On Key Recovery from Exposed Neural XOR Activations in DES and TDES

Kwangjo Kim[1] ⋆

KAIST/IRCS, Daejeon, Republic of Korea
kkj@kaist.ac.kr

Abstract. We study the security of natural ReLU realizations of the Data Encryption Standard (DES) and two-key Triple DES (TDES) when an adversary can submit non-binary inputs and observe selected intermediate neural values. The attack exploits the piecewise-linear extension of XOR: paired inputs that differ in one expanded right-half bit reveal the corresponding round-key bit from the change in a natural XOR node. For DES, the resulting 48-bit first-round subkey leaves only the eight key-schedule bits omitted by PC-2; enumerating these bits and checking an intermediate value recovers the 56-bit effective master key. For two-key EDE TDES, the same technique recovers the first encryption subkey of K_1 and the last DES subkey used by the middle decryption under K_2; two independent 2^8 completions then recover the full 112-bit key pair. The exposed round subkey is recovered with a constant number of leakage observations per bit, giving query complexity linear in the targeted subkey length; the remaining searches are fixed-size for DES. Python experiments report exact recovery for 1,000 random DES keys and 1,000 random two-key TDES key pairs. These results concern a synthetic intermediate-oracle model for natural neural implementations and do not claim a physical attack on conventional DES/TDES hardware or software.

Keywords: DES · Triple DES · neural cryptanalysis · ReLU network · key recovery · intermediate-state oracle

1 Introduction

The Data Encryption Standard (DES) is a 64-bit Feistel block cipher with a 56-bit effective key, standardized in FIPS 46-3 [15]. Its small key space led to practical exhaustive search, while differential and linear cryptanalysis provided influential structural attacks [5,14]. Two-key Triple DES (TDES) extends the lifetime of the DES primitive by composing three DES operations. In its two-key EDE form, encryption is $E_{K_1} \circ$

⋆ Dr. Kim is supported by the National Institute of Information and Communications Technology (NICT), Japan, under the NICT International Research Program.

$D_{K_2} \circ E_{K_1}$. NIST SP 800-67 Rev. 2 documents TDEA [2]; the publication was withdrawn in 2024 and TDEA is no longer approved for applying new cryptographic protection. DES and TDES nevertheless remain useful historical case studies and may occur in legacy systems.

This paper considers a different question from classical DES cryptanalysis. Suppose a cryptographic Boolean circuit is translated into a ReLU network that is exact on binary inputs. Such a translation defines the network on the entire real domain, and its off-domain behavior need not inherit the security of the original Boolean primitive. Natural neural implementations based on corner indicators are particularly transparent: secret-dependent activation boundaries can be exposed by chosen real-valued or encoded perturbations. This representation-leakage phenomenon was formalized by Gerault et al. [8].

We analyze natural ReLU realizations of DES and two-key TDES under an explicit synthetic intermediate-state oracle. The oracle returns the 48 outputs of the natural XOR layer immediately after DES expansion and round-key addition at a selected stage. This is stronger than ordinary plaintext/ciphertext black-box access. For TDES, a separate stage-trace capability supplies the intermediate stage values used only to resolve the eight PC-2 omissions. The model is best viewed as an instrumented neural implementation or a restricted white-box leakage interface; it is not a model of conventional DES software or hardware.

What is new beyond the general representation-leakage framework. Gerault et al. [8] provide the general distinction between Boolean correctness and security over real-valued neural extensions. Our contribution is the cipher-specific end-to-end derivation for DES and two-key EDE TDES: we identify the expansion/key-XOR interface as a coordinate-separable leakage surface, map a recovered 48-bit subkey back through the DES key schedule, exploit the fact that PC-2 omits exactly eight bits, handle the reversed round-key order in the middle TDES decryption stage, and reconstruct both 56-bit component keys. Thus, the contribution is not a new classical attack on DES; it is a complete representation-leakage case study that connects exposed neural activations to external master-key recovery.

Our contributions.

– We give a bit-exact natural ReLU construction of the DES round function, including natural XOR and S-box components, and separate Boolean correctness from off-domain neural behavior.

