Engineering and Technology Journal e-ISSN: 2456-3358

Volume 11 Issue 05 May-2026, Page No.- 10230-10241

DOI: 10.47191/etj/v11i05.62, I.F. – 8.595

Real-Time Prevention of Shoulder Surfing Attacks Through Object Detection

Farras Lathief [1], Rahmad Abdillah[2*], Novriyanto[3], Pizaini[4]

[1,2,3,4]Department of Informatics Engineering, Faculty of Science and Technology, Universitas Islam Negeri Sultan Syarif Kasim Riau, Indonesia

ABSTRACT: Shoulder surfing is a social engineering attack in which an observer covertly views a victim's device screen to steal sensitive information. Existing countermeasures, such as privacy films and screen-obfuscation software, operate passively and fail to definitively block observers at straight-on angles. This study proposes a real-time shoulder surfing prevention application for Android that integrates Google ML Kit Face Detection with the CameraX API and foreground service. The system detects multiple faces in the front camera field of view and triggers an automatic screen lock when more than one face satisfying four filtering conditions is detected: valid facial landmark structure, horizontal rotation angle below 36°, dynamic eye-open probability threshold adapted to ambient lighting, and a minimum bounding box area ratio for proximity validation. Testing on two devices with different hardware specifications across variations in observation distance (0.5–2 m) and lighting (bright and dim) confirmed the system's functional reliability. On the high-end device, the system achieved up to 17.21 FPS with a minimum response time of 74 ms, though this higher processing intensity resulted in greater battery consumption. Evaluation revealed that detection speed drops at extreme distances due to the loss of face tracking efficiency near the minimum detection threshold, and in low-light conditions due to hardware-level shutter speed constraints. Despite these hardware-dependent limitations, functional correctness was maintained on the mid-range device with response times consistently below 300 ms. Supplementary testing on disguised observer scenarios, including the use of a physical mask, sunglasses, and a face mask, further confirmed that the system successfully detected partially occluded faces as active threats, demonstrating robustness against deliberate concealment attempts. The results confirm that the system provides a successful, automated, on-device shoulder surfing prevention mechanism without requiring internet connectivity, cloud processing, or custom model training.

KEYWORDS: Shoulder Surfing, Face Detection, ML Kit, Privacy Protection, Real-Time Detection

I. INTRODUCTION

Information security on digital devices has become a primary concern as the use of smartphones and mobile devices in public spaces continues to grow. One frequently overlooked threat is shoulder surfing-the covert observation of a user's device screen to steal sensitive information such as passwords, PINs, and personal data [1]. This issue is serious given that nearly 1 in 5 adults in the United States report having been a victim of shoulder surfing, particularly among younger users [2].

Various defense mechanisms have been developed, including optical screen protectors (privacy films), shoulder-surfing-resistant authentication mechanisms, attacker detection systems, and software solutions. Anti-spy glass works by optically limiting the viewing angle, but remains vulnerable to observers directly in front of the screen and reduces visual comfort [3]. Authentication schemes such as IllusionPIN or keypad shuffle are designed to make PIN observation difficult in public. Software solutions focus on obscuring the screen display through blur, grayscale, or pixelation to make content difficult to read from common attack angles [4].

Recent studies confirm that shoulder surfing has become a real privacy threat. Attack success rates remain significant at distances of 1–2 meters and angles of 0° and 30° from the screen. Antispy glass is only effective at lateral angles of 30°–45° and fails to block observation from straight-on positions [4], [5]. Shoulder surfing has been identified as a primary gateway to threats such as identity theft and unauthorized device access, as evidenced by a surge in academic publications over the past five years [2], [6].

One significant system is Eye-Shield, a software-based solution employing a vision-based approach to suppress data leakage probability. Eye-Shield's protection is optimal up to 104 cm behind the user and at a minimum angle of 45 degrees, reducing shoulder surfing success rates to below 25% for images and 16% for text [4]. On the other hand, Google's ML Kit Face Detection offers real-time face detection without requiring manual model training or optimization. Prior research has demonstrated that integrating ML Kit with CNN can achieve 88.25% accuracy even under low-lighting conditions [7].

The key weakness of prior approaches is the lack of integration between real-time threat detection systems and automatic protection mechanisms on mobile devices. Many approaches remain passive and cannot close the attack gap from straight-on or over-the-shoulder directions [4], [5], [8].

In this context, ML Kit Face Detection as a native Google solution already integrated into Google Play Services makes face detection implementation more practical and efficient.

This research proposes the development of a shoulder surfing prevention application based on real-time face detection using ML Kit Face Detection on Android devices that can run in the background. The application leverages the ML Kit FaceDetector API for multiple face detection with low latency and no additional model training. Anti-spy glass [3] still functions as a passive protection layer at lateral angles of 30°–45°, while the proposed application closes the gap from straight-ahead and over-the-shoulder directions through an automatic screen-locking mechanism (auto-lock screen) when more than one face is detected in the front camera's field of view.

II. LITERATURE REVIEW

A. Shoulder Surfing

Shoulder surfing is an attack technique that exploits a victim's inattentiveness when entering sensitive information on digital devices in public, such as PINs, passwords, or other personal data. This technique falls under the social engineering category as it exploits human weakness as the weakest link in security systems [9]. Study [10] used a HAIS-Q approach to analyze mobile banking user behavior in Indonesia and found that password management and mobile device usage remain the main weak points exploited by social engineering actors, including shoulder surfing. The risk of attack increases with the mass use of smartphones in public spaces [1].

In terms of viewing angle and distance, perpetrators are generally within approximately 0.5–1.2 meters of the victim [6]. The PrivacyScout study by [5] modeled attack scenarios at four observation distances (50 cm, 100 cm, 150 cm, and 200 cm) with three angles (0°, 30°, and 60°), proving that attack success against text, photos, and PINs remains significant up to 1–2 meters at 0° and 30°. The 0.5–2 meters range and 0°–60° angle constitute the primary attack zone relevant for designing visually-based detection systems far[4], [5], [6].

