Computers & Security 157 (2025) 104565

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Full length article

VeracOS: An operating system extension for the veracity of files

Naser AlDuaij

Department of Computer Science, Kuwait University, Kuwait

ARTICLE INFO

Keywords:
Mobile and ubiquitous computing
Operating systems
Systems security
Internet of Things
Other services and applications topics
File authentication
Generative artificial intelligence
Deepfake

ABSTRACT

As generative artificial intelligence has improved, there is a growing trend of generating false media for spreading misinformation, driving propaganda, and theft through enhanced social engineering. This creates a global concern, leading to a heavy demand for verification and fact-checking of information. Existing solutions aim at educating users or using artificial intelligence to fact-check and detect false documents or media. While these methods provide a measure for combating misinformation, many of these existing methods are inaccurate. Methods such as deepfake detection for videos are an uphill battle as deepfake generation keeps improving and newer methods are created to subvert deepfake detection techniques. VeracOS is introduced and presented as an operating system modification that is easily deployed, can certify files that are created, and ensures that any user can automatically check the authenticity of files across any existing application or platform. VeracOS invents a unique algorithm for certifying and verifying files. VeracOS aims to revolutionize the war against misinformation and exploitation of fake content by introducing several key features: VeracOS allows users or corporations to easily and automatically certify their media. Unlike existing solutions, VeracOS avoids intensive computations, specialized hardware, and private data sharing. VeracOS also allows any user to automatically be notified if the file they are viewing is verified to be authentic. VeracOS does not require the modification of existing applications nor does it require the sharing of private information such as what files or media are being viewed by a user. These key features provide a highly portable and easily deployed system for users of any operating system, including Internet of Things devices and mobile operating systems. Using media files such as images and videos as exemplary file types and using Android as an exemplary operating system, a VeracOS prototype was implemented to allow any user to automatically certify or verify their media files. The results show that VeracOS is easy to use and can be easily run on smartphones without the need for specialized systems, applications, or hardware.

1. Introduction

With the rapid advancements in generative artificial intelligence, computer graphics, and computing power, the ability to generate media based on existing media has never been easier (Zewe, 2023; Ray, 2023; Google LLC, 2024g). This has led to the rampant spread of generated media such as images, audio, and videos that are not only fake but in many cases representative of the likeness of other individuals (Juefei-Xu et al., 2022; KPMG, 2024; GZERO Staff, 2023). These generated and realistic media are often referred to as deepfakes (MIT Media Lab, 2024), a coined word that is a portmanteau of the machine learning method called "deep learning" and the word "fake" (Merriam-Webster, 2024). Global concerns have been raised regarding deepfake media as they can be used to manipulate or deceive people (Philmlee, 2023; Bartz, 2023; Jackson, 2023; Toews, 2020; Hsu, 2023; O'Sullivan, 2019; ETtech, 2023; Department of Homeland Security, 2024; eSafetyCommissioner - Australian Government, 2024; Britt, 2023).

To illustrate the gravity of these global concerns, consider these four real scenarios that were reported: (1) A finance employee at a multinational firm pays out 25 million U.S. dollars after a video call with a deepfake "chief financial officer" (Chen and Magramo, 2024). (2) Scammers using deepfake videos for extortion (Kan, 2023) such as a case of extorting a senior citizen using a deepfake video of a retired officer (Pandey, 2023). (3) An explicit deepfake video of a teenager going viral (Wong, 2024). (4) A deepfake video of a politician that may sway voters such as a video of the vice president of the United States of America showing her speaking unintelligibly going viral (Matt Novak, 2023). These are all examples of deepfake media causing privacy violations, spreading misinformation, personal harm, legal issues, ethical concerns, and driving propaganda (Philmlee, 2023).

To combat extortion, blackmail, misinformation, and any malice from deepfake media, several governments have introduced laws to classify deepfake media used for malicious reasons as criminal offences (US Congress, 2024a,b; California Legislative Information, 2024;

E-mail address: naser.alduaij@ku.edu.kw.

https://doi.org/10.1016/j.cose.2025.104565
Received 23 March 2025; Received in revised form 9 May 2025; Accepted 6 June 2025
Available online 12 July 2025

Reuters, 2024). To detect these deepfake media, several initiatives have been launched by governments, corporations, and academia. For example, the Defense Advanced Research Projects Agency (DARPA) has funded a project to allow individuals to compete for significant prizes by creating tools to detect deepfake media (Knight, 2018). Many corporations have also taken active measures to handle deepfake media on their platforms (X. Corp., 2024; Paul, 2020; Meta, 2024b). Some academics and corporations such as Amazon, Facebook, and Microsoft have partnered to also create a deepfake detection challenge for competitors to provide the best deepfake media detection tool (Kaggle, 2024). Academic research has also been conducted on the detection of deepfake media (MIT Media Lab, 2024; Groh et al., 2022). Additionally, global awareness of deepfake media has been focused on educating and training users to identify deepfake media (Philmlee, 2023; Groh et al., 2022). This shows a growing trend and focus towards deterring the creation of deepfake media and also the detection of deepfake media.

Deepfake media detection can be costly in terms of computation and time (Barnaciak and Ross, 2022), and generally requires training and learning effort from users who are also presumed to be technical users (Groh et al., 2022). In many cases, these detection methods or tools require sharing of the media with third parties or the cloud as they are provided through mobile applications or websites, creating privacy concerns for users (Peras and Mekovec, 2022; Soveizi et al., 2023). Even when these detection techniques are run locally, they are not fully accurate and might result in false negatives or false positives (Barnaciak and Ross, 2022). Additionally, newer detection technologies are constantly being subverted by newer technologies for deepfake creation (Barnaciak and Ross, 2022).

Instead of examining media and detecting which ones are deepfake and which ones are authentic, one can avoid deepfake media by providing the media through an authorized outlet such as a website. For example, news outlets can release their media for users to view them directly on their websites. However, many media items are forwarded and shared via E-mail or through social media instead and not through authorized websites (Belle Wong, 2023). For individuals, a media author can post their media directly on their own website and provide it with a secure cryptographic hash function value for the media itself. Users can compute the hash value of the media item and compare it with the one provided by the author. However, this is a tedious process for the author, who must create separate hash values for every media item shared, and the user, to verify every media item with every unique hash value. Note that many users might not be technical enough to understand this process. More importantly, a large number of media items are forwarded or shared via E-mail or through social media and not downloaded by the users directly from the author or source (Belle Wong, 2023).

VeracOS addresses these issues stemming from deepfake media by introducing operating system modifications to produce and verify verifiable media. VeracOS can be added to any system that can create images, audio, or videos. Specifically, any system that creates media or supports a camera or microphone would benefit from VeracOS modifications. For every created media item, VeracOS automatically signs the media item with a fixed unique and private author or creator signature and appends the value to the end of the media item file. Note that VeracOS does not introduce new media formats, does not modify existing media formats, and still maintains backward compatibility for media viewers and players. For users receiving a media item, VeracOS can automatically verify the authenticity of the media item by verifying the signature in the media file. Users only require a single unique author or creator signature, not a separate signature per media item. VeracOS works by utilizing asymmetric cryptography and secure hash computing algorithms.

VeracOS is, therefore, the first viable system and potential solution to the global concerns of deepfake media, to automatically verify the authenticity of media items against deepfakes by relying on existing and non-computationally intensive technologies and algorithms. VeracOS does not rely on machine learning or computing-intensive methods that are not fully accurate, does not require uploading of media items, does not require user training, and does not require the introduction of new media formats or standards.

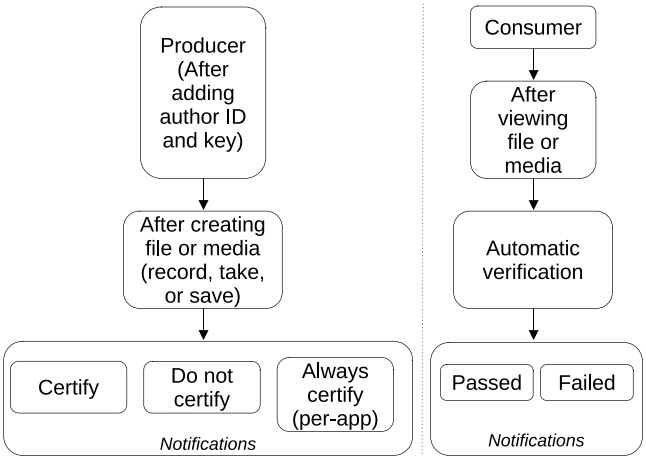

Fig. 1. VeracOS producer and consumer flowcharts.

The contribution of VeracOS is to provide an operating system extension, which can produce files, and the ability to produce truly verifiable files. VeracOS specifically uses media files as exemplary files. Additionally, the VeracOS contribution is also to provide an operating system extension or application with the ability to automatically verify files, such as media files, and alert the user if the item is not authentic and thus, potentially a fake item, such as a deepfake video. VeracOS tackles the issue of deepfake media by, instead, focusing on providing a novel ecosystem that provides authenticity verification for viewers without requiring specialized hardware or hardware modifications, application modifications, or sharing of files and their metadata. VeracOS is the first operating systems extension to facilitate this novel ecosystem. Additionally, VeracOS does not require new file formats as it maintains backward compatibility, and shows negligible overhead, for images and audio, and acceptable overhead for high-resolution videos.