– We prove coordinate-wise separability at the exposed expansion/key-XOR interface. A constant number q of perturbation pairs per coordinate gives $O(48q)$ pair complexity for a 48-bit DES round subkey; in the deterministic noiseless model, $q = 1$ is sufficient.
– We partially invert PC-2 and enumerate its eight omitted bits to recover a complete 56-bit DES key. For two-key EDE TDES, we recover $K_1^{(1)}$ and $K_{16}^{(2)}$ and perform two independent 2^8 completions.
– We evaluate the construction over 1,000 random DES keys and 1,000 random TDES key pairs. All reported round subkeys and effective master keys are recovered exactly under the stated synthetic oracle.

Scope and terminology. We use *linear-query recovery* to mean that the number of leakage observations grows linearly with the number of targeted subkey bits. We do not claim a linear-time break of ordinary DES or TDES from standard binary plaintext/ciphertext access. The fixed 2^8 candidate completions arise from the DES key-schedule structure and are accounted for separately.

Organization. Section 2 defines natural neural components. Section 3 reviews DES and TDES. Sections 4–6 present the realization, oracle, and recovery algorithms. Section 7 reports the experiments, and Sections 8–10 discuss limitations, related work, and conclusions.

2 Preliminaries

Let $\{0,1\}^n$ denote the set of binary vectors of length n. For vectors $x, y \in \{0,1\}^n$, $\mathrm{HD}(x,y)$ is their Hamming distance. We write $x\|y$ for concatenation and number bit positions from one, consistently with the DES tables. The rectified linear unit is

$$\mathrm{ReLU}(t) = \max\{0, t\}. \tag{1}$$

2.1 Natural Corner Representation

For $x \in \mathbb{R}^\ell$, a Boolean corner $a \in \{0,1\}^\ell$, and a parameter $c > 0$, define

$$\phi_c(x; a) = \mathrm{ReLU}\left(\frac{\sum_{i=1}^{\ell}(1 - |x_i - a_i|) - \ell + c}{c}\right). \tag{2}$$

Absolute values can themselves be represented by two ReLUs. On the Boolean domain, $\phi_c(a; a) = 1$, and for $c \le 1$ every different Boolean corner

has value zero. Hence a Boolean output coordinate $f : \{0,1\}^\ell \to \{0,1\}$ has the exact natural extension

$$\mathsf{N}_f(x) = \sum_{a \in \{0,1\}^\ell : f(a)=1} \phi_c(x; a), \qquad x \in \{0,1\}^\ell. \tag{3}$$

For XOR, the two active corners are $(0,1)$ and $(1,0)$:

$$\mathrm{NXOR}_c(x,k) = \phi_c((x,k);(0,1)) + \phi_c((x,k);(1,0)). \tag{4}$$

Thus $\mathrm{NXOR}_c(x,k) = x \oplus k$ for $x, k \in \{0,1\}$, but NXOR_c is piecewise linear for real-valued inputs.

2.2 Correctness and Security Domains

Let $B_K : \{0,1\}^n \to \{0,1\}^m$ be a keyed Boolean primitive and $N_K : \mathbb{R}^n \to \mathbb{R}^m$ its neural realization.

Definition 1 (Binary correctness). *N is correct for B if $N_K(x) = B_K(x)$ for every key K and every $x \in \{0,1\}^n$.*

Binary correctness constrains only the vertices of the hypercube. It does not constrain real-valued inputs between those vertices or the visibility of intermediate neurons. Our attacks use precisely this distinction.

2.3 Key and Subkey Notation

A DES key is represented externally by 56 effective bits. In the FIPS format, one odd-parity bit is inserted into each byte, producing a 64-bit key. PC-1 removes parity and permutes the effective bits into $C_0 \| D_0 \in \{0,1\}^{56}$. At round r, rotations produce $C_r \| D_r$, and PC-2 selects 48 of these bits as the round key K_r. Superscripts distinguish TDES component keys; for example, $K_{16}^{(2)}$ is round key 16 derived from master key K_2.

