\text{ABCDEF} \text{FEDCBA}9876543210,$$

$$P = 0123456789\text{ABCDEF} \text{FEDCBA}9876543210,$$

$$C = 67673138549669730857065648\text{EABE}43.$$

The supplied verification output reports $E_{\text{B}}(K, P) = E_{\text{DNN}}(K, P) = C$. Thus, both implementations pass this known-answer test, supporting the correctness of the ReLU-DNN realization for the tested input. The reported test alone does not establish equivalence for every key and plaintext; that stronger conclusion depends on the component-wise exactness conditions stated above.

5 DNN Leakage and Camellia Master-Key Recovery

Camellia’s key schedule derives whitening and round keys from rotated versions of the 128-bit internal values K_L , K_R , K_A , and K_B . Consequently, sufficiently complete leaked subkey material can be mapped back

to the external master key. In particular, the initial whitening keys satisfy $kw_1 \parallel kw_2 = K_L$ for all three key lengths. For Camellia-128, $K = K_L$; thus, recovering both 64-bit pre-whitening keys immediately reveals the entire master key. For Camellia-192, $K = K_L \parallel R$, where R is 64 bits and the key schedule defines $K_R = R \parallel \overline{R}$. For Camellia-256, $K = K_L \parallel K_R$, with 128-bit K_R . Therefore, recovering K_L alone leaves 64 or 128 unknown master-key bits, respectively.

Table 1. Camellia variants under a pre-whitening-key leakage model.

Property	128	192	256
Pre-whitening relation	$kw_1 \parallel kw_2 = K_L$		
Pre-whitening size	128 bits	128 bits	128 bits
External key	K_L	$K_L \parallel R$	$K_L \parallel K_R$
Recoverable material	K_L (128)	K_L (128)	K_L (128)
Recovered fraction	Full key	First 128 bits	First 128 bits
Remaining key bits	0	64	128
Direct inversion	Yes	Partial	Partial
Additional recovery	None	R	K_R
Inversion target	K_L	(K_L, R)	(K_L, K_R)

Note: Recoverable material assumes leakage sufficient to reconstruct the complete 128-bit pre-whitening key; it is not guaranteed by arbitrary DNN access.

Full recovery of Camellia-192/256 additionally requires information about K_R . For example, if suitable subkey pairs reveal rotated copies of both K_L and K_R , inverse 128-bit rotations recover those values; in the 192-bit case, the leftmost 64 bits of K_R yield R , while its rightmost 64 bits provide the check $K_{R,R} = \overline{K_{R,L}}$. Alternatively, the Feistel-style steps used to derive K_A and K_B can be reversed when the necessary intermediate values are available, since an update $X' = X \oplus F(Y, \Sigma_i)$ implies $X = X' \oplus F(Y, \Sigma_i)$ for known Y .

In the considered neural-leakage setting, extraction of the required internal key material can therefore reduce master-key reconstruction to deterministic key-schedule inversion. This conclusion is conditional on the leakage model: one arbitrary 64-bit round key, or a partial observation of an activation, does not by itself establish full master-key recovery.

5.1 Camellia-128

The Camellia-128 program recovers the 128-bit subkey pair $S = k_3 \parallel k_4 = \text{ROTL}_{128}(K, 15)$ through an explicitly constructed internal oracle whose

bitwise response is $O_i(x) = \text{ReLU}(x - S_i) + \text{ReLU}(S_i - x)$. For each bit, queries at $x = 1/2 \pm \epsilon$, with $\epsilon = 0.01$, give a response difference $\Delta_i = 2\epsilon(1 - 2S_i)$; its sign identifies S_i . Thus, 128 query pairs (256 oracle calls) recover S , and the external key follows as $K = \text{ROTR}_{128}(S, 15)$. The v4 script is configured for 1,000 trials, each using a fresh random 128-bit master key and a fresh random binary probe. In the reported experiment, all 1,000 reconstructed keys matched their originals exactly (1,000/1,000; 100%), with zero failures. The program also includes Boolean/ReLU encryption checks against a known test vector. Recovery assumes a synthetic internal key-schedule oracle; the first-round effective mask $kw_1 \oplus k_1$ alone does not determine the full master key.

5.2 Camellia-192

The Camellia-192 program directly accesses the round keys generated by its ReLU key schedule and reconstructs the internal values as

$$\begin{aligned} K_R &= \text{ROTR}_{128}(k_3 \parallel k_4, 15), \\ K_L &= \text{ROTR}_{128}(k_9 \parallel k_{10}, 45). \end{aligned} \tag{22}$$

Writing $K_R = R \parallel \overline{R}$, the external key is $K = K_L \parallel R$. The complement relation supplies an additional consistency check. In the reported 1,000-trial experiment using the v2 program, 1,000 independently sampled 192-bit master keys were recovered exactly (1,000/1,000; 100%), with zero recovery failures and zero complement-consistency failures. The program also includes an embedded known-answer test for the Boolean and ReLU implementations. This experiment validates key-schedule inversion under complete subkey access; it does not perform a perturbation-based extraction of those subkeys.

5.3 Camellia-256

The Camellia-256 program applies the same symmetric-query rule to two 128-bit targets, $k_3 \parallel k_4$ and $k_9 \parallel k_{10}$, using $\epsilon = 0.10$. A total of 256 query pairs (512 scalar oracle calls) recover both targets. Inverse rotations by 15 and 45 bits, respectively, yield K_R and K_L , and hence the full external key $K = K_L \parallel K_R$. The v1 script defaults to 1,000 fresh random-key trials and checks the recovered subkey pairs, both key components, and the complete 256-bit key; it also includes Boolean/ReLU encryption comparisons. In the reported experiment, all 1,000 external keys were recovered exactly (1,000/1,000; 100%), with zero failures. As in the 128-bit experiment,

recovery assumes an explicitly constructed synthetic internal oracle with real-valued query access.