A VeracOS prototype was implemented by modifying the Android operating system. Media was created using the default unmodified Android camera application and then viewed by the widely popular and unmodified VLC media player (Videolabs, 2024) and the default unmodified Android Gallery application. The rest of the paper is organized as follows: Section 2 discusses usage model, Section 3 discusses threat model, Section 4 discusses related work, Section 5 discusses background, Section 6 discusses architecture, Section 7 discusses implementation, Section 8 discusses evaluation and results, and finally, Section 9 discusses the conclusion and future work.

2. Usage model

VeracOS is designed to be simple to use and easily deployed. VeracOS framework is efficient, scalable, and can be easily adopted in any operating system. A media *producer* is any digital system that creates media or contains a camera or microphone. A media producer can produce images, video, or audio. These systems can be smartphones, tablets, laptops, desktops, digital cameras, digital recorders, Internet of Things devices, or even server farms that post-process media. A media *consumer* is any system or device that views images or plays video or audio. A producer may also be a consumer at the same time. An author or creator of media can be an individual or it can be a group of individuals sharing the same authorship, such as a corporation or agency.

Fig. 1 shows a flowchart for both the producer and consumer when creating a file and viewing a file, respectively. Users of producer systems need to run a VeracOS modified operating system and provide an author or creator secret key that has to be unique based on identity but does not have to be unique across multiple systems owned by the same author or creator. For example, a company may use the same secret key for all of its producer systems even if they use several different systems to produce media. The secret key may be provided as a file or through a Uniform Resource Locator (URL) link. This signature is used to sign the produced media; signing can also be postponed as it does not have to be performed immediately upon the creation of media. The signature used has to be unique and private to the entity, whether an individual or a company.

Users of consumer systems can either use the modified VeracOS operating system or download an application. For example, if the user is using a modified Android operating system that deploys VeracOS, the user does not need to do anything to verify media. Any media viewed will be automatically verified. If the media cannot be verified or was never signed by VeracOS, an alert is provided to the user regarding the currently viewed media. The user may ignore the alert. VeracOS does not break backward compatibility or prevent the consumer from viewing media regardless of its authenticity. If users are not running a VeracOS operating system, a VeracOS application may be downloaded and installed instead; this application does not require any operating system modifications. However, the user must manually provide the media to the application for verification purposes. In all cases of VeracOS, the media or any metadata is never uploaded or shared with any third party.

VeracOS uses a digital signature to verify authors and creators of media content. As such, VeracOS producers use an author or creator secret key to sign any media created. VeracOS consumer systems would automatically use the known author or creator public key to verify the authenticity of the signed media. The exact details of this algorithm and how it is performed automatically on a creator basis rather than a specific media item basis are discussed in detail in Section 6. Note that for strictly offline systems, users may add the public keys of authors and creators as individual files or through local URLs. Alternatively, a database file with all known author ID and public key tuples may also be provided as the entirety of the verification process can be fully performed offline. Using VeracOS, existing applications do not need to be modified.

3. Threat model

The threat model under VeracOS is concerned with the producer and consumer components. For the producer, there are several threats that exist. First, an adversary may intercept and modify image, audio, or video frames before certification. This issue persists beyond VeracOS and is not facilitated by VeracOS. Second, an adversary may gain access to the author ID and key pair to produce their own fake media. VeracOS assumes that the producer applies correct security measures to keep their data safe and secure against such adversaries. For the consumer, an adversary might intercept the verification process at the operating system level by falsely reporting that the media has been verified. To do so, the adversary must first compromise the operating system itself. Finally, a malicious application might report fake notifications of VeracOS verification. The consumer is urged to download trustworthy applications and to verify that the notifications are reported by VeracOS rather than applications.

4. Related work

Fact-checking against misinformation has been a focus of news reporting. Various news agencies and corporations are combating misinformation in their platforms (Meta, 2024a; CNN, 2024; Associated Press, 2024). Part of combating misinformation is preventing or detecting deepfake media. There is an ever-increasing interest in using artificial intelligence tools to protect against and prevent malicious behavior, such as cyber attacks (Babulak, 2023; Mohanraj and Babulak, 2019). Juefei-Xu et al. (2022) provides a survey study of over three hundred studies and papers regarding deepfake generation and detection. Juefei-Xu et al. (2022) overviews and considers the back-and-forth battle between generation and detection. Deepfake detection is not guaranteed with perfect accuracy (Barnaciak and Ross, 2022) and newer deepfake generation techniques may still subvert newer deepfake detection methods (Juefei-Xu et al., 2022).

Tracing or watermarking, similar to fingerprinting, research in media hides data in media by using neural networks or other artificial intelligence methods but does not provide a fully accurate detection method (Zhao et al., 2023; Guarnera et al., 2020). AMP (England et al., 2021) relies on fragile watermarking and a consortium framework for provenance but requires modifications to applications. Zheng et al. (2020) relies on trusted cameras that can assign hashes to produced content. This method requires certifying every camera separately even if it belongs to the same media creator. The objective of Zheng et al. (2020) is to identify the source camera. This method is also not fully accurate and might result in failed tamper detections. There is also no clear information on how (Zheng et al., 2020) can handle post-processed media files or streaming media.

VeracOS is not concerned with deepfake generation and does not require detection. VeracOS aims to prevent deepfakes from infiltrating social media or going viral by allowing users to automatically and easily be able to verify the authenticity of media without sharing the media or any metadata. Media creators or authors in VeracOS do not have to handle different media items separately nor do they have to certify every camera in their possession.

Instead of detecting deepfakes, some applications focus on media provenance (Hao, 2018; Coalition for Content Provenance and Authenticity, 2024). Truepic (Truepic, 2024) allows users to upload original media to their platform to allow consumers to compare the received media with what is available on Truepic. This requires users to share their media with a third party and also requires that the producer and consumer trust and use Truepic. Serelay (Serelay Limited, 2024) also uses a similar method to Truepic but uses fingerprints instead. Consumers of media also have to interact with Serelay to check their media. Similarly, PROVER (Prover, 2024) uses unique hashes of user-created videos allowing consumer users to verify the authenticity of their videos. OriginalMy (OriginalMy, 2024) also allows users to certify their media. All of these methods require the sharing of the original media by the author or creator and require users to share what media they are viewing to verify them. None of these methods work for offline verification.

Gipp et al. (2016) and Costales et al. (2023) both use blockchain to secure the authenticity of videos by making sure a record of the original video is created and traced. Hasan and Salah (2019) creates a framework using blockchain and the InterPlanetary File System (IPFS) (Protocol Labs, 2024) to track videos and trace them to their source. Hasan and Salah (2019) requires using blockchain, IPFS, the internet, and relies on a separate blockchain transaction for every video. While the cost per transaction may be minimal, the amount of videos circulating would cause a tremendous increase in the cost to support this framework. This method requires the sharing of videos from the consumer side when trying to verify unless they locally adopt a blockchain system and IPFS.

Unlike VeracOS, all of these solutions either require sharing of the media with a third party or require handling the media on a per-item case. For example, (Truepic, 2024; Serelay Limited, 2024; Prover, 2024) all trace individual media items which require the trust of authors and creators and require the trust of users as they query their media items, and thus leak information to those entities, such as the exact media item consumers are viewing. VeracOS is not computation

intensive, provides a scalable framework, and does not require sharing of media, modifying or reviewing blockchain transactions, or violating consumer privacy by sharing media metadata or the media itself. Additionally, all of these methods cannot and are not equipped to be able to verify live streaming of media such as video or audio.

Vronicle (Liu et al., 2022) focuses on video provenance but relies on hardware-assisted trusted execution environments that may not be available on systems. Vronicle certifies videos on a per-camera basis using these hardware extensions and requires application modifications for the consumer. Due to the heavy computations in Vronicle, its processing times are far inferior to VeracOS. Relevant research focuses on securing hardware to ensure unmodified data from the system (Saroiu and Wolman, 2010) but generally requires hardware modifications. ProvCam (Liu et al., 2024) uses a secure camera module approach but also requires hardware and device driver modifications. TalkLock (Shahid and Roy, 2023) creates dynamic Quick-Response (QR) codes for audio by generating meta-information from speech signals in real-time. However, TalkLock only applies to audio and is also not fully accurate.

Signature-based schemes such as Aletheia (danielquinn, 2024) use cryptographic signatures, similar to VeracOS. Aletheia relies on the Domain Name System and modifies the file format by modifying the file header for videos and images. As such, Aletheia generally requires application modifications. Other signature-based schemes focus on embedding information in each and every frame (Baser et al., 2024; Chen et al., 2020; Zhang et al., 2024). In addition to being computationally intensive and requiring application modifications, some of these systems are not fully accurate and do not envision a framework that combats deepfake media for the masses. For example, SecureSpectra (Baser et al., 2024) adds an inaudible high-frequency signature to audio to be verified and also requires a training phase. In all cases, these systems do not clearly address the accuracy issues stemming from widely used compression formats.

Hashing videos have been used in Singh (2021) to assign hashes to unencrypted or encrypted frames to identify videos that are being illegally shared. Singh (2021) shows that video hashing is a scalable and efficient method to identify matches without knowing the underlying content of the video. Providing software with hashes is a well-known method for providing downloadable software such as applications or operating systems (Canonical Limited, 2024). Hashing is not only performed and provided to users to verify if the download is corrupted but to also verify that they were not maliciously tampered with (Avast Software, 2024). For every piece of software or file, a unique hash is provided for the user to verify and compare to what is available on the respective website. VeracOS does not require creating or sharing hashes for every media item and does not require users to separately verify hashes for each media item.