3 DES and Two-Key TDES

3.1 DES

For a 64-bit plaintext P, DES applies the initial permutation IP and splits the result into $L_0, R_0 \in \{0,1\}^{32}$. For $r = 1, \ldots, 16$,

$$L_r = R_{r-1}, \tag{5}$$

$$R_r = L_{r-1} \oplus F(R_{r-1}, K_r), \tag{6}$$

Algorithm 1 DES Encryption

Input: $P \in \{0,1\}^{64}$ and a 64-bit parity-formatted key K
Output: $C = E_K(P)$
$K_1, \ldots, K_{16} \leftarrow \textsc{KeySchedule}(K)$
$L_0 \| R_0 \leftarrow \mathrm{IP}(P)$
for $r = 1$ to 16 **do**
    $L_r \leftarrow R_{r-1}$
    $R_r \leftarrow L_{r-1} \oplus F(R_{r-1}, K_r)$
return $\mathrm{IP}^{-1}(R_{16} \| L_{16})$

where

$$F(R, K_r) = P\big(S(E(R) \oplus K_r)\big). \tag{7}$$

The expansion E maps 32 bits to 48 bits, S applies eight parallel $6 \to 4$ S-boxes, and P permutes 32 bits. The final ciphertext is $\mathrm{IP}^{-1}(R_{16} \| L_{16})$. DES Encryption is described in Algorithm 1. For simplicity, the implementation details of DES are omitted. DES Decryption uses the same Feistel structure with round keys in reverse order.

3.2 Two-Key EDE TDES

Two-key EDE TDES uses $K_3 = K_1$ and computes

$$\mathrm{TDEA}_{K_1,K_2}(P) = E_{K_1}\big(D_{K_2}(E_{K_1}(P))\big). \tag{8}$$

Writing $A = E_{K_1}(P)$ and $B = D_{K_2}(A)$ isolates the three stages. The first stage begins with $K_1^{(1)}$. Because DES decryption reverses the key order, the middle stage begins with $K_{16}^{(2)}$. These are the two subkeys targeted by our leakage oracles.

TDES does not enlarge the block size: it remains 64 bits and therefore retains birthday-bound limitations such as those exploited by Sweet32 [4]. Our attack is unrelated to block collisions; it targets the neural implementation's intermediate geometry.

4 Natural ReLU Realization of DES

The natural realization retains all DES permutations and the key schedule. It replaces every XOR by Eq. (4). For S-box S_j and output bit q, let

$$\begin{aligned} \mathcal{A}_{j,q} &= \{a \in \{0,1\}^6 : (S_j(a))_q = 1\}, \\ \mathsf{NS}_{j,q}(x) &= \sum_{a \in \mathcal{A}_{j,q}} \phi_c(x; a). \end{aligned} \tag{9}$$

Algorithm 2 Natural ReLU DES Encryption

Input: $P \in \{0,1\}^{64}$, key K, and $0 < c \leq 1$
Output: Neural ciphertext C

$K_1, \ldots, K_{16} \leftarrow \textsc{KeySchedule}(K)$
$L_0 \| R_0 \leftarrow \mathrm{IP}(P)$
for $r = 1$ to 16 **do**
  $U \leftarrow E(R_{r-1})$
  for $i = 1$ to 48 **do**
    $X_i \leftarrow \mathrm{NXOR}_c(U_i, K_{r,i})$
  for $j = 1$ to 8 **do**
    $Y^{(j)} \leftarrow (\mathsf{NS}_{j,1}(X^{(j)}), \ldots, \mathsf{NS}_{j,4}(X^{(j)}))$ ▷ $X^{(j)} = X_{6j-5} \ldots X_{6j}$
  $V \leftarrow P(Y^{(1)} \| \cdots \| Y^{(8)})$
  $L_r \leftarrow R_{r-1}$
  for $i = 1$ to 32 **do**
    $R_r[i] \leftarrow \mathrm{NXOR}_c(L_{r-1}[i], V_i)$
return $\mathrm{IP}^{-1}(R_{16} \| L_{16})$

Each natural S-box therefore agrees exactly with the DES S-box on binary inputs. The construction is mathematical and need not be trained.

Lemma 1 (Boolean-domain equivalence). *For every Boolean plaintext and DES key,* **Algorithm** *2 returns the same ciphertext as conventional DES.*

Proof. Eq. (4) equals XOR at every Boolean input pair, and Eq. (9) equals each S-box output bit at all 64 Boolean inputs. IP, E, P, IP^{-1}, and the key schedule are unchanged. Induction on the 16 Feistel rounds proves equality of all round states and hence of the final ciphertext.

The implementation may threshold or round natural component outputs during a binary correctness test. The leakage oracle, however, observes the real value of a selected XOR node before any such discretization. This distinction must be preserved in both the implementation and the security statement.