B. Object Detection Concepts

Object detection is a fundamental task in Computer Vision aimed at identifying and locating objects in images or video. Before the era of deep learning, traditional algorithms such as SIFT, HOG, and Viola-Jones dominated the field but proved less adaptive to variations in scale, lighting, and background complexity [11], [12]. The Viola-Jones algorithm itself was implemented by [13] to detect shoulder surfing on ATM systems, demonstrating high accuracy at close range though limited to ATM scenarios.

A major revolution occurred since AlexNet (2012) popularized CNN and became the foundation for modern architectures such as R-CNN, Faster R-CNN (two-stage detectors) as well as YOLO and SSD (one-stage detectors) [11]. Although YOLO continues to evolve, its implementation on Android requires complex training, TensorFlow Lite conversion, and manual optimization [11], [14]. Research [15] integrates TensorFlow Lite and ML Kit for a face recognition-based attendance system running in real-time. Therefore, this study selects ML Kit Face Detection, which is already optimized for on-device face detection on Android without the overhead of custom model development.

C. ML Kit Face Detection API

ML Kit Face Detection is a mobile SDK provided by Google through Google Play Services to detect and track human faces in real-time with fully on-device processing, requiring no internet connection or cloud server [16]. Unlike traditional approaches requiring developers to train custom models, perform model format conversion, and optimize for specific hardware, ML Kit provides a pre-trained and pre-optimized model with a high-level API [16].

The main architectural components of the ML Kit Face Detection API are as follows [16]: (a) **Face Detection**: detects face presence and returns a bounding box and Euler angles (X, Y, Z); minimum face size of 100×100 pixels required. (b) **Face Tracking**: assigns a unique ID to track the same face across frames without identity-based face recognition. (c) **Landmark Detection**: estimates key facial points (eyes, nose, mouth, ears, cheeks), optional. (d) **Contour Detection**: generates 133 contour points representing facial feature shapes. (e) **Classification Module**: classifies eye openness and smiling probability with a confidence score between 0 and 1. (f) **FaceDetectorOptions**: provides configuration for minimum face size and enabling/disabling optional modules as a speed-accuracy trade-off. Several previous studies [8], [17], [18], [19] have demonstrated the reliability of ML Kit for biometric authentication, liveness detection, and attendance systems, but have not directed it toward shoulder surfing prevention.

Beyond the detection of facial features, ML Kit Face Detection also reports the orientation of each detected face through a set of Euler angles. Face orientation is expressed as three rotational axes relative to the camera, as defined in the official ML Kit documentation [16]: (a) **Euler X**: Represents the vertical tilt of the face. A positive Euler X angle indicates that the face is tilted upward, while a negative value indicates a downward tilt. (b) **Euler Y**: Represents the horizontal rotation of the face. A positive Euler Y angle indicates that the face is turned to the right relative to the camera, while a negative value indicates a turn to the left. (c) **Euler Z**: Represents the in-plane rotation of the face. A positive Euler Z angle indicates that the face is rotated counterclockwise relative to the camera's orientation.

D. Research Gap

Existing research generally focuses on recognizing the presence of an observer or modifying the screen display, rather than on detecting and automatically responding to conditions with more than one face as a concrete indicator of shoulder surfing. The ShouldAR system [20] requires special augmented reality hardware, making it impractical for ordinary smartphones. The Viola-Jones-based system [13] is

limited to ATM scenarios. Meanwhile, existing ML Kit research [7], [8], [15], [17], [18], [19] is mostly leveraged for authentication and attendance, and has not been directed toward shoulder surfing prevention with fully integrated auto-lock screen mechanisms.

The primary gap lies in the absence of integration between native SDK-based visual detection technology, such as ML Kit Face Detection, as a real-time and automated shoulder surfing prevention system. The current literature presents no solution that explicitly leverages multiple face detection capability and ML Kit performance configuration to trigger an instant auto-lock screen mechanism on Android. This research is proposed to fill this gap by integrating ML Kit Face Detection with the CameraX API and Android foreground service.

III.METHODOLOGY

A. Research Design

This study employs an experimental design focused on the development and testing of a real-time face detection-based shoulder surfing prevention system on Android devices. The system is designed by integrating ML Kit Face Detection as the on-device face detection module with the CameraX API as the image acquisition component, enabling the inference process to run efficiently on-device without reliance on server-based computation or custom model training.

The research design adopts an iterative Software Development Life Cycle (SDLC) approach encompassing the following stages: (1) Literature Study, (2) Requirements Analysis, (3) architecture design, (4) Implementation, (5) Testing & Evaluation.

B. Requirements Analysis

The system must satisfy the following functional and non-functional requirements. In terms of functional requirements, the system must: (a) support background/foreground service execution so that monitoring persists even when the app is not in the foreground; (b) request and verify camera access permissions per Android policies; (c) provide an auto-lock screen mechanism triggered when more than one face is detected, complying with Android standards (Device Administrator/Accessibility); (d) perform real-time face detection using the ML Kit FaceDetector configured with PERFORMANCE_MODE_ACCURATE; (e) provide a settings menu for users to enable or disable the detection service; and (f) safely terminate the service, release the camera, and free resources when the feature is disabled. In terms of non-functional requirements, the system must achieve a detection speed of at least 10–20 FPS, efficiently manage CPU, GPU, and memory to avoid excessive battery consumption, adhere to Android security and privacy principles, maintain minimum compatibility with Android 8 (SDK 26), sustain long-duration foreground service stability across various lighting conditions, and adopt a modular software architecture to facilitate configuration adjustments and future security feature additions.

C. System Architecture

The system consists of five interconnected main modules:

1. **Foreground Service Module**: Ensures the threat detection process runs consistently even when the application is not in the foreground. This module operates efficiently without displaying a direct interface to the user, complies with Android security and privacy principles, and keeps object detection and screen lock automation active while the user switches to other applications.