A popular method used for ensuring authenticity of E-mails is the Pretty Good Privacy (PGP) (Broadcom Inc., 2024) encryption standard. PGP provides a way for users to sign their E-mails or files as a way to ensure their authenticity when sharing them by encryption. PGP is generally used in E-mails but is seldom used by users due to its complexity and inconvenience (Koh et al., 2019)

Some platforms such as the Android Play Store (Google LLC, 2024l) and Microsoft Windows (Microsoft Corporation, 2024b) require the signing of applications or device drivers with a certificate. These signatures or certifications are verified before installation and the user is informed if there is no valid signature to deter them from installing from unknown or unverified sources. This method is used to verify individual software releases and is heavily tied to certain ecosystems.

VeracOS's goal is to provide a scalable and easily deployed framework that media producers and media consumers can easily use to create verifiable media and to verify media. VeracOS does not require separate signatures for each media item, does not share the actual media, does not expose what the consumer is viewing to third parties, and does not require application modifications. VeracOS allows any entity to easily and automatically verify the original producer of the media.

Short name	*Name*
SOI	Start of image
APP0	Application marker (e.g., EXIF attribute information)
APP1	Application marker 2
DQT	Quantization table
DHT	Huffman table
DRI	Restart interval
SOF	Frame header
SOS	Scan header
	Compressed data
EOI	End of image

Fig. 2. General structure of a JPEG file.

5. Background

VeracOS framework modifies the operating system and media files produced. First, a brief overview of media files is given. Second, an overview of Android is given.

5.1. Media files

There are a plethora of media formats currently available and supported by many systems. VeracOS is designed for all files and aims to be applicable to all files. As an exemplary system, VeracOS uses media files, specifically the most popular media formats produced for video, audio, and images: The most common video format is the MPEG-4 (MP4) format, the most common audio format for recording is the MPEG Audio Layer III (MP3), the most common image format is the Joint Photographic Experts Group (JPEG or JPG).

The majority of media formats are organized and structured similarly (Schoning et al., 2017). Generally, the first part of a file includes the header which contains information about the file, format, size, and additional metadata. The second part of the file includes the image data in case of images or frames in case of audio and video. Some formats also include a header for every frame. Media viewers and players can deduce the size of the image or the size of each frame from either the metadata or from previous frames. Thus, media viewers and players can deduce the exact size of the media file and readable data from headers. Fig. 2 shows the general structure of a JPEG file as an example, with the header and the metadata first, followed by the image data, *compressed data*, and then an end of file, *end of image*, marker.

In terms of streaming audio or video, the structure is different since the size of the streamed media is not always predetermined. As such, any streamed formats usually contain an initial header or metadata file followed by a stream of frames. In some cases, every frame includes a header or metadata (Boris Asadanin, 2018).

5.2. Android

VeracOS uses Android as an exemplary operating system for several reasons. First, unlike traditional operating systems which are more straightforward in terms of a VeracOS implementation, mobile operating systems are highly vertically integrated which presents a more interesting challenge for supporting VeracOS (AlDuaij et al., 2019).

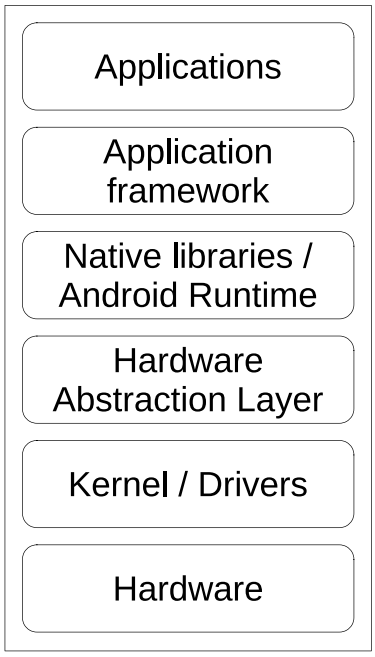

Fig. 3. Android architecture.

Second, mobile systems usually include embedded cameras and microphones, enabling users to easily produce and share media. Third, there are more smartphone and tablet users in the world than traditional operating systems or desktop and laptop users (Kepios Pte. Ltd., 2024). Note that any VeracOS modifications to Android can be repurposed for general operating systems as well.

Fig. 3 shows the Android architecture. On top of the stack are user and system applications. These applications communicate or call into the operating system through a layer of API frameworks and libraries. To communicate with hardware devices, Android offers a Hardware Abstraction Layer (HAL) for the API and libraries. This layer abstracts higher-level API from the details and intricacies of hardware. The HAL layer communicates directly through the kernel to hardware device drivers such as the storage or camera device driver. The device driver itself communicates with the hardware.

VeracOS is specifically concerned with the storage, camera, and microphone devices that enable the creation of videos and audio. Applications use common and public Android APIs to take images or record video and audio (Google LLC, 2024c). Similarly in other general operating systems, an API is used by applications to communicate with camera or microphone hardware (Microsoft Corporation, 2024a; Linux Kernel Organization, Inc., 2024; Apple Inc., 2024a). Through common and public APIs, applications can either request raw frames directly to package them in a media-formatted file or request the operating system directly create the media file based on the media taken or recorded.

Android Applications share and access media files through a media store abstraction (Google LLC, 2024k). By using a content provider (Google LLC, 2024d) and a content resolver (Google LLC, 2024e), as shown in Fig. 4, applications can store media and retrieve media for viewing. iOS provides a similar framework with a media store accessible through similar APIs (Apple Inc., 2024c,b,d). In some cases, applications do not want to store their media in a shared media storage to avoid exposing their files to other applications due to privacy. For those applications, there are APIs to store these files directly in application-specific storage (Google LLC, 2024f).

6. Architecture

VeracOS involves three major components, as shown in Fig. 5. First, the novel VeracOS algorithm that adds or checks the verification

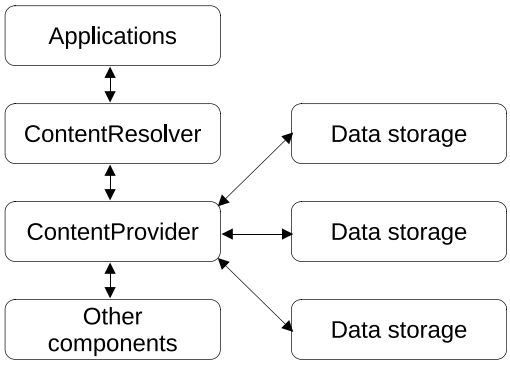

Fig. 4. Android content provider and resolver.

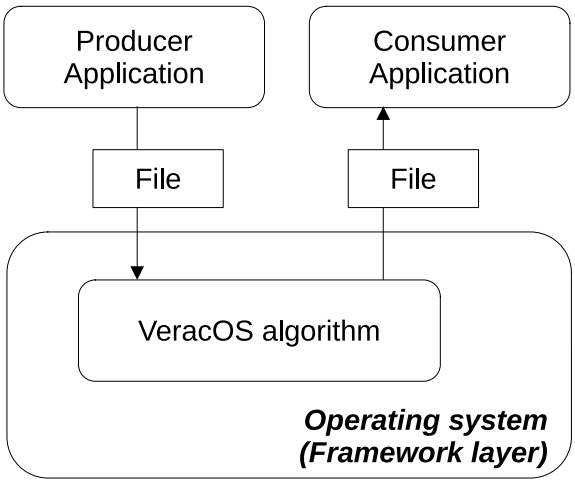

Fig. 5. VeracOS architecture.

component to the media. Second, the media producer that produces any media and thus utilizes the VeracOS algorithm for adding verification to the media. Third, the media consumer that consumes or views any media. These three components are integrated to address the following key challenges: (a) Providing a true verification component to media, (b) Allowing true verification of any media, (c) Creating a portable and adaptable framework that can be easily deployed on any system. This includes an efficient, secure, and privacy-friendly implementation that does not require special hardware or intensive and costly computations, (d) Maintaining backward compatibility. This section uses the most common media files and Android as an exemplary system for VeracOS. First, the operating system modifications and reasoning are discussed for producers. Second, operating system modifications and reasoning are discussed for consumers. Third, the novel algorithm used for VeracOS is listed and discussed along with its mathematical description. Fourth, special cases such as traditional operating systems and streaming are also discussed along with VeracOS limitations.

6.1. Media producers

Given the background of media files and operating systems, there are multiple ways to intercept produced or created media and add the verification component. One way to attempt this is to modify the storage, camera, and microphone hardware and include it as part of the firmware. This method requires all storage, camera, and microphone manufacturers to include an implementation of VeracOS. First, this makes VeracOS almost infeasible to support given the number of existing manufacturers and the requirement that they have to adopt and adhere to a new standard. Second, this might break compatibility

in certain cases such as live streaming. Third, even if manufacturers adopt these changes, the operating systems must be modified to adjust to these updates. Similarly, implementing VeracOS via the HAL layer would require the use of lower-level functionality and no access to higher-level APIs that would be useful and efficient for VeracOS.

Alternatively, on the other end of the spectrum, VeracOS media producers can be implemented with a user application instead. However, this hinders usage as it restricts users wanting to verify media to a single application. Users relying on different applications to create media will need to eventually export their media to the VeracOS application for processing. If VeracOS provides an API instead, all existing applications used by media producers need to be updated with VeracOS. More importantly, the unique and private author or creator signature would be exposed to these third-party applications, creating a massive privacy and security risk of leaking keys. VeracOS goal is to provide a secure and efficient system for a media producer that can automatically verify all created media, regardless of which application is being used to create the media.