5 Attack and Oracle Model

5.1 Two Explicit Capabilities

To avoid conflating intermediate neural leakage with TDES stage instrumentation, we separate the adversary's capabilities.

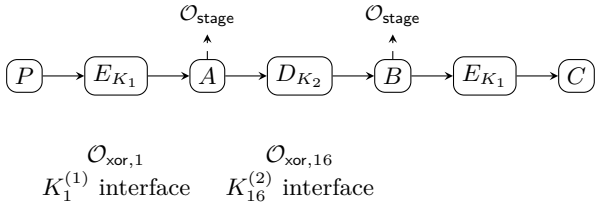

Fig. 1. Attack surface for two-key EDE TDES. Round-XOR leakage recovers the two targeted DES subkeys; stage values A and B are separate instrumentation used for PC-2 candidate resolution.

Round-XOR leakage oracle $\mathcal{O}_{\mathsf{xor}}$. For a selected DES stage and round key $K_r \in \{0,1\}^{48}$, the oracle accepts a 64-bit stage input X and returns the 48-dimensional vector

$$\begin{aligned} \mathcal{O}_{\mathsf{xor},r}(X)[i] &= \mathrm{NXOR}_c(\eta(E(R_0)[i]), \eta(K_r[i])), \\ L_0 \| R_0 &= \mathrm{IP}(X). \end{aligned} \tag{10}$$

In symmetric mode, $\eta(b) = b$. In change mode, $\eta(0) = \varepsilon$ and $\eta(1) = 1 - \varepsilon$ for small $\varepsilon > 0$. This oracle exposes selected internal neural values and is strictly stronger than a conventional encryption API.

TDES stage-trace oracle $\mathcal{O}_{\mathsf{stage}}$. For the EDE composition, we additionally assume instrumentation that reveals

$$A = E_{K_1}(P), \qquad B = D_{K_2}(A), \qquad C = E_{K_1}(B). \tag{11}$$

The round-subkey attack itself uses $\mathcal{O}_{\mathsf{xor}}$; the values A and B from $\mathcal{O}_{\mathsf{stage}}$ are used only to select the correct master-key candidate after the eight PC-2 omissions are enumerated. Separating these capabilities makes explicit that the TDES end-to-end recovery assumes more than neural XOR leakage alone.

5.2 Pair Generation

For target coordinate i, let $\rho(i) = E[i]$ be the right-half position copied to expansion coordinate i. Choose arbitrary L_0, R_0 and form R_0' by flipping only $R_0[\rho(i)]$. Applying IP^{-1} gives two stage inputs X and X' for which the targeted expanded bit is complementary. Expansion duplicates boundary bits, so other expansion coordinates may also change; this does not affect coordinate i because the oracle returns coordinates separately.

Algorithm 3 Recovery of a 48-Bit DES Round Key

Input: Oracle $\mathcal{O}_r$ and odd repetition count q
Output: $\widehat{K}_r \in \{0,1\}^{48}$
1: **for** $i = 1$ to 48 **do**
  $v_0, v_1 \leftarrow 0, 0$
  for $t = 1$ to q **do**
    $(X, X', u) \leftarrow \textsc{GeneratePair}(i)$
    $z \leftarrow \mathcal{O}_r(X)[i]$; $z' \leftarrow \mathcal{O}_r(X')[i]$
    $(a, b) \leftarrow (z, z')$ if $u = 0$, else (z', z)
    if $a < b$ **then**
      $v_0 \leftarrow v_0 + 1$
    else
      $v_1 \leftarrow v_1 + 1$
  $\widehat{K}_r[i] \leftarrow 0$ if $v_0 > v_1$, else 1
2: **return** $\widehat{K}_r$

Lemma 2 (Bit separability at the round-XOR interface). *Let $u \in \{0,1\}$ and $u' = 1 - u$. Given u and the ordered observations*

$$z = \mathrm{NXOR}_c(\eta(u), \eta(k)), \qquad z' = \mathrm{NXOR}_c(\eta(u'), \eta(k)), \tag{12}$$

the secret bit k is uniquely determined for the encodings and parameters used in our experiments.