Foreground service Module

```
START Foreground_Service_Module
IF All_Application_Permissions_Granted THEN
INITIALIZE Ambient_Light_Sensor (Lux)

CREATE Notification_Channel (Priority = LOW)
CREATE Ongoing_Notification (Text =
"ScreenGuard Active")

RUN_AS_FOREGROUND_SERVICE
(Ongoing_Notification)

CALL Image_Acquisition_Module()
END IF
END
```

2. **Front Camera Image Acquisition Module**: Uses the device's front camera to obtain a real-time image stream through the CameraX API with the ImageAnalysis use case. The front camera is selected based on the characteristic of shoulder surfing attacks that typically occur when users directly interact with their screen in public spaces.

Image Acquisition Module

```
START Image_Acquisition_Module
  SELECT_CAMERA = Front_Camera
(DEFAULT_FRONT_CAMERA)
   SET_QUEUE_STRATEGY =
   "STRATEGY_KEEP_ONLY_LATEST"

   APPLY_SETTINGS_TO (CameraX_API)

ON New_Image_Frame_Received DO
Inference_Start_Time =
GET_CURRENT_SYSTEM_TIME()
CONVERT Image_Frame TO InputImage_Object

SEND InputImage_Object TO
Face_Processing_Module
 END ON
END
```

3. **Face Processing and Detection Module (ML Kit Face Detection)**: Frames from the acquisition module are converted into InputImage objects and processed using the ML Kit Face Detection API. The FaceDetector is configured with PERFORMANCE_MODE_ACCURATE, LANDMARK_MODE_ALL, setMinFaceSize(0.15f), and CLASSIFICATION_MODE_ALL. Detection focuses on identifying human faces per frame without identity-based face recognition.

Face Processing Module

```
START Face_Processing_Module
SET Face_Detector_Options:
Performance_Mode =
PERFORMANCE_MODE_ACCURATE
Landmark_Mode = LANDMARK_MODE_ALL
Minimum_Face_Size = 0.15f

Classification_Mode =
CLASSIFICATION_MODE_ALL
APPLY Face_Detector_Options TO
ML_Kit_FaceDetector

RECEIVE InputImage_Object FROM
Image_Acquisition_Module
PROCESS_IMAGE (InputImage_Object)
   IF Successful THEN
      RETURN Detected_Face_List TO
      Threat_Analysis_Module
   END IF
END
```

4. **Threat Analysis and Action Automation Module**: Receives the peopleFacingAtScreen output from ML Kit and counts the number of detected faces per frame. Filtering is applied through four conditions: (i) the face has valid landmark structure (nose or mouth), (ii) the horizontal rotation angle (headEulerAngleY) < 36°, (iii) a dynamic eye-open probability threshold adjusted by ambient light (0.4f for dark conditions < 50 Lux, and 0.8f for bright environments), and (iv) a minimum facial area ratio (> 0.005f) to ensure sufficient proximity. Only faces meeting all four criteria are counted as individuals actively facing the screen. If only one face is detected, the situation is treated as normal. If more than one face is detected, the auto-lock screen mechanism is triggered via the Device Administrator or Accessibility Service. An isLocking flag prevents duplicate triggers within a single event.

Threat Analysis Module

```
START Threat_Analysis_Module
Screen_Observer_Count = 0

FOR EACH Face IN Detected_Face_List DO
Valid_Structure = (Has_Nose) OR (Has_Mouth)

Straight_Angle =
(Absolute_Value(headEulerAngleY) < 36°)
// Dynamic Eye Threshold based on Ambient Light

IF (Current_Light_Level < 50) THEN
Eye_Threshold = 0.4f
ELSE
Eye_Threshold = 0.8f
END IF

Eyes_Open = (Left_Eye_Probability >
Eye_Threshold) OR (Right_Eye_Probability >
Eye_Threshold)
// Distance Filter
Face_Area_Ratio = (Face_Width * Face_Height) /
Frame_Area
Sufficient_Distance = (Face_Area_Ratio > 0.005f)


(Valid_Structure AND Straight_Angle AND
Eyes_Open AND Sufficient_Distance) THEN

Screen_Observer_Count = Screen_Observer_Count +
1
          END IF
END FOR

IF (Screen_Observer_Count > 1) AND
(Is_Locking EQUALS FALSE) THEN
Is_Locking = TRUE
Screenshot =
SAVE_CAMERA_FRAME_TO_JPEG(Camera_Fra
me)

  // Check User Preference
IF Prevention_Mode EQUALS
    "MODE_LOCK"
    THEN
XECUTE_IMMEDIATE_SCREEN_LOCK() //
 DevicePolicyManager.lockNow()
TERMINATE_SERVICE() // stopSelf()
  ELSE
    SHOW_WARNING_NOTIFICATION()
    WAIT 5 Seconds (Cooldown Period)
    Is_Locking = FALSE
  END IF

  CALL Threat_Inspection_Module()
END IF
END
```

5. **Threat Inspection Module:** Serves as a logging and review mechanism for each detected threat event. Every time a shoulder surfing condition is met, this module

records the detection timestamp, number of detected faces, system response time, and a front camera screenshot as visual evidence. All data is stored locally and accessible via the Threat Inspection feature interface.