As such, VeracOS takes a unique approach by intercepting at the operating system framework layer. VeracOS intercepts the API used after taking images, recording audio, and recording video. Specifically, VeracOS intercepts the MediaProvider (Google LLC, 2024j) module and API after the file creation stage of the media data. First, by intercepting at this layer, the hardware is already decoupled from the media. VeracOS would therefore work on any operating system regardless of the manufacturer or model of the storage, camera, or microphone and their device drivers. Second, at this stage, any post-processing of media frames or data would be complete. Post-processing of media is common and it is crucial that VeracOS intercepts only after the final media is created. Third, VeracOS would work regardless of the application that creates the media and so applications need not be updated. Fourth, backward compatibility is maintained with unsigned or unverified media. Fifth, signing or verifying the media happens at the last stage of media creation, allowing VeracOS to simply append the verification component without breaking compatibility nor requiring intensive computations or the use of the cloud.

In the case of live-streaming media, since there is no predetermined file size, the verification component is handled on a per-frame basis instead of at the end of a media file, and thus changes would be required for the camera or microphone APIs that provide frames.

Traditional operating systems like Linux and Windows do not have a similar framework to mobile operating systems for accessing media storage. File system or storage access on traditional operating systems is straightforward. To provide certification of media on these operating systems, a simple daemon can be provided to scan for media files added to the system. Once discovered, the verification component may be added depending on user preferences. More specifically, this daemon can focus on generic media directories in a user directory such as *Pictures* and *Videos*.

Media producer systems only require an author or creator signature or key to be added to an operating system. VeracOS provides a *Settings* menu item to provide the signature or key through a file or a URL link. This signature or key is then used by the operating system to add the verification component to every created media file. The Setting menu item for VeracOS also allows the user to specify whether to always certify media automatically on a per-application basis or to always ask the user via an actionable notification. Note that VeracOS does not certify downloaded media generally added to the downloads folder given their different source of origin.

6.2. Media consumers

Similar to the case of media producers, there are multiple ways to intercept consumed or viewed media and verify the media. To verify all media being viewed, even through existing unmodified applications, the operating system needs to be modified to automatically check these media files. Without modifying the operating system, any media file can be viewed by a third-party application without triggering the verification check. VeracOS intercepts at the framework layer with similar reasoning to media producer interception at the framework layer. Specifically, VeracOS intercepts the APIs for playing media (Google LLC, 2024i,h,b). In some cases, applications might create their own media player without relying on the standard operating system media player API. For this case, VeracOS also intercepts the content resolver (Google LLC, 2024e) API which lets applications query media files for their data before playing them. Unless the application is accessing private files, applications use this API to access available and non-private media files (Google LLC, 2024f).

Given this, VeracOS does not impose its certification or verification on files that applications would like to keep private in their own private or temporary directories. A more aggressive and less privacy-friendly form of VeracOS could intercept the kernel *open* or *read* functions or the Java *File* class to monitor every single access to any file on the system. For traditional operating systems, this method may be used in addition to intercepting platform media playing APIs. In the case of live streaming media, VeracOS can intercept the media playing, decoding, or viewing APIs for raw or encoded frames.

VeracOS operating system can access an online database of author or creator signatures or keys. Alternatively, users may provide a link to or a file of author signatures or keys. Verifying the signature requires simple computations and a single check to authenticate that the media provided is the original author-created media. For users that do not have the VeracOS operating system extension, an application to verify the media is provided. This VeracOS application can be compiled to run on any operating system. However, this requires users to manually verify their media by providing it to this application.

6.3. VeracOS algorithm

To support verifying media, VeracOS computes the hash value of the entire media item and then encrypts the hash value with the author or creator signature, a secret encryption key as part of an asymmetric key pair. This value, along with a unique author ID, is then appended to the media item, without the modification of the media format while still maintaining backward compatibility with viewers and players of the media item. For users receiving the media item, the item authenticity can be easily verified by getting the author or creator public decryption key from the author ID to decrypt the hash value of the given item and comparing it with the actual media item hash value. This means that the author or creator of the media item is the only entity that can encrypt the hash value. This hash value represents the exact media item created by the author. Anyone or any system can decrypt this hash value and compare it to the received media item's locally computed hash value, which is appended to the end of the media file. The only information required from the consumer would be a tuple of author ID and key. Encrypting the hash value ensures that the media item was signed only by the author and if the decrypted hash value matches the calculated hash for the file, it ensures that the file was not modified since certifying it.

If a malicious entity modifies the hash value to match a modified media item, the malicious entity will not be able to encrypt the hash value with the author's secret encryption key. Any hash value appended to the media item by the malicious entity will not decrypt to the true hash value of the media item. If a malicious entity tries to only modify the media item instead, the hash values will not be equivalent. If the author ID is modified along with the media and encrypted hash, the user will know that the verification is based on a different author. This allows media consumers to easily verify media without leaking any information on what media they are viewing nor requiring the use of third-party applications or tools. Media consumers can also easily verify media even when they are offline.

Media producers can share their author or creator signature and media consumers can add these signatures or keys to verify their media. A database of keys from known authors or creators of media can be provided with VeracOS as a file. Alternatively, VeracOS also allows providing these keys via an online link. VeracOS envisions a trusted authority to provide all of the author or creator verified keys through online servers, in a system or framework similar to certificate authorities (Michael Labos, 2024) such as the ones used for websites.

Algorithm 1 VeracOS algorithm for producers

```
function AddVerification(filePath)
    if fileType(filePath) ∉ mediaTypes then return
    end if
    authorID ←GetAuthorID()
    encryptKey ←GetAuthorEncryptKey(authorID)
    calcHash ←CalculateHash(filePath)
    encHash ←EncryptHash(encryptKey, calcHash)
    AppendAuthorID(authorID, filePath)
    AppendEncryptedHash(encHash, filePath)
    NotifyUserOptional(mediaCertified)
end function
```

Fig. 6 shows the general flow of the producer and consumer. The novel algorithm for producers is listed in Algorithm 1 and shows the process of creating a verification component. First, the file type for the file is checked against a list of VeracOS media file types. Note that VeracOS does not preclude other file types; it simply focuses on media as exemplary file types. Second, the author ID and the unique secret encryption key are retrieved from VeracOS. Third, the hash for the file is calculated. Fourth, the hash is then encrypted. Fifth, the file is then appended with the author ID in cleartext and then the encrypted hash. Note that the author ID is unique per author and may be reused in multiple systems and producers. The author ID must remain a secret and only accessible by authorized entities. Finally, an optional notification is sent to the user regarding the certification of the media file.

Algorithm 2 VeracOS algorithm for consumers

```
function VerifyFile(filePath)
    if fileType(filePath) ∉ mediaTypes then return
    end if
    authorID ←GetAuthorID(filePath)
    decryptKey ←GetDecryptionKey(authorID)
    encHash ←GetEncryptedHash(filePath)
    decHash ←DecryptHash(decryptKey, encHash)
    calcHash ←CalculateHash(filePath)
    if calcHash == decHash then
        NotifyUser(verificationPassed)
    else
        NotifyUser(verificationFailed)
    end if
end function
```

The novel algorithm for consumers is listed in Algorithm 2 and shows the process to verify a component. First, the file type for the file is checked against a list of VeracOS media file types. Note, again, that VeracOS does not preclude other file types; it simply focuses on media as exemplary file types. Second, the author ID is retrieved and verified by retrieving the respective public decryption key. This decryption key is tied to a single author ID and can be retrieved locally, through a link, or via authorized and trusted servers. Third, the encrypted hash is retrieved directly from the file. Fourth, the hash is then decrypted. Fifth, the hash of the file is calculated. Note that VeracOS calculates the hash of the original file, without the appended author ID and encrypted hash. Finally, the hash from the file and the calculated hash are compared. If the hashes are equal, the media is verified to be produced by the author without being tampered with. If the hashes are not equal, the media is either never signed by VeracOS or not verified as truly produced by the given author.

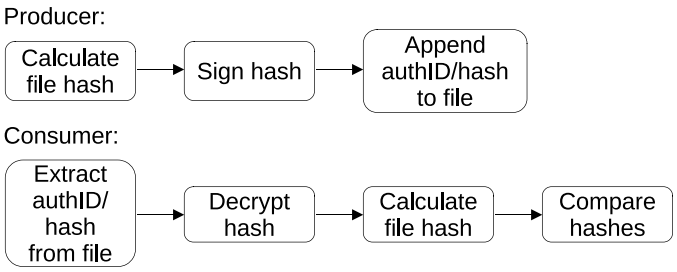

Fig. 6. VeracOS producer and consumer algorithm flow.

These algorithms can also be applied on a per-frame basis in the case of live streaming. Note that the VeracOS algorithm can also be expanded and used for any file type as the core functionalities can be applied to any file. VeracOS simply focuses on media as exemplary file types.

6.4. VeracOS mathematical description

To improve clarity and rigor in the cryptographic operations of VeracOS, the following subsections present a formal description of the core mechanisms used in file certification and verification. The goal is to mathematically define the behavior of the VeracOS producer and consumer algorithms, particularly their use of cryptographic primitives.

Notation

Let the following denote the key elements used throughout the system:

- Let $m \in \{0,1\}^*$ denote the full content of a media file.
- Let $H : \{0,1\}^* \rightarrow \{0,1\}^n$ be a cryptographic hash function (e.g., RSA-2048), where n is the hash output size.
- Let sk_A be the private signing key of an author A, and pk_A be the corresponding public verification key.
- Let $\sigma \in \{0,1\}^*$ denote the digital signature produced by the author.

Certification (producer side)

Given a media file m, the certification procedure executed by the VeracOS producer is:

1. Compute the hash of the media:
   $$h = H(m)$$
2. Sign the hash using the author's private key:
   $$\sigma = \mathrm{Sign}_{\mathrm{sk}_A}(h)$$
3. Append the tuple (id_A, σ) to the end of the file, where id_A is a unique identifier for the author.