Proof. For Boolean encoding, Eq. (4) gives the ordered pairs $(0,1)$ for $k = 0$ and $(1,0)$ for $k = 1$ when $u = 0$. The order reverses when $u = 1$, which is known to the attacker. With change encoding and $0 < \varepsilon \ll c \leq 1$, the same strict ordering is preserved by continuity inside the corresponding linear regions. Comparing the two observed values therefore determines k.

The implementation repeats this comparison $q = 5$ times and takes a majority vote. Repetition is not required by the deterministic noiseless model: Lemma 2 implies that $q = 1$ suffices. We retain $q = 5$ in the reported TDES implementation as an engineering choice that also provides a natural hook for future noisy-oracle experiments; it should not be interpreted as evidence of physical-noise robustness.

6 DES and TDES Master-Key Recovery

6.1 Round-Subkey Recovery

Algorithm 3 applies **Lemma** 2 independently to all 48 coordinates.

Algorithm 4 Full Recovery of Two-Key EDE TDES

Input: Stage oracles $\mathcal{O}_1^{(1)}, \mathcal{O}_{16}^{(2)}$; trace (P, A, B, C)
Output: Effective master keys $(\widehat{K}_1, \widehat{K}_2)$
$\widehat{K}_1^{(1)} \leftarrow$ RecoverRoundKey$(\mathcal{O}_1^{(1)})$
$\widehat{K}_{16}^{(2)} \leftarrow$ RecoverRoundKey$(\mathcal{O}_{16}^{(2)})$
$\mathcal{C}_1 \leftarrow$ EnumeratePC2Completions$(\widehat{K}_1^{(1)}, 1)$
$\mathcal{C}_2 \leftarrow$ EnumeratePC2Completions$(\widehat{K}_{16}^{(2)}, 16)$
$\widehat{K}_1 \leftarrow$ the sole surviving $K \in \mathcal{C}_1$ with $E_K(P) = A$
$\widehat{K}_2 \leftarrow$ the sole surviving $K \in \mathcal{C}_2$ with $D_K(A) = B$
Verify $E_{\widehat{K}_1}(D_{\widehat{K}_2}(E_{\widehat{K}_1}(P))) = C$
return $(\widehat{K}_1, \widehat{K}_2)$

6.2 From a DES Round Key to the Master Key

PC-2 selects 48 of the 56 bits in $C_r \| D_r$. Inverting PC-2 therefore fixes 48 positions and leaves exactly eight unknown. For K_1, rotate both 28-bit halves right by one position to obtain a partial $C_0 \| D_0$. For K_{16}, the DES left-shift schedule sums to 28 positions over all 16 rounds (4 one-bit shifts and 12 two-bit shifts). Thus each 28-bit half returns to its original orientation, so $C_{16} \| D_{16} = C_0 \| D_0$ and no rotation is needed. Enumerate the eight missing bits, invert PC-1, restore odd parity, and test the 256 candidates.

For standalone DES, the supplied code computes the natural-DES ciphertext of the all-zero plaintext under the true key and tests each candidate against that known pair. Full recovery of two-key TDES is summarized in **Algorithm** 4. The K_1 candidates are tested against $A = E_{K_1}(P)$, while K_2 candidates are tested against $B = D_{K_2}(A)$. A second full plaintext/ciphertext pair validates the recovered key pair end to end.

6.3 Complexity

We distinguish a *perturbation pair* from an individual oracle invocation. For each of the 48 target coordinates and each repetition, Algorithm 3 constructs one pair (X, X') and invokes the vector-valued oracle once on each member. Hence DES recovery uses $48q$ perturbation pairs, or $96q$ oracle invocations; every invocation returns a 48-dimensional activation vector, from which only the targeted coordinate is consumed. Master-key completion performs at most 2^8 DES candidate evaluations. Two-key TDES doubles these principal costs:

$$Q_{\mathrm{TDES}}^{\mathrm{inv}} = 2 \cdot 96q, \qquad W_{\mathrm{complete}} \leq 2 \cdot 2^8 \tag{13}$$

plus a constant number of final validations. Equivalently, TDES uses $2 \cdot 48q$ perturbation pairs. More generally, the leakage phase is $O(nq)$ for an n-bit exposed XOR interface. For DES, $n = 48$ and the eight-bit completion is a fixed structural cost.