	Threat Inspection Module
1	**START** Threat_Inspection_Module)
2	Completion_Time =
3	GET_CURRENT_SYSTEM_TIME()
4	Response_Time =
5	CalculateDifference(Inference_Start_Time,
6	Completion_Time)
7	Calculate_Average_FPS()
8	
9	**CREATE** Log_Data_Map:
10	Event_Time = Completion_Time
11	Face_Count = Observer_Count
12	Response_Speed = Response_Time
13	FPS = Average_FPS
14	Photo_Evidence = Screenshot
15	Light_Condition = Sensor_Lux_Value
16	
17	**CONVERT** Log_Data_Map TO JSON
18	SAVE JSON TO SharedPreferences
19	(PreferencesHelper)
20	**END**

D. Technology Stack

Table I. Technology Stack

Component	Technology	Justification
Application Platform	Android (min SDK 26)	Supports CameraX, ML Kit, and modern foreground services.
Programming Language	Kotlin	Official Android language; optimal integration with CameraX and ML Kit.
Camera Framework	CameraX	Front-camera acquisition and ImageAnalysis use case for real-time detection.
Face Detection Library	Google ML Kit Face Detection	On-device face detection without custom model training.
Background Execution	Foreground Service	Keeps detection process active while the app runs in the background.
Permission & Security	Runtime Permission, Device Admin / Accessibility	Manages camera permission and screen locking per Android policies.

E. Testing & Evaluation

The testing and evaluation phase is designed to verify that the system operates according to its functional specifications while meeting expected performance standards. This phase is divided into two categories:

1. Functional and Integration Testing

Functional testing is conducted using the black box method, focusing on observing system outputs based on given inputs without evaluating the internal code structure. Key test scenarios include: (a) verification of service activation and deactivation; (b) front camera access and face detection in foreground service; (c) threat detection logic (single face vs. multiple faces); (d) error handling such as permission failure or sudden service termination; and (e) Threat Inspection feature testing to verify that users can review threat events including timestamp, face count, and screenshots. Integration testing focuses on the cohesion of the main modules-CameraX, ML Kit Face Detection, threat analysis, foreground service, and Threat Inspection-observing frame capture synchronization, face filtering based on landmark criteria, headEulerAngleY, and eye-open probability, inter-module data flow, and system response time from face count change to device lock.

2. Performance Evaluation

Performance evaluation assesses the system's ability to detect and respond to shoulder surfing threats accurately and stably as a foreground service. Evaluation covers four aspects:

a) **Operational Performance and System Response**: Measured through per-frame detection latency and average FPS during service operation and the event is recorded in Threat Inspection.

b) **Device Resource Consumption**: Covering CPU usage, memory footprint, and battery drain over a 30-minute active service session. CPU and memory utilization are measured using the Android Studio Profiler, while isolated battery drain estimation is calculated using Android's built-in battery statistics utility.

c) **Lighting Condition Variation**:Ttesting under bright (>50 Lux) and dim (<50 Lux) conditions to assess consistency of ML Kit face detection and stability of threat decisions, particularly minimizing false negatives and false positives.

d) **Detection Distance Range**: Testing at distances ranging from 50 cm to 200 cm to validate face detection stability. This evaluates the system's spatial precision by combining the setMinFaceSize(0.15f) configuration with a custom bounding box area filter (faceAreaRatio > 0.005f). This ensures the system strictly monitors the primary shoulder surfing attack zone (0.5–2 meters) as identified in the literature [3], [4], [5].

IV. RESULTS AND DISCUSSION

A. Implementation Environment

Implementation and testing were conducted on two real Android devices with different front camera qualities, aiming to compare ML Kit Face Detection performance under

different image acquisition conditions. Table VII presents the full specifications of both devices.

Table II. Test Device Specifications

Spesifikasi	Perangkat A	Perangkat B
Device Name	POCO F4	Redmi 10
Operating System	Android 14	Android 11
RAM	8 GB	6 GB
Processor	Snapdragon 870	Snapdragon 680
Front Camera (MP)	20 MP	8 MP

B. User Interface

The interface consists of three main screens:

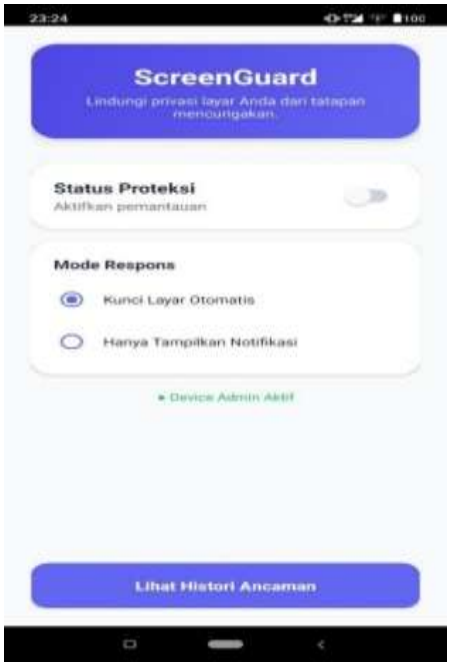

Figure 1. Main Screen (Dashboard)

Acts as the application's control center. On this screen, users can easily activate or deactivate security monitoring (Protection Status) and select their desired preventive action (Automatic Screen Lock or Notification Only). A quick access button to view the event history is also provided.

Figure 2. Threat History Screen

This screen displays a complete, chronologically ordered list of shoulder surfing event logs, from newest to oldest. Each record shows the time of the event and the number of faces detected. Users are provided options to delete records individually or clear the entire history at once.

Figure 3. Threat Detail Screen

This screen presents comprehensive details of a specific event selected from the history screen. Users can view photo evidence of the intruder's face, which can be enlarged (zoomable). Below the photo, event metrics are displayed, including the time, face count, room lighting levels at the time, response speed and average FPS.

C. Functional and Integration Testing Results

Black box functional testing was conducted simultaneously on both devices across eight scenarios. Table VIII presents the comparative functional and integration testing results.