This results in the augmented media file:

$$m' = m \,\|\, \mathrm{id}_A \,\|\, \sigma$$

Verification (consumer side)

Given a received media file m', the consumer performs the following:

1. Parse m' to extract m, id_A, and σ.
2. Retrieve pk_A using id_A.
3. Compute $h' = H(m)$.

4. Verify the signature:

$$\text{Verify}_{\text{pk}_A}(h', \sigma) \stackrel{?}{=} \text{true}$$

If the signature is valid, then m is guaranteed to have originated from the author A and has not been modified.

Security guarantees

This scheme satisfies the following security properties under standard cryptographic assumptions:

- **Authenticity:** Only an entity with sk_A can produce a valid σ for a given m.
- **Integrity:** Any modification to m will result in $H(m) \neq h'$, invalidating the verification.
- **Non-repudiation:** The signature σ binds A to the content of m, assuming secure key management.

This formalization captures the essence of the cryptographic operations in VeracOS.

6.5. Cross-operating system feasibility

Modern mobile operating systems are designed similarly, with a tall interface to devices via multiple layers of software (Andrus et al., 2014; AlDuaij et al., 2019). iOS provides the *Photo capture*, *PhotoKit*, and *Media Player* APIs to allow applications to create and access media, and to view shared storage and their files. Intercepting these APIs allows for adding the producer and consumer components to modify and verify media. Modern traditional operating systems, such as Apple macOS and Microsoft Windows, generally allow automated indexing of files on a system. These indexers and the operating system camera capture APIs can be intercepted to add the producer and verification component. The media player APIs can be intercepted by the consumer component to verify the media being viewed.

6.6. VeracOS limitations

Deepfake Media Signed By A Trusted Producer: VeracOS verifies that media truly originated from a trusted producer. VeracOS does not prevent deepfake media from being created. For example, if a trusted producer, such as a news agency, decides to create their own deepfake media and disseminate this media, it will be successfully verified by VeracOS. The goal of VeracOS is to prevent deepfake media from being disseminated by unaffiliated producers. It is up to users to trust this producer, such as a news agency, with the actual media. However, VeracOS can be modified in the future to also include deepfake media detection methods to identify deepfake media from a trusted producer. This producer can then be flagged as a potentially untrustworthy producer.

Private Application Files: In order for VeracOS not to break the privacy model introduced in Android and various operating systems, such as iOS, VeracOS accesses and modifies files that are provided outside of the private internal storage of applications. Applications may be modified to share those files or use VeracOS code. Alternatively, users may export their files to a VeracOS application or shared media storage. As future work, VeracOS may create an API that includes a static or dynamic option for applications to allow VeracOS to access their private storage.

Legacy Systems and Their Applications: Media modified and produced by VeracOS is still viewable properly on legacy systems and their applications. However, due to these legacy systems not including the VeracOS modifications, the verification process will clearly not be available. As an alternative for users who do not have access to the VeracOS extension, a VeracOS-based application can be used to verify any media provided to it via shareable storage or exported by the consumer from different applications.

Malicious Operating System: A malicious user may share a hacked version of the VeracOS framework to trick users into using a false VeracOS system. To subvert this, VeracOS can offer a trustworthy server to share valid VeracOS components or operating system with correct hashes as is done by many existing operating systems. Additionally, VeracOS does not introduce additional security features beyond what is offered by existing operating systems to prevent root access and modification of the framework or the kernel. Formal verification of VeracOS may be useful in this case.

7. Implementation

VeracOS implements the producer component in the MediaProvider module that is part of the Android Open Source Project (AOSP) operating system. VeracOS intercepts the *scanFile* functions since these functions are called when a new media item is to be discovered. VeracOS implements the consumer component in the ContentResolver API, by intercepting the *query* functions, the MediaPlayer API, by intercepting the *setDataSource* and *prepare* functions, the MediaExtractor API, by intercepting the *setDataSource* function, and the BitmapFactory API, by intercepting the *decodeFile* functions. These are the functions used to access these media files.

In all cases, VeracOS verifies the file type before proceeding to ensure that the file is a media file by checking the extension. A more sophisticated method may look at the file header instead. VeracOS utilizes the MessageDigest API along with the SHA-256 secure cryptographic hash function. For asymmetric key generation, encryption, and decryption, RSA-2048 is used. Notifications and intents, in the case of the producer, are used to notify the user of any certifications or verifications. The *Settings* package or application is also modified to add VeracOS options.

Keys are provided through local files or downloaded via *URLConnection* API and generally stored in local storage. Alternatively, the Android Keystore system or the *KeyChain* API may be used.

8. Evaluation

For evaluation, examples of VeracOS certifying and verifying media are shown, benchmark tests were run, and timing analysis was performed. The Google Pixel 6a (bluejay, Google Tensor Octa-core, 6GB RAM, 128GB storage) was used. VeracOS is implemented using the Android Open Source Project (Google LLC, 2024a) version 14, release UQ1 A.240205.002.

8.1. VeracOS producer and consumer examples

Fig. 7 shows a screenshot of a notification after recording a video with the camera. The three actions are interactive via notification. Fig. 8 shows a screenshot taken when viewing a video through a popular and widely used media player, VLC media player (Videolabs, 2024) without modifying it for VeracOS. Note that the notification informs the user of the verification result which, in this case, has passed the verification. Finally, Fig. 9 shows a screenshot taken after viewing an image with the default Android Gallery application. The image being viewed was modified before viewing by flipping a single bit in the image file from a zero to a one, without certifying it again. The bit was not part of the header nor the verification component as it is in the middle of the file, representing image data. This shows, that due to the properties of secure cryptographic hash functions, a simple bit change causes a different hash component that cannot match the original hash of the file therefore resulting in a failed verification. All notifications appear when initially viewing the media and a failed verification does not prevent the user from viewing the media.

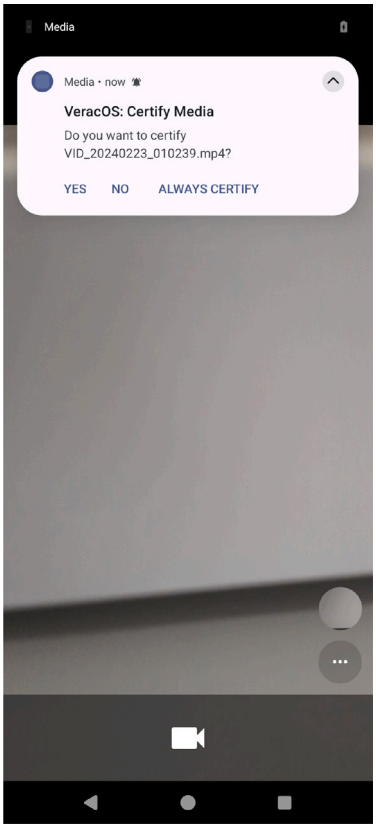

Fig. 7. VeracOS producer or certification notification.

Fig. 8. VeracOS consumer passed verification notification.

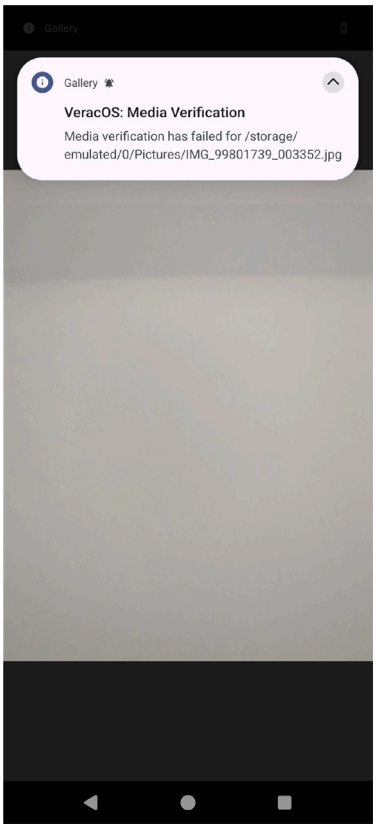

Fig. 9. VeracOS consumer failed verification notification.

Table 1
Producer timing breakdown.

Subcomponent	Image	Audio	Video	Video
File type	JPEG	MP4	MP4	MP4
File size (MB)	1.16	1.22	109.60	490.40
Resolution/Sample rate	3024 × 4032	44.1 KHz	2160 × 3840	2160 × 3840
Duration (s)	–	60	10	60
Calculate hash (ms)	30.2	34.0	1958.6	8507.4
Encrypt hash (ms)	5.6	5.0	4.8	6.4
Append hash to file (ms)	0.8	0.8	0.8	0.7
Notify user (ms)	1	1	1	1
Total (ms)	37.6	40.8	1965.2	8515.5

8.2. Timing analysis

VeracOS was instrumented to measure the time spent in each component, producer and consumer, as well as their subcomponents. For each component, a media file was either produced or consumed. This process was repeated five times and the average of all timings in milliseconds (ms) was taken.

For the producer, an image was taken five times averaging 1.16 Megabytes (MB) per file, a one-minute audio (44.1 KHz) was recorded five times averaging 1.22 MB, a ten-second high-definition (4K, 2160 × 3840) video was recorded averaging 109.60 MB, and a one-minute high-definition (4K, 2160 × 3840) video was recorded averaging 490.40 MB. The subcomponents of the producer or verification process include calculating the hash for the file, encrypting the hash, and appending the author ID and hash to the file.

Table 1 shows the time spent by the producer, with respect to time spent certifying media and creating the verification component. The difference in average timings is directly related to the file size, not the file content or type. Images and audio perform similarly since they average similar file sizes. Table 1 also shows the average timings for

Table 2
Consumer timing breakdown.