7 Experimental Evaluation

7.1 Implementation and Methodology

The experiments are implemented in Python with NumPy. DES follows the FIPS permutation, S-box, Feistel, and key-schedule tables. Effective 56-bit keys are sampled uniformly as seven bytes and converted to the 64-bit DES format by inserting odd parity. The neural leakage layer evaluates the natural XOR corner sum with $c = 0.5$. The implementation observes the real-valued XOR output before any thresholding used for Boolean-domain correctness checks.

The standalone DES experiment evaluates 1,000 independently generated keys in both symmetric and change pair modes. It recovers the first-round subkey, enumerates the eight PC-2 omissions, and compares the recovered external key with the original. The retained Windows run reports 1,000/1,000 exact recoveries and a total time of 34,482.5 seconds. Because only selected example timings and the total were retained in the supplied program report, we do not infer unavailable minimum, maximum, variance, or software-version statistics.

The two-key TDES experiment uses random (K_1, K_2) pairs, random seed 20260810, $q = 5$, $c = 0.5$, and $\varepsilon = 10^{-2}$. For every pair it recovers $K_1^{(1)}$ and $K_{16}^{(2)}$, generates 256 candidates per component key, filters them with (P, A, B), and validates the result on two complete EDE plaintext/ciphertext pairs. An independent execution of the supplied TDES program completed all 1,000 pairs in 680.2 seconds. The mean, minimum, and maximum recovery times per pair were 0.68, 0.59, and 6.71 seconds.

Code Availability: To support reproducibility and independent verification, the Python proof-of-concept implementations for DES and two-key Triple DES (TDES), together with the experimental results, are publicly available at:

https://github.com/kjkim0410/DES-TDES-DNN-Key-Recovery

The repository includes Boolean and ReLU-based implementations, correctness verification, and master-key recovery experiments under synthetic internal-activation leakage assumptions, including 10-trial and 1,000-trial evaluations.

Table 1. Parameters and retained reproducibility metadata. "Not retained" means that the supplied experiment record does not support a more specific claim.

Item	Retained value
Implementation	Python with NumPy
DES trials	1,000 keys
TDES trials	1,000 key pairs
TDES random seed	20260810
c	0.5
ε (TDES change mode)	10^{-2}
q (TDES reported run)	5
DES run OS	Windows
Python/NumPy versions	Not retained
CPU / thread configuration	Not retained

7.2 Results

The experimental results are summarized in Table 2.

Table 2. Exact-recovery results over 1,000 random instances. HD denotes Hamming distance from the true effective key or round subkey.

Experiment	Target	Exact recovery	Mean/max. HD
DES, symmetric	K_1, 56-bit K	1,000/1,000	0/0
DES, change	K_1	1,000/1,000	0/0
Two-key TDES	$K_1^{(1)}$, $K_{16}^{(2)}$	1,000/1,000	0/0
Two-key TDES	56-bit K_1, K_2	1,000/1,000	0/0

The zero Hamming distances show that candidate completion did not merely produce functionally equivalent outputs on the validation vectors: it reconstructed the original 56-bit effective keys. In the reported TDES trials, each recovered subkey generated 256 candidates and the instrumented stage relation selected one survivor for each component key. The implementation then validated the recovered pair on a second complete EDE plaintext/ciphertext pair. In general, a single finite input/output test need not provide a mathematical guarantee of candidate uniqueness, so implementations should explicitly detect multiple survivors and use additional validation vectors when necessary.

7.3 Interpretation and Reproducibility

The experiments validate the algorithms under a deterministic, noiseless synthetic oracle. A 100% success rate is expected from Lemma 2; it is not a statistical estimate of success on a physical side channel. In this model $q = 1$ is theoretically sufficient, whereas the reported TDES implementation uses $q = 5$. We do not claim noise robustness without a calibrated noisy-oracle experiment. Future work should report success curves versus noise level, quantization precision, signal-to-noise ratio, and q, together with confidence intervals.

8 Discussion, Limitations, and Countermeasures

8.1 What the Attack Demonstrates

The experiments demonstrate a representation-level vulnerability. A natural continuous extension of a secure Boolean operation may reveal a secret through its off-domain slopes or intermediate activations. DES is especially convenient for illustrating the effect because its first nonlinear layer is preceded by a coordinate-wise XOR with a 48-bit round key, and PC-2 leaves only eight bits to complete.