Table III. Functional and Integration Testing Results

No	Description	Expected	Actual	Device A	Device B
1	User grants camera and device admin permissions, then activates detection.	Foreground Service is active in the background and notification appears in the status bar.	Service runs in the background and notification is active as designed.	Pass	Pass
2	User denies camera access permission when requested by the application.	System displays a warning and does not run the detection process.	Application detects the permission denial and prevents the detection feature from running until permission is granted.	Pass	Pass
3	User denies device admin permission.	System displays a notification that the auto-lock screen feature cannot function without the permission.	System detects the absence of admin permission and disables the automatic lock function.	Pass	Pass
4	Front camera captures a frame with a single user's face.	ML Kit detects one active face; system remains in safe status (screen not locked).	Detector recognizes one face; application does not trigger auto-lock.	Pass	Pass
5	A second face appears in the front camera's field of view (potential shoulder surfing).	ML Kit detects peopleFacingAtScreen > 1; triggers Device Admin to lock the screen instantly.	Device screen locks automatically immediately after the second face is detected.	Pass	Pass
6	A second face is within the camera's area but eyes are closed or face is turned away from screen.	Filter detects that headEulerAngleY or eyeOpenProbability parameter is not met; screen remains unlocked.	System ignores the face not actively facing the screen; screen remains on.	Pass	Pass
7	User accesses the "Threat Inspection" feature.	Displays a complete threat history list with screenshots and timestamps.	Threat data is stored and displayed accurately in the history interface.	Pass	Pass
8	User deactivates detection on the application.	System stops the foreground service, closes the camera, and releases resources.	Service terminates completely and camera access is released safely.	Pass	Pass

D. System Performance Evaluation

Table IV. Testing Device A Result

Distance	Lightning Condition	Avg FPS	Response Time (ms)	Status
0,5 m	Bright (> 50 Lux)	17,21	94 ms	Pass
1 m	Bright (> 50 Lux)	14,19	125 ms	Pass
1,5 m	Bright (> 50 Lux)	9,35	117 ms	Pass
2 m	Bright (> 50 Lux)	8,41	106 ms	Pass
0,5 m	Dim (< 50 Lux)	4,40	79 ms	Pass
1 m	Dim (< 50 Lux)	6,85	74 ms	Pass
1,5 m	Dim (< 50 Lux)	7,92	78 ms	Pass
2 m	Dim (<50 Lux)	8,70	75 ms	Pass

Table V. Testing Device B Result

Distance	Lightning Condition	Avg FPS	Response Time (ms)	Status
0,5 m	Bright (> 50 Lux)	4,17	250 ms	Pass
1 m	Bright (> 50 Lux)	1,64	210 ms	Pass
1,5 m	Bright (> 50 Lux)	2,19	227 ms	Pass
2 m	Bright (> 50 Lux)	1,29	223 ms	Pass
0,5 m	Dim (< 50 Lux)	2,66	220 ms	Pass
1 m	Dim (< 50 Lux)	1,10	247 ms	Pass
1,5 m	Dim (< 50 Lux)	1,31	233 ms	Pass
2 m	Dim (< 50 Lux)	1,31	214 ms	Pass

Table VI. Operational Performance Evaluation Results

Device	CPU Usage	Memory Consumption	Batery Usage (30 min)
A	14 – 16 %	150 – 160 mb 100 – 110 mb	9,79%
B	15 – 17 %	120 – 130 mb 110 – 120 mb	7,67%

The performance evaluation results presented in Tables IX, X, and XI demonstrate that observation distance, lighting condition, and device hardware specification each influence the system's detection speed (FPS) and response time. On Device A (POCO F4, Snapdragon 870, 20 MP front camera), the system achieved an average FPS ranging from 17.21 FPS at 0.5 m under bright conditions down to 8.41 FPS at 2 m under the same conditions. This decline occurs because as the observation distance increases, the face size in the frame decreases and approaches the minimum detection threshold of the setMinFaceSize(0.15f) configuration. At this marginal condition, the ML Kit module struggles to maintain stable face tracking, forcing the inference engine to perform computationally expensive full-frame re-detection on almost every frame. This intensive re-scanning greatly increases the computational load and decreases the average FPS. Under dim lighting conditions (<50 Lux), Device A recorded a notably lower FPS range of 4.40–8.70 FPS across all tested distances. This reduction in overall speed is primarily caused by a hardware-level optical constraint: in low-light environments, the camera's auto-exposure algorithm automatically decreases the shutter speed to absorb sufficient light, which inherently caps the maximum raw frame rate output of the camera sensor. Furthermore, the presence of visual noise in low-light images forces the ML Kit pipeline to process degraded inputs, slightly impeding classification efficiency. Despite this drop in FPS, the response time on Device A remained highly optimal, ranging from 74 ms to 125 ms across all scenarios, confirming that the auto-lock screen activates almost instantly when a second face is detected.

Device B (Redmi 10, Snapdragon 680, 8 MP front camera) recorded markedly lower FPS values across all test scenarios, ranging from 1.10 FPS to 4.17 FPS. The substantial performance gap relative to Device A is primarily attributable to the lower computational throughput of the Snapdragon 680 processor and the lower front camera resolution, which affects the quality of the InputImage object supplied to the ML Kit pipeline. Despite this, Device B successfully passed all test scenarios, demonstrating that the system remains functionally stable even on mid-range hardware. Response times on Device B ranged from 210 ms to 250 ms-slower than Device A but consistently below 300 ms, which is still acceptable for practical shoulder surfing prevention. The results confirm that both distance and lighting condition systematically affect FPS and response time on both devices. Lighting conditions further modulate detection stability through the dynamic eye-open probability threshold (0.4f under dim, 0.8f under bright conditions), which reduces false negatives in low-light scenarios.

Regarding resource consumption (Table XI), Device A consumed 14–16% CPU and 150–160 MB memory when the application was open in the foreground, dropping to 100–110 MB when running as a foreground service. Device B consumed 15–17% CPU and 120–130 MB in the foreground, with background memory usage at 110–120 MB. The high average FPS and low response time on Device A indicate that

its processor handles more face detection frames per unit of time compared to Device B. This higher processing intensity causes Device A's processor to remain more active during the testing session, thereby consuming more battery power (9.79%) than Device B (7.67%) over a 30-minute active session. These results indicate that the system operates within acceptable resource boundaries for a continuously running foreground service on Android.