Subcomponent	Image	Audio	Video	Video
File type	JPEG	MP4	MP4	MP4
File size (MB)	1.15	1.21	107.30	520.51
Resolution/Sample rate	3024 × 4032	44.1 KHz	2160 × 3840	2160 × 3840
Duration (s)	–	60	10	60
Extract author ID/hash (ms)	2.6	1.9	2.7	1.5
Decrypt hash (ms)	11.2	10.1	10.3	10.5
Calculate hash (ms)	33.4	32.7	2723.0	13,751.5
Hash comparison (ms)	0	0	0	0
Notify user (ms)	10.4	8.3	7.3	9.9
Total (ms)	57.6	53.0	2743.3	13,773.4

videos around 110 MB in size and around 490 MB in size. The majority of the time is spent calculating the hash which includes opening the file, reading the file, and updating the hash with the buffer data. As such, the timing is directly related to the size of the file. For 1 MB files, the average time spent is around 39 ms. For 110 MB files, the average time spent is around 2000 ms. For 490 MB files, the average time spent is around 8515 ms. Note that during this time, the user is not blocked and may proceed to view the media as they await the notification. Encrypting the hash is almost fixed around 5 ms since the hash produced and encrypted is the same size regardless of the file size. Finally, appending the hash to the file and notifying the user measures a maximum average of around 1 ms.

For the consumer, the same media produced and created by the producer are viewed five times and the averages of timings are taken. The subcomponents of the consumer or viewing process includes extracting the author ID and the saved encrypted hash from the end of the file, decrypting the hash, calculating the hash of the file itself without the VeracOS verification component, comparing hashes, and notifying the user.

Table 2 shows the time spent by the consumer, with respect to time spent verifying media. Similar to the producer, the timings are directly related to the file size and not the file content or type. Images and audio perform similarly since they average the same file size. Extracting the author ID and hash takes on average 1.5–2.7 ms for all cases. Decrypting averages around 10.1–11.2 ms. The majority of the time is spent calculating the actual hash of the file. For around 1.2 MB of images or audio, it took around 53–58 ms. For around 110 MB of video, it took around 2743 ms. For around 521 MB of video, it took around 13773 ms. Comparing the hashes takes less than 1 ms. Notifying the user takes on average 7–11 ms, due to the actions added to the notification. The majority of the time is spent calculating the hash which includes opening the file, reading the file, and updating the hash with the buffer data.

Note that key creation, which occurs only when a user creates an author ID for a producer, takes an average of 265 ms. Loading keys from a file instead of memory takes an average of 4 ms.

From Table 1, the average producer throughput for the 1 MB files is 30.4 MB/s and the average producer throughput for the larger, 110 MB and 490 MB, files is 56.7 MB/s. From Table 2, the average consumer throughput for the 1 MB files is 21.4 MB/s and the average consumer throughput for the larger, 110 MB and 521 MB, files is 38.5 MB/s. Inversely, the processing time of the producer for the 1 MB files is 32.9 ms/MB and for the larger files it is 17.6 ms/MB. The processing time of the consumer for the 1 MB files is 46.9 ms/MB and for the larger files it is 26.0 ms/MB.

The majority of time shown for VeracOS is spent calculating hashes as it requires reading and processing an entire file. Faster algorithms as well as efficient third-party native libraries may be used instead. Additionally, more recent systems provide significantly better performance than the Google Pixel 6a in terms of disk, CPU, and memory tests (PassMark Software, 2025). Regardless of the system being used, the delay in certification or verification is tolerable since users are not blocked and are not prevented from producing or viewing media during that time.

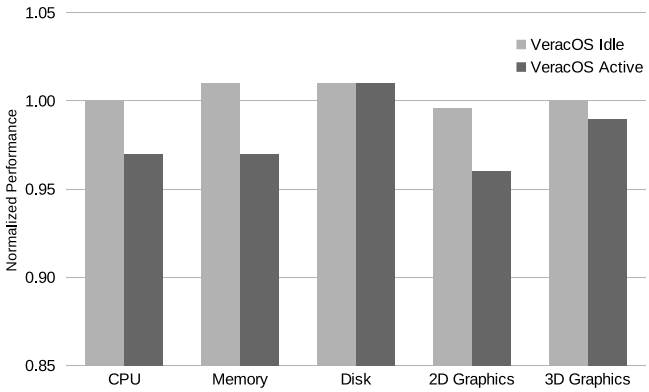

Fig. 10. PassMark Benchmarks (higher is better).

8.3. Performance measurements

For evaluating VeracOS performance, the widely used Android PassMark benchmark (PassMark Software, 2024) application was used which contains a set of resource-intensive tests to evaluate CPU, memory, I/O, and graphics performance. PassMark was run in three different configurations: First, it was run using idle vanilla AOSP without any VeracOS changes. Second, it was run with VeracOS idle. Third, it was run with VeracOS fully active. Given that VeracOS certification or verification takes mere seconds or less and the PassMark test takes over four minutes, a more representative test would ensure that VeracOS is active for the entire duration of the test. To simulate the worst-case performance as a stress test, two threads were spawned and run during the PassMark test. The first thread continuously calculates the hash from a sixty-second video file it reads and the second thread continuously encrypts and then decrypts a hash value. The hash algorithm used is SHA-256 and the encryption algorithm used is RSA-2048 as mentioned in Section 7. PassMark was run five times for each configuration, and the results were averaged across each configuration.

Fig. 10 shows the test results from running the PassMark benchmark tests. All results are normalized to the vanilla AOSP case, higher results are better. The difference in results between vanilla AOSP and VeracOS idle is negligible. The difference between vanilla AOSP and VeracOS active, is very slight. There is almost no difference in terms of disk and 3D tests. There is a slight difference in terms of CPU, memory, and 2D tests as the VeracOS active case performs computations that stress the CPU versus idle vanilla AOSP. Even in the unrealistic worst case of VeracOS actively and continuously performing file certification and verification tasks, the performance of VeracOS is good.

8.4. VeracOS versus other solutions

Table 3 shows a comparison of VeracOS with other solutions related to the prevention or detection of deepfake material. Table 3 lists the restrictions to compare against other solutions. The first restriction lists whether the solution provided is on a per-camera basis rather than being applied universally across systems. The second and third restrictions list whether the solution requires hardware modifications or application modifications. The fourth restriction lists whether the solution requires the sharing of the file or metadata to certify or verify the file. Finally, the performance is listed depending on what is available.

A few solutions are used on a per-camera basis rather than on a per-producer basis. As such, a producer would need to ensure that all cameras in their possession contain the required modifications. Most of these solutions require hardware modifications. The vast majority of solutions require application support to work, this could be a standalone application or it could be in the form of application

Table 3
Comparison of VeracOS with other solutions.

Solutions \ Restrictions	Per-camera	H/W Mod.	Application Mod.	Shares files	Performance
VeracOS					8.5–13.8 s/60 s video (2160 × 3840)
Vronicle (Liu et al., 2022)	✓	✓			4 s/30 frames (1280 × 720)
(Zhao et al., 2023; Guarnera et al., 2020)			✓		Only detection
AMP (England et al., 2021)			✓	✓	80–248 ms latency
(Zheng et al., 2020)	✓		✓		Only images
Truepic (Truepic, 2024)			✓	✓	Cloud
Serelay (Serelay Limited, 2024)			✓	✓	Cloud
PROVER (Prover, 2024)			✓	✓	Cloud
OriginalMy (OriginalMy, 2024)			✓	✓	Cloud
(Gipp et al., 2016; Costales et al., 2023; Hasan and Salah, 2019)			✓	✓	Blockchain transaction/video
ProvCam (Liu et al., 2024)	✓	✓			0.12 s Camera stop latency
Aletheia (danielquinn, 2024)			✓		N/A
TalkLock (Shahid and Roy, 2023; Baser et al., 2024)			✓		Only audio

modifications. For example, Aletheia modifies the file format header, requiring applications to adjust to these format changes. Many solutions also require the sharing of the actual file or its metadata with third parties, creating a privacy risk. Finally, the performance is listed for each solution. For some solutions, the majority of the task is performed on the cloud. Two solutions are either strictly audio or images. The blockchain solutions require a blockchain transaction per video. The watermark or fingerprinting solutions are generally used only for detection. The performance of VeracOS is far superior to Vronicle, however, it does not compare favorably to ProvCam as ProvCam is a per-camera basis solution that requires hardware and device driver modifications.

VeracOS provides a solution that does not require hardware or application modifications. VeracOS does not require sharing of the files or any of its metadata and can be used in a per-camera based or as large as a per-organization granularity to represent all the videos from an entity. The performance for images and audio is shown to be acceptable. For videos, VeracOS shows overhead that is non-blocking and would not hinder the user experience. Furthermore, VeracOS provides flexibility to be extended to various file types and not only media files.

9. Conclusion and future work

VeracOS is an operating system extension created for combating misinformation and exploitation of fake files by certifying and verifying them. VeracOS modifies Android and creates a novel algorithm to allow certification of files and verification of files, using media files as exemplary files. Certifying created media and verifying viewed media is automatic. Unlike existing solutions and methods, VeracOS requires no specialized hardware, application modifications, sharing of private data, or a learning curve from a user. VeracOS is the first viable system to provide this unique and novel algorithm for file certification and verification by relying on leveraging existing features and APIs in operating systems. VeracOS intercepts APIs and allows for certification and verification of files, specifically media, without requiring application modifications or specialized hardware. VeracOS is also an easily portable system that can be easily deployed and utilized in various operating systems, including mobile operating systems, Internet of Things devices, and embedded devices. A VeracOS prototype was built using Android and tested to show that VeracOS operates automatically, works on existing and less powerful hardware such as smartphones, and performs well even in extreme cases. Part of the testing included using the default Android camera application to create media and using the widely popular VLC application and the default Android Gallery application to view media. For future work, deploying VeracOS to other systems, other files, intercepting via kernel functions, and live streaming could be implemented. Dealing with special cases such as when some applications further compress media files to reduce the size of files should be addressed. Additionally, using an overlay network to hide and mask user queries of author IDs would be beneficial for increased privacy.

