Supplementary File 1: Algorithms and Pseudocode

This supplementary file contains the full pseudocode for the four algorithms of the proposed Forensic-Aware Blockchain-Integrated Extended BiLSTM framework for real-time intrusion detection in healthcare IoT (IoMT) systems. These algorithms are referenced in the Protocol section of the main manuscript.

Algorithm 1: Data Preprocessing and Feature Engineering for IoMT Data

---

Input: Raw IoMT traffic stream R (records for all nodes), Set of IoMT nodes $N = \{n_1, n_2, \dots, n_K\}$
Per-feature stats from training, Window length T, stride $s(1 \le s \le T)$
Encoder $\phi(\cdot)$ for categorical fields, Small constant $\varepsilon>0$ for numerical stability

Output: Preprocessed, windowed packet stream $X = X^{(n_j)}$ ready for model ingestion

Begin:

X ← ∅
for each node $n_j \in$ N do
S ← ExtractStream(R, n_j) // raw records for node n_j
C ← ∅ // cleaned/normalized records for n_j
for i = 1 to |S| do
(x, c) ← SplitFeatures(S[i])
if IsCorrupt($S[i]$) then continue
for f=1 to d // missing-value imputation
if isNA($x[f]$) then $x[f] \leftarrow$ Mean$f(S)$
end for
x←Denoise(x) // e.g., median/EMA filter over time
for f=1 to d do // min-max normalization to [0, 1]
x'[f] ← (x[f] − minf) / (maxf − minf + ε)
end for
z←ϕ(c) // one-hot/label encoding
$x*$ ← Concat(x', z) // final feature vector
append $x*$ to C
end for
$X^{(n_j)} \leftarrow \emptyset$
for t = 1 to |C|−T+1 step s do // temporal segmentation
Wt ← [C[t], C[t+1], …, C[t+T−1]] // window ∈ $RT \times d'$
if SanityCheck(Wt)=true then append Wt to $X^{(n_j)}$
end for
append $X^{(n_j)}$ to X
end for
return X

End

Algorithm 2: Extended BiLSTM-Based Intrusion Detection

Input: Preprocessed packet stream $X = X^{(n_j)}$, Pretrained Extended BiLSTM model $M\theta$
Classification threshold τ ∈ [0,1]

Output: Predicted intrusion labels {Label$_i$} for each input window W_i

Begin:

O ← ∅ // initialize output list
for each window $W_i \in X$ do
// Step 1: Temporal Feature Extraction (Extension 1)
Apply 1-D convolution to capture short-term temporal patterns
$U \leftarrow \varphi(Conv1D(W_i)),\ U = \{u_1, u_2, \ldots, u_T\}$
// Step 2: Bidirectional LSTM Encoding (Core Model)
for t = 1 to T do
$\vec{h}_t = \text{LSTM}_f(u_t, \vec{h}_{t-1})$
$\overleftarrow{h}_t = \text{LSTM}_b(u_t, \overleftarrow{h}_{t+1})$
end for
for t = 1 to T do
$h_t = [\vec{h}_t; \overleftarrow{h}_t]$ // concatenate forward & backward states
end for
// Step 3: Residual Connection and Normalization (Extension–2)
for t = 1 to T do
$\tilde{h}_t = LayerNorm(h_t + u_t)$
end for
// Step 4: Attention Mechanism (Temporal Prioritization)
for t = 1 to T do
$\alpha_t = exp\left(W_a \tilde{h}_t\right) / \sum_{k=1}^{T} exp\ (W_a \tilde{h}_k)$
end for
c ← ∑αt $\tilde{h}_t$ // context vector
// Step 5: Dense layer & prediction
v ← Wc c + bc
ŷ ← σ(v) // sigmoid output in [0,1]
// Step 6: Classification decision
if ŷ ≥ τ then
Label$_i$ ← "Malicious"
else
Label$_i$ ← "Safe"
end if
append (W_i, Label$_i$) to O
end for
return O

End

Algorithm 3: Training Loop with Early Stopping

Input: Model $M\theta$; training set Dtrain; validation set Dval; batch size B; learning rate η;
Output: Trained parameters θ^*

Begin:

Initialize θ; $m\leftarrow 0$, $v\leftarrow 0$; bestVal$\leftarrow +\infty$; stall$\leftarrow 0$
for epoch = 1, 2, … do
Shuffle Dtrain and split into mini-batches $\{Bb\}$ of size B
for each batch Bb do
Forward pass (Alg. 2) $\Rightarrow$ predictions $\{\hat{y}\}$
Compute LWBCE on Bb
Backpropagate through time $\Rightarrow g$
Adam moments: m
Bias-correct: $\hat{\mathrm{m}}$
Update: θ
end for
Compute validation loss Lval on Dval
if Lval<bestVal then bestVal $\leftarrow L$val; save $\theta^* \leftarrow \theta$; stall $\leftarrow 0$
else stall $\leftarrow$ stall+1
if stall $\geq K$ then break
end for
return θ^*

End

Algorithm 4: Extended BiLSTM-Blockchain Intrusion Detection and Alert Mitigation

Input: Preprocessed packet stream $X = X^{(n_j)}$; Pretrained Extended BiLSTM model $M\theta$; classification threshold $\tau \in [0, 1]$; Blockchain ledger B(PoA); AuditSafe ∈ {0,1}