E. Discussion

The overall results confirm that the proposed system successfully fulfills its core objective: providing real-time, automated shoulder surfing prevention on Android devices through ML Kit Face Detection integrated with a CameraX-based foreground service. All eight functional and integration test scenarios passed on both Device A and Device B, demonstrating that the system's modular architecture-comprising the Foreground Service, Image Acquisition, Face Processing, Threat Analysis, and Threat Inspection modules-operates cohesively across different hardware configurations. This cross-device consistency represents a meaningful practical contribution, as it indicates that the system is viable for deployment beyond high-end devices and remains accessible to users with mid-range smartphones.

Compared to prior work, the system addresses several gaps that earlier. Eye-Shield [4], one of the most closely related systems, provides software-based protection effective up to 104 cm and reduces shoulder surfing success rates to below 25% for images and 16% for text. However, as detailed in its methodology, Eye-Shield utilizes the front camera solely to track the legitimate user's eyes to continuously obfuscate the screen content based on binocular disparity. It functions as a passive screen-content obfuscation layer rather than an active threat-response mechanism; it does not detect the presence of an observer nor trigger any automated device lockdown. The proposed system operates at a complementary and more definitive security layer: it actively tracks any unauthorized face entering the primary shoulder surfing attack zone (0.5–2 m) and responds with an automatic screen lock within 74–125 ms on Device A and 210–250 ms on Device B, completely removing attacker access to the device content. PrivacyScout [5] by Bâce et al. characterized the attack zone at distances of 50–200 cm and angles of 0–60°, confirming that passive optical solutions fail at straight-on (0°) and near (50–100 cm) positions. The present system directly addresses this gap by combining headEulerAngleY filtering (rejecting faces rotated beyond 36°) with proximity filtering (faceAreaRatio > 0.005f), ensuring detection is focused precisely on the threat zone identified by PrivacyScout. Unlike the Viola-Jones ATM system [13], which is constrained to a single fixed scenario, and the ShouldAR system [20], which requires specialized augmented reality hardware, the proposed system runs entirely on a standard Android smartphone without additional hardware, requiring no internet connection, no cloud processing, and no custom model training.

Additional testing was conducted to evaluate the system's robustness against disguised observers, representing a realistic threat scenario in which a shoulder surfer deliberately conceals their identity through the use of sunglasses, a face mask, or a physical mask. Figure 4 illustrate the system's Threat Detail screen captured during the disguised observer test scenarios.

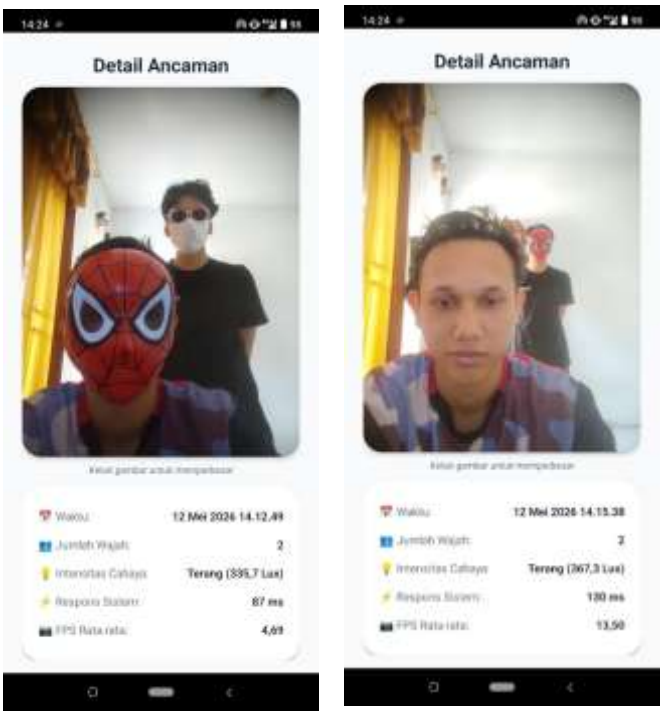

Figure 4. Threat Detail Result Disguised Observer Test Scenarios.

In tested scenarios, the system successfully identified the disguised face as an active threat and triggered the auto-lock screen mechanism accordingly. These results indicate that the system's detection reliability is not contingent upon complete facial visibility or the presence of unobstructed facial features. The ML Kit Face Detection pipeline, operating in conjunction with the four-condition filtering logic comprising facial landmark validation, headEulerAngleY thresholding, dynamic eye-open probability assessment, and bounding box area ratio verification, demonstrated sufficient discriminative capability to classify a partially occluded face as a valid screen-facing observer. These findings further substantiate the practical applicability of the proposed system in real-world public environments, where malicious observers may attempt to circumvent detection through deliberate physical concealment.

From a practical standpoint, the system is particularly well-suited for use during privacy-sensitive activities in public spaces, including accessing mobile banking applications, social media accounts, messaging platforms, and other applications containing personal or confidential data. The auto-lock screen mechanism ensures that even if a user is momentarily inattentive, the device is secured the instant a second face is detected in the camera's field of view. The dynamic eye-open probability threshold further improves robustness by adapting detection sensitivity to ambient lighting, reducing false negatives in dim environments such

as cafes, public transport, or night-time outdoor settings where shoulder surfing risk remains high.

F. Limitations

Despite the positive results, several limitations of the system were identified during evaluation. First, detection performance is heavily constrained by extreme observation distances. When an observer is at the maximum limit of the threat zone, the face area approaches the minimum detection threshold (setMinFaceSize(0.15f)). This marginal condition causes the ML Kit pipeline to lose its tracking efficiency, forcing it to continuously execute computationally expensive full-frame re-detections. Consequently, the frame rate drops below the ideal 10–20 FPS non-functional requirement. While the system still functions correctly, this reduced frame rate limits its ability to capture rapidly approaching observers or transient shoulder surfing events that last for only a fraction of a second.