Declaration of competing interest

The author declares that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The author would like to thank Kuwait University and its College of Science for their support and facilities.

Data availability

No data was used for the research described in the article.

References

AlDuaij, N., Van't Hof, A., Nieh, J., 2019. Heterogeneous multi-mobile computing. In: Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services. MobiSys 2019, Association for Computing Machinery, New York, NY, USA, pp. 494–507.

Andrus, J., Van't Hof, A., AlDuaij, N., Dall, C., Viennot, N., Nieh, J., 2014. Cider: native execution of iOS apps on android. In: Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems. ASPLOS '14, New York, NY, USA, pp. 367–382.

Apple Inc., 2024a. Capture setup. https://developer.apple.com/documentation/avfoundation/capture_setup. (Accessed 11 February 2024).

Apple Inc., 2024b. PHPhotoLibrary. https://developer.apple.com/documentation/photokit/phphotolibrary?language=objc. (Accessed 18 February 2024).

Apple Inc., 2024c. Selecting photos and videos in iOS. https://developer.apple.com/documentation/photokit/selecting_photos_and_videos_in_ios?language=objc. (Accessed 18 February 2024).

Apple Inc., 2024d. UIImagePickerController. https://developer.apple.com/documentation/uikit/uiimagepickercontroller?language=objc. (Accessed 18 February 2024).

Associated Press, 2024. AP fact check. https://apnews.com/ap-fact-check. (Accessed 09 February 2024).

Avast Software, 2024. What is the MD5 hashing algorithm and how does it work? https://www.avast.com/c-md5-hashing-algorithm. (Accessed 09 February 2024).

Babulak, E., 2023. AI Tools for Protecting and Preventing Sophisticated Cyber Attacks. IGI Global.

Barnaciak, C., Ross, D.A., 2022. How Easy Is It to Make and Detect a Deepfake?. https://insights.sei.cmu.edu/blog/how-easy-is-it-to-make-and-detect-a-deepfake/.

Bartz, D., 2023. Microsoft chief says deep fakes are biggest AI concern. https://www.thomsonreuters.com/en-us/posts/technology/practice-innovations-deepfakes/.

Baser, O., Kale, K., Chinchali, S.P., 2024. SecureSpectra: Safeguarding digital identity from deep fake threats via intelligent signatures. In: Proceedings of Interspeech 2024. Interspeech '24, pp. 1115–1119.

Belle Wong, J.D., 2023. Top social media statistics and trends Of 2024. https://www.forbes.com/advisor/business/social-media-statistics/.

Boris Asadanin, 2018. Internet Video Streaming - ABR part 2. https://eyevinntechnology.medium.com/internet-video-streaming-abr-part-2-dbce136b0d7c.

Britt, K., 2023. How are deepfakes dangerous?. https://www.unr.edu/nevada-today/news/2023/atp-deepfakes.

Broadcom Inc., 2024. OpenPGP. https://www.openpgp.org/. (Accessed 09 February 2024).

California Legislative Information, 2024. AB-730 Elections: deceptive audio or visual media. https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201920200AB730. (Accessed 07 February 2024).

Canonical Limited, 2024. Ubuntu releases. https://releases.ubuntu.com/. (Accessed 09 February 2024).

Chen, H., Darvish, B., Koushanfar, F., 2020. SpecMark: A spectral watermarking framework for IP protection of speech recognition systems. In: Proceedings of Interspeech 2020. Interspeech '20, pp. 2312–2316.

Chen, H., Magramo, K., 2024. Finance worker pays out $25 million after video call with deepfake 'chief financial officer'. https://edition.cnn.com/2024/02/04/asia/deepfake-cfo-scam-hong-kong-intl-hnk/index.html.

CNN, 2024. Facts first. https://edition.cnn.com/politics/fact-check. (Accessed 09 February 2024).

Coalition for Content Provenance and Authenticity, 2024. Overview - C2PA. https://c2pa.org/. (Accessed 09 February 2024).

Costales, J.A., Shiromani, S., Devaraj, M., 2023. The impact of blockchain technology to protect image and video integrity from identity theft using deepfake analyzer. In: 2023 International Conference on Innovative Data Communication Technologies and Application. ICIDCA, pp. 730–733.

danielquinn, 2024. aletheia. https://danielquinn.github.io/aletheia/. (Accessed 21 February 2024).

Department of Homeland Security, 2024. Increasing threat of deepfake identities. https://www.dhs.gov/sites/default/files/publications/increasing_threats_of_deepfake_identities_0.pdf. (Accessed 07 February 2024).

England, P., Malvar, H.S., Horvitz, E., Stokes, J.W., Fournet, C., Burke-Aguero, R., Chamayou, A., Clebsch, S., Costa, M., Deutscher, J., Erfani, S., Gaylor, M., Jenks, A., Kane, K., Redmiles, E.M., Shamis, A., Sharma, I., Simmons, J.C., Wenker, S., Zaman, A., 2021. AMP: authentication of media via provenance. In: Proceedings of the 12th ACM Multimedia Systems Conference. MMSys '21, New York, NY, USA, pp. 108–121.

eSafetyCommissioner - Australian Government, 2024. Deepfake trends and challenges. https://www.esafety.gov.au/sites/default/files/2022-01/Deepfake-position-statement%20_v2.pdf. (Accessed 07 February 2024).

ETtech, 2023. Deepfake menace: Govt issues advisory to social media platforms to comply with IT rules. https://economictimes.indiatimes.com/tech/technology/deepfake-menace-govt-issues-advisory-to-intermediaries-to-comply-with-existing-it-rules/articleshow/106297813.cms.

Gipp, B., Kosti, J., Breitinger, C., 2016. Securing video integrity using decentralized trusted timestamping on the bitcoin blockchain. In: 10th Mediterranean Conference on Information Systems. MCIS.

Google LLC, 2024a. Android Open Source Project. https://source.android.com/. (Accessed 19 February 2024).

Google LLC, 2024b. BitmapFactory. https://developer.android.com/reference/android/graphics/BitmapFactory. (Accessed 19 February 2024).

Google LLC, 2024c. Choose a camera library. https://developer.android.com/media/camera/choose-camera-library. (Accessed 11 February 2024).

Google LLC, 2024d. Content provider basics. https://developer.android.com/guide/topics/providers/content-provider-basics. (Accessed 18 February 2024).

Google LLC, 2024e. ContentResolver. https://developer.android.com/reference/android/content/ContentResolver. (Accessed 18 February 2024).

Google LLC, 2024f. Data and file storage overview. https://developer.android.com/training/data-storage. (Accessed 18 February 2024).

Google LLC, 2024g. Generate text, images, code, and more with google cloud AI. https://cloud.google.com/use-cases/generative-ai. (Accessed 07 February 2024).

Google LLC, 2024h. MediaExtractor. https://developer.android.com/reference/android/media/MediaExtractor. (Accessed 19 February 2024).

Google LLC, 2024i. MediaPlayer. https://developer.android.com/reference/android/media/MediaPlayer. (Accessed 18 February 2024).

Google LLC, 2024j. MediaProvider. https://source.android.com/docs/core/ota/modular-system/mediaprovider. (Accessed 18 February 2024).

Google LLC, 2024k. MediaStore. https://developer.android.com/reference/android/provider/MediaStore. (Accessed 18 February 2024).

Google LLC, 2024l. Sign your app. https://developer.android.com/studio/publish/app-signing. (Accessed 09 February 2024).

Groh, M., Epstein, Z., Firestone, C., Picard, R., 2022. Deepfake detection by human crowds, machines, and machine-informed crowds. Proc. Natl. Acad. Sci. 119.

Guarnera, L., Giudice, O., Battiato, S., 2020. DeepFake detection by analyzing convolutional traces. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. CVPRW, pp. 2841–2850.

GZERO Staff, 2023. How AI and deepfakes are being used for malicious reasons. https://www.gzeromedia.com/global-stage/digital-governance/how-ai-and-deepfakes-are-being-used-for-malicious-reasons.

Hao, K., 2018. Deepfake-busting apps can spot even a single pixel out of place. https://www.technologyreview.com/2018/11/01/139227/deepfake-busting-apps-can-spot-even-a-single-pixel-out-of-place/.

Hasan, H.R., Salah, K., 2019. Combating deepfake videos using blockchain and smart contracts. IEEE Access 7, 41596–41606.

Hsu, T., 2023. As deepfakes flourish, countries struggle with response. https://www.nytimes.com/2023/01/22/business/media/deepfake-regulation-difficulty.html.

Jackson, A., 2023. The rising tide of deepfakes as AI growth cause concern. https://cybermagazine.com/technology-and-ai/the-rising-tide-of-deepfakes-as-ai-growth-cause-concern.

Juefei-Xu, F., Wang, R., Huang, Y., Guo, Q., Ma, L., Liu, Y., 2022. Countering malicious DeepFakes: Survey, battleground, and horizon. Int. J. Compter Vis. 130, 1678–1734.