Output: Labels {Label$i$} and ledger updates for each window Wi

Begin:

O ← ∅ // output list of (Wi, Label$i$)
for each node nj and window $W_i \in X^{(n_j)}$ do
// Model inference
$\hat{y}$ ← Predict_Extended_BiLSTM(Mθ,Wi)
// Decision
if $\hat{y} \geq \tau$ then
Labeli ←"Malicious"; $\delta t \leftarrow 1$
// Block construction
Di ← ExtractEventMeta(W_i, nj)
Ti ← Now(); PrevHash ← Head(B)
Hi ← hash(Di ‖ Ti ‖ $\hat{y}$ ‖ PrevHash)
Sigi ← Sign(Hi)
Bi ← {Hi, Ti, Di, Sigi, PrevHash}; $B \leftarrow B \cup \{Bi\}$
// Mitigation via smart contract
TriggerContract(S(δt))
else
Labeli ←"Safe"; $\delta t \leftarrow 0$
if AuditSafe=1 then AppendLightTx(B, nj, Ti) end if
end if
Append (Wi, Label$i$) to O
end for
return O

End

Algorithm 5: AQU-IMF-RFE Feature Selection	
Input:	Preprocessed training data Z; class labels Y; target feature count k; AO population size N; maximum iterations MaxIter; AO control parameters; Random-Forest estimator; cross-validation folds = 10.
Output:	Optimal selected feature subset F*.
Begin:	

```
INPUT: Preprocessed training data Z (features x samples); class labels Y;
target feature count k; population size N; max iterations MaxIter;
AO control parameters; RF estimator; folds = 10.
OUTPUT: Ranked / selected feature subset F*.

// ---------- STAGE 1: Mutual-Information relevance scoring ----------
for each feature f in Z do
Compute MI(f, Y) = H(f) - H(f | Y)          // Eq. 4-6
where H(f)   = - SUM p(x) log p(x)
H(f|Y) = - SUM p(y) SUM p(x|y) log p(x|y)
end for
Rank features by descending MI; S <- features with highest MI    // Eq. 17

// ---------- STAGE 2: Aquila Optimizer refinement ----------
Initialize population Z_S of N binary feature-subset vectors from S  // Eq. 18
Evaluate fitness f(Z_i) for each agent; Z_best <- best agent
iter <- 1
while (iter <= MaxIter) do
Z_mean <- (1/N) * SUM_i Z_i                 // Eq. 10
for i = 1 to N do
if (iter <= (2/3)*MaxIter) and (rand < 0.5) then
// Expanded exploration (high soar, vertical stoop)
Z_i <- Z_best*(1 - iter/MaxIter) + (Z_mean - Z_best*rand)   // Eq. 9
else if (iter <= (2/3)*MaxIter) and (rand >= 0.5) then
// Narrowed exploration (contour flight, short glide)
Z_i <- Z_best*Levy(D) + Z_R + (y - x)*rand               // Eq. 11-13
else if (iter > (2/3)*MaxIter) and (rand < 0.5) then
// Expanded exploitation (low flight, slow descent)
Z_i <- alpha*(Z_best - Z_mean) + delta*((UB-LB)*rand+LB)    // Eq. 14
else
// Narrowed exploitation (walk and grab prey)
Z_i <- QF*Z_best - (G1*Z_i*rand) - (G2*Levy(D)) + rand*G1   // Eq. 15-16
end if
Binarize Z_i via transfer function; repair to valid subset
if f(Z_i) < f(Z_best) then Z_best <- Z_i      // minimization
end for
iter <- iter + 1
end while
S_AO <- feature subset encoded by Z_best
```

```
// ---------- STAGE 3: Recursive Feature Elimination (RFECV) ----------
F <- S_AO
repeat
acc <- 0
for fold = 1 to 10 do          // 10-fold cross-validation
split F into train (9 folds) and validation (1 fold)
train Random Forest on the training folds
acc <- acc + accuracy(validation fold)
record per-feature importance from the RF model
end for
acc <- acc / 10
rank features by averaged RF importance
remove the lowest-importance feature(s) from F
until |F| = k  OR  cross-validated accuracy stops improving

F* <- F       // final selected subset (|F*| = 14 / 24 / 16 per dataset)
return F*
```

End

Parameter	Value
Population size (N)	100
Maximum iterations (MaxIter)	10
Exploitation factor (α)	0.1
Learning rate (β)	0.1
Attraction factor (η)	0.005
RFE cross-validation folds	10
RFE estimator	Random Forest

Algorithm 6: recordIntrusion (core logging function)

Input: nodeId, patientId, attackClass, probabilityBp, dataDigest D_i, signature Sig_i, isolate flag

require caller is an authorized gateway

require probabilityBp ≤ 10000

$T_i \leftarrow$ current block timestamp

$H_i \leftarrow$ Keccak-256($D_i \parallel T_i \parallel$ probabilityBp $\parallel$ PrevHash)

recover signer from Sig_i over D_i (ecrecover); require signer == caller

append block (H_i, T_i, D_i, Sig_i, PrevHash, nodeId, patientId, attackClass, probability, isolate) to ledger

PrevHash $\leftarrow H_i$

emit BlockCreated; if isolate, emit NodeIsolated; emit AdminAlert

return H_i