Second, lighting condition introduces a performance trade-off that the dynamic eye-open threshold partially mitigates but does not fully resolve. Under very dim conditions (approaching 0 Lux or complete darkness), the front camera's inability to produce a clear image may cause ML Kit to fail to detect faces altogether, resulting in false negatives regardless of threshold settings.

Third, the system's detection reliability is dependent on the computational throughput and front camera quality of the host device. Devices with lower processors or inferior low-light performance yield reduced FPS and potentially higher rates of missed detections at greater distances. These hardware-dependent limitations suggest that the system's overall performance is partially gated by the device tier and that a minimum hardware specification should be recommended for optimal deployment.

Fourth, this study is limited in scope to detecting the general presence of a face within the camera's field of view. The system does not evaluate individual identity, classify facial attributes such as age or gender, or perform liveness detection to distinguish a real person from a photograph or screen. These capabilities were intentionally excluded from the current implementation, as the primary objective is to detect whether an additional person is present, not to identify who that person is.

Fifth, the system's front-facing camera architecture inherently covers only the straight-on and over-the-shoulder attack vectors. Observers approaching from extreme lateral angles (approximately 30°–45° from the side) may fall outside the camera's field of view and thus evade face detection entirely. In such cases, an anti-spy screen protector remains a relevant passive protection layer: its micro-louver optical film physically blocks viewing angles beyond approximately 30°–45° from the perpendicular, precisely the angular range that the proposed application cannot cover. These two mechanisms are therefore complementary rather than mutually exclusive. The anti-spy glass addresses lateral shoulder surfing through passive optical restriction, while the proposed application closes the critical gap at straight-on (0°) and near over-the-shoulder angles through active face detection and automatic screen locking. Users seeking maximum protection in high-risk public environments are advised to deploy both solutions concurrently.

V. CONCLUSIONS

This study has successfully developed and evaluated a real-time shoulder surfing prevention system for Android based on ML Kit Face Detection integrated with the CameraX API and a persistent foreground service. The system operates entirely on-device without requiring an internet connection, cloud processing, or custom model training, making it immediately deployable on any Android device running SDK 26 or higher. All eight functional and integration test scenarios passed on both test devices, confirming the correctness and stability of the modular architecture across hardware configurations ranging from a high-end Snapdragon 870 device to a mid-range Snapdragon 680 device.

Performance evaluation demonstrated that observation distance and lighting condition jointly influence detection speed and response time, and that device hardware specification is the primary determinant of overall FPS. Specifically, the drop in FPS at greater distances is attributed to the loss of face tracking efficiency near the minimum detection threshold, triggering computationally heavy full-frame re-detections. Similarly, low-light conditions inherently cap the raw frame rate due to longer camera exposure times. Despite these constraints, on Device A, the system achieved up to 17.21 FPS with a minimum response time of 74 ms, satisfying the real-time requirement for shoulder surfing prevention. On Device B, functional correctness was maintained across all scenarios with response times consistently below 300 ms, confirming that the system remains practically viable even on mid-range hardware. The four-condition face filtering logic-combining facial landmark validation, headEulerAngleY thresholding, dynamic eye-open probability, and bounding box area ratio-proved successful in reducing false positives from non-threatening faces and false negatives in low-light environments. These results collectively demonstrate that the proposed system offers a robust and practical alternative to existing passive protection methods, optimally utilizing native on-device machine learning for automated privacy security without requiring additional hardware.

Additionally, supplementary testing conducted on disguised observer scenarios, encompassing the use of a physical mask, sunglasses, and a face mask, confirmed that the system's detection capability is not limited to unobstructed faces. In all tested scenarios, the system successfully identified the disguised face as an active threat and triggered the auto-lock screen mechanism accordingly, demonstrating that the four-condition filtering logic retains sufficient discriminative capability even under conditions of partial facial occlusion.

For future work, four directions are recommended. First, the system should be tested on a wider range of Android devices

with different processor types and front camera specifications to determine the minimum hardware requirements needed for reliable performance. Second, the face detection pipeline could be extended with a face recognition feature to distinguish between the device owner and unknown individuals, allowing the system to respond more precisely rather than simply counting the number of faces detected. Third, the detection speed on mid-range and lower-end devices could be improved by allowing the system to automatically dynamically adjust its ML Kit processing mode (PERFORMANCE_MODE_FAST vs. PERFORMANCE_MODE_ACCURATE) based on the device's hardware capability, which would make the system more practical for a broader range of users without sacrificing baseline security. Fourth, the system's reliability could be further enhanced by incorporating eye gaze detection to determine whether an observer is actively looking at the screen, providing a more precise threat assessment and further minimizing false positives.

ACKNOWLEDGMENT

The authors would like to thank the Department of Informatics Engineering, Faculty of Science and Technology, Universitas Islam Negeri Sultan Syarif Kasim Riau, for the academic support provided throughout this research. The source code of the proposed system is publicly available at https://github.com/Farrasl/ScreenGuard.git.