Source Code Availability. To facilitate reproducibility and independent verification, the Python implementations for Camellia-128, Camellia-192, and Camellia-256, including the ReLU-based neural representations, internal-state access models, and master-key reconstruction experiments, are publicly available at https://github.com/kjkim0410/Camellia-DNN-Key_Recovery. The repository also provides instructions for reproducing the reported experiments.

6 Conclusion

This paper examined the functional correctness and key-leakage implications of explicitly constructed ReLU implementations of Camellia. Exact neural realizations of Boolean gates, S-box selectors, and bit permutations provide a basis for agreement with conventional Camellia on binary inputs. For Camellia-128, component-wise exactness supports an inductive correctness argument, while the reported known-answer test provides evidence for the tested key and plaintext. Agreement on binary inputs, however, does not by itself establish security under real-valued queries or observations of internal states.

The key-recovery analysis links selected subkey material to the external master key through public key-schedule identities. For Camellia-128, recovering $k_3 \parallel k_4$ permits reconstruction of the entire key by an inverse 128-bit rotation. For Camellia-192 and Camellia-256, the pairs $k_3 \parallel k_4$ and $k_9 \parallel k_{10}$ reveal rotated copies of K_R and K_L . The 192-bit key is then reconstructed using the additional 64-bit component and its complement relation, whereas the 256-bit key is obtained by concatenating the two recovered 128-bit components. Recovery of the initial whitening keys alone reveals K_L and is therefore insufficient for full recovery of the longer variants.

The experiments employ different access assumptions. The Camellia-128 and Camellia-256 programs extract target subkey bits from explicitly constructed synthetic internal oracles using symmetric real-valued perturbations. Their coordinate-wise procedures require 128 and 256 query pairs, respectively, followed by deterministic key reconstruction. The Camellia-192 program instead reads generated round keys directly and validates key-schedule inversion. In the reported validation, all 1,000 independently sampled 192-bit master keys were recovered exactly, with no complement consistency failures. Across 1,000 independently sampled master

keys for each Camellia variant, all three recovery procedures achieved exact master-key reconstruction, resulting in 3,000 successful trials with zero recovery failures under their respective internal-access assumptions.

These results establish conditional master-key reconstruction from specified internal information; they do not demonstrate key recovery from ordinary plaintext/ciphertext-only access to standard Camellia. The distinction between an exposed activation, a synthetic oracle, and direct sub-key access is essential when interpreting the results. Future work should identify naturally accessible leakage targets in complete implementations, evaluate the effects of finite precision and noisy observations, and extend reproducible validation to more realistic leakage environments. Evaluating restrictions on nonbinary inputs and protection of internal key-dependent values would further clarify how neural implementations can preserve both functional correctness and implementation security.

Acknowledgment

This work was conducted by the author during a research visit to Kyushu University, Japan, with support from the International Exchange Program of the National Institute of Information and Communications Technology (NICT), Japan.

References

1. Carlet, C.: Boolean functions for cryptography and error correcting codes. In: Crama, Y., Hammer, P.L. (eds.) *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, pp. 257–397. Springer, New York, NY, USA (2010). https://doi.org/10.1007/978-0-387-93873-0_7
2. G  rault, D., Hambitzer, A., Ronen, E., Shamir, A.: How to securely implement cryptography in deep neural networks. Report 288, IACR Cryptology ePrint Archive (2025), <https://eprint.iacr.org/2025/288.pdf>
3. G  rault, D., Hambitzer, A., Ronen, E., Shamir, A.: Deep neural cryptography. In: Daemen, J., Thom  , E. (eds.) *Advances in Cryptology – EUROCRYPT 2026*. Lecture Notes in Computer Science, vol. 16546, pp. 513–543. Springer (2026), <https://dblp.org/rec/conf/eurocrypt/GeraulthRS26>
4. Kwon, D., Kim, J.Y., Kim, S.J., Kim, S.W., Park, Y.S., Hong, S., Cheon, J.H., Kim, H.Y., Lee, J.G., Lee, S., Lee, S.: New block cipher: ARIA. In: Lim, J.I., Lee, D.H. (eds.) *Information Security and Cryptology – ICISC 2003*. Lecture Notes in Computer Science, vol. 2971, pp. 432–445. Springer (2004). https://doi.org/10.1007/978-3-540-24691-6_32
5. National Bureau of Standards / National Institute of Standards and Technology: Data encryption standard (des), federal information processing standards publication 46. Tech. Rep. FIPS PUB 46, U.S. Department of Commerce, NIST (1977), <https://csrc.nist.gov/pubs/fips/46/final>, withdrawn. Superseded by later revisions (FIPS 46-1, 46-2, 46-3).

6. National Institute of Standards and Technology: Advanced encryption standard (aes). FIPS PUB 197 (2001), <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf>
7. NTT Social Informatics Laboratories: Camellia: A 128-bit block cipher suitable for multiple platforms. <https://info.is1.ntt.co.jp/crypt/eng/camellia/>, official Camellia website, accessed October 8, 2026
8. Standardization Administration of China: Information security technology: SM4 block cipher algorithm. Chinese National Standard GB/T 32907-2016, Standardization Administration of China (Aug 2016), <https://openstd.samr.gov.cn/bzgk/std/newGbInfo?hcno=7803DE42D3BC5E80B0C3E5D8E873D56A>