The attack does not improve classical cryptanalysis of ordinary DES/TDES. It requires access unavailable from a conventional encryption API: nonstandard neural encodings, intermediate XOR values, and, for the implemented TDES completion, stage traces A and B. It also assumes exact knowledge of the network construction and no masking, quantization ambiguity, clipping, noise, or access control at the probed layer.

8.2 Countermeasures

The most direct countermeasure is domain enforcement: reject, quantize, or cryptographically authenticate inputs before the neural computation, ensuring that only canonical binary encodings reach secret-dependent layers. The intermediate activations must not be exposed through an API, debugging hook, or shared-memory interface. More generally, a neural implementation should use a construction whose security is defined over its entire accepted real domain, rather than relying only on agreement at Boolean vertices [8].

Additional hardening may include masking or secret sharing of intermediate values, randomized encodings, constant-interface execution, and leakage-aware testing. These techniques change the oracle and therefore require a new proof or experiment; merely adding inference noise is not a principled defense if an attacker can average repeated queries.

8.3 Validity Threats and Future Work

Internal validity is limited by the use of a synthetic noiseless oracle and software-generated traces. External validity is limited because no deployed neural cryptographic product or physical measurement channel was evaluated. Construct validity depends on the use of "linear time": throughout this paper it measures oracle-query growth, not total wall-clock time or classical DES security.

Future work should evaluate output-only attacks, minimize stage-trace access, model finite precision and activation quantization, and inject calibrated noise. A useful progression is to report success curves under successively weaker oracles and to test whether secure neural compilation eliminates the observed separability.

9 Related Work

Classical DES analysis established differential cryptanalysis [5], linear cryptanalysis [14], exhaustive-search feasibility, and implementation attacks such as timing analysis [11]. TDES extends key-search resistance but retains a 64-bit block, enabling collision-based attacks in high-volume protocols [4]. These attacks operate in models different from ours and do not rely on a neural realization.

Neural cryptanalysis has often meant learning an unknown cipher from binary input/output samples. Bellini et al. [3] explain why learning generic high-dimensional Boolean functions in a black-box setting requires exponential data and why positive experiments frequently concern reduced-round or otherwise structured targets. Our approach does not train a predictor and does not contradict those limitations: it exploits an explicitly constructed continuous representation and a stronger intermediate oracle.

The closest work is Deep Neural Cryptography by Gerault et al. [8], which distinguishes Boolean correctness from security against real-valued queries, demonstrates attacks on natural implementations, and develops secure compilation techniques. Table 3 makes the boundary of our contribution explicit. We do not claim the general representation-leakage idea as new. Instead, we specialize it to DES/TDES structure and carry the leakage through the key schedule to full external master-key recovery.

Relation to white-box cryptanalysis. Because our model exposes selected intermediate values, it is related to the adversarial setting of white-box cryptography. The seminal white-box AES work of Chow et al. [6] considers protecting embedded keys even when an adversary has extensive

Table 3. Relationship to the closest representation-leakage work.

Aspect	Gerault et al. [8]	This work
Core issue	Boolean vs. real-domain neural security	Same framework
Primitive focus	General cryptographic neural realizations	DES and two-key EDE TDES
Leakage target	Natural-representation behavior	Expansion/key-XOR activations
Cipher structure	General methodology	Feistel, E expansion, reversed TDES decryption
Key recovery	Framework-dependent	48-bit subkey $\rightarrow$ 56-bit DES key; two components $\rightarrow$ 112-bit TDES pair
Evaluation	General attacks/compilation	1,000 DES keys and 1,000 TDES pairs

implementation access. Our oracle can be viewed as a restricted neural analogue of such an interface: it exposes only specified neural quantities and does not grant unrestricted extraction or modification of all implementation components. The model therefore lies between ordinary black-box chosen-input cryptanalysis and unrestricted white-box implementation access. We deliberately do not cite LLM "white-box" training-data extraction as a survey of white-box cryptography because that is a different use of the term.

This viewpoint is also motivated by implementation-level leakage studies of neural systems. Recent work reports that physical execution of neural networks on edge devices can reveal private interactions through side channels [12]. Such physical leakage is different from our explicit synthetic oracle and does not validate our attack as a physical side channel; it only reinforces the broader need to treat internal neural computation as potentially security-relevant.