REFERENCES

1. L. Bošnjak and B. Brumen, "Shoulder surfing experiments: A systematic literature review," Comput. Secur., vol. 99, Dec. 2020, doi: https://doi.org/10.1016/j.cose.2020.102023.
2. H. Farzand, S. Macdonald, K. Marky, and M. Khamis, "'What you think is private is no longer' -- Investigating the Aftermath of Shoulder Surfing on Smartphones in Everyday Life through the Eyes of the Victims," Nov. 2024, doi: https://doi.org/10.48550/arXiv.2411.18265.
3. C. Bassinet, D. Michael, R. Yoann, and W. Clemens, "Mobile phone screen protector glass: A TL investigation of the intrinsic background signal," Frontiers. Public Health, vol. 10, pp. 1–9, Sep. 2022, doi: https://doi.org/10.3389/fpubh.2022.969330.
4. B. J. Tang and K. G. Shin, Eye-Shield: Real-Time Protection of Mobile Device Screen Information from Shoulder Surfing. Anaheim, CA: USENIX Association, 2023. Accessed: Nov. 22, 2025. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/tang
5. M. Bâce, A. Saad, M. Khamis, S. Schneegass, and A. Bulling, "PrivacyScout: Assessing Vulnerability to Shoulder Surfing on Mobile Devices," Proceedings on Privacy Enhancing Technologies, vol. 2022, no. 3, pp. 650–669, Mar. 2022, doi: https://doi.org/10.56553/popets-2022-0090.
6. H. Farzand, "Understanding Shoulder Surfing & Informing the Design of Protection Mechanisms," Thesis (PhD), University of Glasgow, Glasgow, 2025. doi: https://doi.org/10.5525/gla.thesis.85064.
7. F. Fauzi Abdullah and S. Agustin, "Penerapan Biometric Face Recognition Menggunakan Metode Convolutional Neural Network pada Aplikasai Berbasis Android," INDEXIA : Informatic and Computational Intelligent Journal, vol. 6, no. 1, pp. 1–11, May 2024, doi: https://doi.org/10.30587/indexia.v6i1.4958.
8. B. Hartanto, B. Wiryawan Yudanto, and Kustanto, "Implementasi Google ML Kit Untuk Liveness Detection Dalam Sistem Face Recognition: Analisis Kinerja dan Keamanan Pada Aplikasi Mobile," Biner : Jurnal Ilmiah Informatika dan Komputer, vol. 4, no. 1, pp. 32–38, Jan. 2025, doi: https://doi.org/10.32699/biner.v4i1.8870.
9. E. M. Safitri, Z. Ameilindra, and R. Yulianti, "Analisis Teknik Social Engineering Sebagai Ancaman Dalam Keamanan Sistem Informasi: Studi Literatur," JIFTI-Jurnal Ilmiah Teknologi Informasi dan Robotika, vol. 2, no. 2, Dec. 2020, doi: https://doi.org/10.33005/jifti.v2i2.26.
10. M. Anastasiah and H. Pandia, "Analisis Perilaku Pengguna Mobile Banking Terhadap Keamanan Informasi Menggunakan Metode Human Aspects of Information Security Questionnaire (HAIS-Q)," Journal Of Social Science Research, vol. 4, no. 2, 2024, doi: https://doi.org/10.31004/innovative.v4i2.9684.
11. M. Jamali, P. Davidsson, R. Khoshkangini, M. G. Ljungqvist, and R. C. Mihailescu, "Context in object detection: a systematic literature review," Artif. Intell. Rev., vol. 58, no. 6, Jun. 2025, doi: 10.1007/s10462-025-11186-x.
12. P. Tsirtsakis, G. Zacharis, G. S. Maraslidis, and G. F. Fragulis, "Deep learning for object recognition: A comprehensive review of models and algorithms," International Journal of Cognitive Computing in Engineering, vol. 6, pp. 298–312, Dec. 2025, doi: 10.1016/j.ijcce.2025.01.004.
13. G. Chinnappa, R. Rajasekaran, S. Sekar, S. Kumar Ashokkumar, S. Ravikumar, and L. Shanmugam, "ATM Shoulder Surfing Detection Using Viola-Jones Algorithm," TIJER, vol. 11, no. 7, Jul. 2024, Accessed: Nov. 23, 2025. [Online]. Available: https://tijer.org/tijer/papers/TIJERC001306.pdf
14. firebase.google.com, "Face Detection | ML Kit for Firebase." Accessed: Dec. 08, 2025. [Online]. Available: https://firebase.google.com/docs/ml-kit/detect-faces
15. F. Abdillah Ahmad and N. Pratiwi, "Implementation of Face Recognition, Attendance Detection, and

Geolocation using TensorFlow Lite and Google ML Kit in a Mobile Attendance Application," Sistemasi: Jurnal Sistem Informasi, vol. 14, no. 1, pp. 172–186, 2025, [Online]. Available: http://sistemasi.ftik.unisi.ac.id

16. developers.google.com, "Face detection | ML Kit | Google for Developers." Accessed: Dec. 05, 2025. [Online]. Available: https://developers.google.com/ml-kit/vision/face-detection
17. O. Nachmani, T. Saun, M. Huynh, C. R. Forrest, and M. McRae, "'Facekit'-Toward an Automated Facial Analysis App Using a Machine Learning–Derived Facial Recognition Algorithm," Plastic Surgery, vol. 31, no. 4, pp. 321–329, Nov. 2023, doi: 10.1177/22925503211073843.
18. I. Saputra, T. Desyani, and A. Syaripudin, "Implementasi Login Aplikasi dengan Fitur Autentikasi Pengguna Menggunakan Flutter dan ML Kit Face Recognation," JORAPI : Journal of Research and Publication Innovation, vol. 3, no. 1, pp. 106–115, Jan. 2025, Accessed: Dec. 06, 2025. [Online]. Available: https://jurnal.portalpublikasi.id/index.php/JORAPI/article/view/1358
19. B. B. Wibowo and E. B. Setiawan, "Implementasi Face Recognition dan Geolocation Pada Sistem Presensi Karyawan Berbasis Mobile APPS," KOMPUTA : Jurnal Ilmiah Komputer dan Informatika, vol. 13, no. 1, Apr. 2024.
20. M. Corbett, B. David-John, J. Shang, and J. I. Bo, "ShouldAR: Detecting Shoulder Surfing Attacks Using Multimodal Eye Tracking and Augmented Reality," Proc. ACM Interact. Mob. Wearable Ubiquitous Technol., vol. 8, no. 3, Sep. 2024, doi: 10.1145/3678573.