Kaggle, 2024. Deepfake detection challenge. https://www.kaggle.com/c/deepfake-detection-challenge. (Accessed 07 February 2024).

Kan, M., 2023. FBI: Scammers Using Public Photos, Videos for Deepfake Extortion Schemes. https://www.pcmag.com/news/fbi-scammers-using-public-photos-videos-for-deepfake-extortion-scheme.

Kepios Pte. Ltd., 2024. Digital around the world. https://datareportal.com/global-digital-overview. (Accessed 20 February 2024).

Knight, W., 2018. The US military is funding an effort to catch deepfakes and other AI trickery. https://www.technologyreview.com/2018/05/23/142770/the-us-military-is-funding-an-effort-to-catch-deepfakes-and-other-ai-trickery/.

Koh, J.S., Bellovin, S.M., Nieh, J., 2019. Why joanie can encrypt: Easy email encryption with easy key management. In: Proceedings of the Fourteenth EuroSys Conference 2019. In: EuroSys 2019, Association for Computing Machinery, New York, NY, USA.

KPMG, 2024. Deepfakes: Real threat. https://kpmg.com/kpmg-us/content/dam/kpmg/pdf/2023/deepfakes-real-threat.pdf. (Accessed 07 February 2024).

Linux Kernel Organization, Inc., 2024. 1.11. Camera control reference. https://www.kernel.org/doc/html/next/userspace-api/media/v4l/ext-ctrls-camera.html. (Accessed 11 February 2024).

Liu, Y.M., Nakatsuka, Y., Sani, A.A., Agarwal, S., Tsudik, G., 2022. Vronicle: verifiable provenance for videos from mobile devices. In: Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services. MobiSys '22, New York, NY, USA, pp. 196–208.

Liu, Y.M., Yao, Z., Chen, M., Amiri Sani, A., Agarwal, S., Tsudik, G., 2024. ProvCam: A camera module with self-contained TCB for producing verifiable videos. In: Proceedings of the 30th Annual International Conference on Mobile Computing and Networking. MobiCom '24, New York, NY, USA, pp. 588–602.

Matt Novak, 2023. Viral video of Kamala Harris speaking Gibberish is actually A deepfake. https://www.forbes.com/sites/mattnovak/2023/05/08/viral-video-of-kamala-harris-speaking-gibberish-is-deepfake.

Merriam-Webster, 2024. Deepfake definition & meaning - merriam-webster. https://www.merriam-webster.com/dictionary/deepfake. (Accessed 07 February 2024).

Meta, 2024a. About fact-checking on Facebook and Instagram. https://www.facebook.com/business/help/2593586717571940. (Accessed 07 February 2024).

Meta, 2024b. Deepfake detection challenge results: An open initiative to advance AI. https://ai.meta.com/blog/deepfake-detection-challenge-results-an-open-initiative-to-advance-ai/. (Accessed 07 February 2024).

Michael Labos, 2024. What is a Certificate Authority (CA)?. https://www.ssl.com/article/what-is-a-certificate-authority-ca/.

Microsoft Corporation, 2024a. CameraCaptureUI class. https://learn.microsoft.com/en-us/uwp/api/windows.media.capture.cameracaptureui. (Accessed 11 February 2024).

Microsoft Corporation, 2024b. Driver Signing. https://learn.microsoft.com/en-us/windows-hardware/drivers/install/driver-signing. (Accessed 09 February 2024).

MIT Media Lab, 2024. Detect DeepFakes: How to counteract misinformation created by AI. https://www.media.mit.edu/projects/detect-fakes/overview/. (Accessed 07 February 2024).

Mohanraj, R., Babulak, E., 2019. A secure energy efficient IoT based fractional correlated Bayesian data transmission in WSNs. J. Commun. Inf. Networks 4 (1), 54–66.

OriginalMy, 2024. OriginalMy. https://originalmy.com/. (Accessed 09 February 2024).

O'Sullivan, D., 2019. When seeing is no longer believing - Inside the Pentagon's race against deepfake videos. https://edition.cnn.com/interactive/2019/01/business/pentagons-race-against-deepfakes/.

Pandey, M., 2023. Senior citizen falls prey to deepfake extortion plot. https://analyticsindiamag.com/senior-citizen-falls-victim-to-deep-fake-extortion-plot/.

PassMark Software, 2024. PassMark PerformanceTest. https://play.google.com/store/apps/details?id=com.passmark.pt_mobile. (Accessed 21 February 2024).

PassMark Software, 2025. Android Devices - PassMark Rating. https://www.androidbenchmark.net/passmark_chart.html. (Accessed 07 May 2025).

Paul, K., 2020. Twitter to label deepfakes and other deceptive media. https://www.reuters.com/article/us-twitter-security-idUSKBN1ZY2OV/.

Peras, D., Mekovec, R., 2022. A conceptualization of the privacy concerns of cloud users. Inf. Comput. Secur. 30, 653–671.

Philmlee, D., 2023. Practice Innovations: Seeing is no longer believing - the rise of deepfakes. https://www.thomsonreuters.com/en-us/posts/technology/practice-innovations-deepfakes/.

Protocol Labs, 2024. IPFS is an open system to manage data without a central server. https://ipfs.tech/. (Accessed 09 February 2024).

Prover, 2024. Authenticity verification of user generated video files. https://prover.io/. (Accessed 09 February 2024).

Ray, T., 2023. Generative AI can easily be made malicious despite guardrails, say scholars. https://www.zdnet.com/article/generative-ai-can-easily-be-made-malicious-despite-guardrails-say-scholars/.

Reuters, 2024. China seeks to root out fake news and deepfakes with new online content rules. https://www.reuters.com/article/us-china-technology/china-seeks-to-root-out-fake-news-and-deepfakes-with-new-online-content-rules-idUSKBN1Y30VU/. (Accessed 07 February 2024).

Saroiu, S., Wolman, A., 2010. I am a sensor, and I approve this message. In: Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications. HotMobile '10, New York, NY, USA, pp. 37–42.

Schoning, J., Gert, A., Acik, A., Kietzmann, T., Heidemann, G., Konig, P., 2017. Exploratory multimodal data analysis with standard multimedia player. In: 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications. VISIGRAPP 2017, pp. 272–279.

Serelay Limited, 2024. Serelay trusted media capture. https://www.serelay.com/. (Accessed 09 February 2024).

Shahid, I., Roy, N., 2023. "Is this my president speaking?" tamper-proofing speech in live recordings. In: Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services. MobiSys '23, New York, NY, USA, pp. 219–232.

Singh, P., 2021. Robust homomorphic video hashing. In: 2021 IEEE 4th International Conference on Multimedia Information Processing and Retrieval. MIPR, pp. 91–96.

Soveizi, N., Turkmen, F., Karastoyanova, D., 2023. Security and privacy concerns in cloud-based scientific and business workflows: A systematic review. Future Gener. Comput. Syst. 148, 184–200.

Toews, R., 2020. Deepfakes Are Going To Wreak Havoc On Society. We Are Not Prepared. https://www.forbes.com/sites/robtoews/2020/05/25/deepfakes-are-going-to-wreak-havoc-on-society-we-are-not-prepared.

Truepic, 2024. Authenticity infrastructure for the internet. https://truepic.com/. (Accessed 09 February 2024).

US Congress, 2024a. S.3805 - Malicious deep fake prohibition act of 2018. https://www.congress.gov/bill/115th-congress/senate-bill/3805. (Accessed 07 February 2024).

US Congress, 2024b. H.R.3230 - DEEP FAKES accountability act. https://www.congress.gov/bill/116th-congress/house-bill/3230. (Accessed 07 February 2024).

Videolabs, 2024. VLC for Android. https://play.google.com/store/apps/details?id=org.videolan.vlc. (Accessed 21 February 2024).

Wong, J., 2024. Amid rise in AI deepfakes, experts urge school curriculum updates for online behaviour. https://www.cbc.ca/news/canada/education-curriculum-sexual-violence-deepfake-1.7073380.

X. Corp., 2024. Help us shape our approach to synthetic and manipulated media. https://blog.twitter.com/en_us/topics/company/2019/synthetic_manipulated_media_policy_feedback. (Accessed 07 February 2024).

Zewe, A., 2023. Explained: Generative AI. https://news.mit.edu/2023/explained-generative-ai-1109.

Zhang, X., Xu, Y., Li, R., Yu, J., Li, W., Xu, Z., Zhang, J., 2024. V2A-mark: Versatile deep visual-audio watermarking for manipulation localization and copyright protection. In: Proceedings of the 32nd ACM International Conference on Multimedia. MM '24, New York, NY, USA, pp. 9818–9827.

Zhao, Y., Liu, B., Ding, M., Liu, B., Zhu, T., Yu, X., 2023. Proactive deepfake defence via identity watermarking. In: 2023 IEEE/CVF Winter Conference on Applications of Computer Vision. WACV, pp. 4591–4600.

Zheng, Y., Cao, Y., Chang, C.-H., 2020. A PUF-based data-device hash for tampered image detection and source camera identification. IEEE Trans. Inf. Forensics Secur. (ISSN: 1556-6013) 15, 620–634.

Naser AlDuaij is an Assistant Professor in Computer Science at Kuwait University and a consultant. He received two B.S.E. degrees in Computer Science and Computer Engineering with Mathematics and Physics minors from the University of Michigan at Ann Arbor, an M.S. degree from Yale University, and an M.S., M.Phil., and Ph.D. degrees from Columbia University in Computer Science. Prior to his graduate studies, he worked as a software engineer for VMware Inc. His research interests include the Internet of Things, operating systems, security, mobile and ubiquitous computing, virtualization, and embedded systems.