Other intersections between neural networks and cryptography pursue different goals. Adversarial neural cryptography trains communicating agents to learn encryption-like behavior [1,7]; neural synchronization has also been proposed for key exchange and subsequently cryptanalyzed [9,10]. Cryptographic techniques for secure neural inference protect model parameters and inputs during computation [13]. These directions are complementary to the representation-leakage question considered here.

10 Conclusion

We presented full effective-master-key recovery for natural ReLU realizations of DES and two-key EDE TDES under an explicit synthetic intermediate-state oracle. Paired perturbations reveal each bit of a selected 48-bit DES round key independently. Partial inversion of PC-2 then reduces each component master key to 256 candidates, which instrumented stage values resolve. The leakage query count grows linearly with the number of targeted subkey bits, and the supplied Python experiments recover all keys in 1,000 random DES trials and all key pairs in 1,000 random TDES trials.

The central conclusion is not that conventional DES and TDES have acquired a new linear-time attack in practice. The outcome is a DES/TDES-specific instantiation of the representation-leakage framework introduced in Deep Neural Cryptography [8], extended through the DES key schedule to effective master-key recovery. The demonstrated complexity claim is linear in leakage-query count, not a linear-time attack on the conventional ciphers. Rather, exact Boolean correctness does not guarantee security for a neural extension over real inputs or against exposure of hidden activations. A cryptographic DNN must define and enforce its accepted domain, protect intermediate values, and be evaluated in a security model that covers the complete neural interface.

References

1. Martin Abadi and David G. Andersen. Learning to protect communications with adversarial neural cryptography. arXiv:1610.06918, 2016.
2. Elaine Barker and Nicky Mouha. Recommendation for the triple data encryption algorithm (tdea) block cipher. Technical Report NIST SP 800-67 Revision 2, National Institute of Standards and Technology, 2017. Withdrawn January 1, 2024.
3. Enrico Bellini, Andreas Hambitzer, Maria Protopapa, and Massimiliano Rossi. Limitations of the use of neural networks in black box cryptanalysis. In *Information Technology and Communications Security*, Lecture Notes in Computer Science, pages 100–124. Springer, 2021.
4. Karthikeyan Bhargavan and Gaëtan Leurent. On the practical (in-)security of 64-bit block ciphers: Collision attacks on HTTP over TLS and OpenVPN. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 456–467. ACM, 2016.
5. Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. In *Advances in Cryptology — CRYPTO '90*, Lecture Notes in Computer Science, pages 2–21. Springer, 1991.
6. Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. White-box cryptography and an AES implementation. In *Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 250–270. Springer, 2002.

7. Marcelo Coutinho, Rodrigo Albuquerque, Felipe Borges, Luis García Villalba, and Tae Hoon Kim. Learning perfectly secure cryptography to protect communications with adversarial neural cryptography. *Sensors*, 18(5):1306, 2018.
8. David Gerault, Anna Hambitzer, Eyal Ronen, and Adi Shamir. Deep neural cryptography. Cryptology ePrint Archive, Paper 2025/288, 2025.
9. Ido Kanter, Wolfgang Kinzel, and Eytan Kanter. Secure exchange of information by synchronization of neural networks. *Europhysics Letters*, 57:141–147, 2002.
10. Alexander Klimov, Anton Mityagin, and Adi Shamir. Analysis of neural cryptography. In *Advances in Cryptology — ASIACRYPT 2002*, Lecture Notes in Computer Science, pages 288–298. Springer, 2002.
11. Paul C. Kocher. Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems. In *Advances in Cryptology — CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.
12. Zhuoran Liu, Senna van Hoek, Péter Horváth, Dirk Lauret, Xiaoyun Xu, and Lejla Batina. Real-world edge neural network implementations leak private interactions through physical side channel. *CoRR*, abs/2501.14512, 2025. arXiv:2501.14512.
13. Zoltán Á. Mann, Christian Weinert, Damien Chabal, and Joppe W. Bos. Towards practical secure neural network inference: The journey so far and the road ahead. *ACM Computing Surveys*, 56(5):1–37, 2023.
14. Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology — EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer, 1994.
15. National Institute of Standards and Technology. Data Encryption Standard (DES). Technical Report FIPS PUB 46-3, NIST, October 1999.