sciendo

Proceedings on Privacy Enhancing Technologies ; 2022 (1):586–607

Maximilian Zinkus*, Tushar M. Jois, and Matthew Green

SoK: Cryptographic Confidentiality of Data on Mobile Devices

Abstract: Mobile devices have become an indispensable component of modern life. Their high storage capacity gives these devices the capability to store vast amounts of sensitive personal data, which makes them a high-value target: these devices are routinely stolen by criminals for data theft, and are increasingly viewed by law enforcement agencies as a valuable source of forensic data. Over the past several years, providers have deployed a number of advanced cryptographic features intended to protect data on mobile devices, even in the strong setting where an attacker has physical access to a device. Many of these techniques draw from the research literature, but have been adapted to this entirely new problem setting.

This involves a number of novel challenges, which are incompletely addressed in the literature. In this work, we outline those challenges, and systematize the known approaches to securing user data against extraction attacks. Our work proposes a methodology that researchers can use to analyze cryptographic data confidentiality for mobile devices. We evaluate the existing literature for securing devices against data extraction adversaries with powerful capabilities including access to devices and to the cloud services they rely on. We then analyze existing mobile device confidentiality measures to identify research areas that have not received proper attention from the community and represent opportunities for future research.

Keywords: personal data, cryptography, mobile, cloud

DOI 10.2478/popets-2022-0029

Received 2021-05-31; revised 2021-09-15; accepted 2021-09-16.

1 Introduction

Mobile devices have become an indispensable component of modern life. Projections estimate that there are as many as four billion smartphone users [1]. Simultaneously, smartphone connectivity, data storage, and sensing capabilities continue to improve. The adoption of these devices has complex implications for user data privacy: the smartphone vastly increases the amount of personal information that individuals can carry on their person, while simultaneously exposing that data to an unprecedented risk of theft. Portability and ease of physical access makes smartphones a target for malicious actors and law enforcement alike: to the former it provides new opportunities for criminality [2–4], and to the latter it offers new avenues for investigation and surveillance [5–8]. Further, "the cloud" has effectively become an extension of the device itself [9], and can enable data extraction even when physical devices are kept physically secure [10–12].

Scope of this work. The space of "mobile device security and privacy" is vast, and has been extensively studied in the research literature. In practice, this area includes a large set of technology areas, including software security, hardware security, access control, and network security. Further, privacy *against device software* is commonly considered [13]. An area that has received less study, however, is a specific layer of the mobile device security stack: the *cryptographic systems that protect data confidentiality*. These user-controlled cryptographic systems have increasingly been included into mobile devices as a form of "defense in depth" to provide robustness when other layers fail; we refer to this as *post-compromise data confidentiality*.

The inclusion of these cryptographic systems reflects a growing understanding in industry that traditional software-based access control has proven insufficient to stop real-world attacks — an understanding that is strongly backed by the historical record [14]. The importance of these systems has only increased as data shifts to cloud-based servers, which are vulnerable to remote attacks. Put succinctly: the focus of this work is the protection of user data stored in a locked (i.e. passcode-protected) modern mobile device, against adversaries who may physically seize the device and may target associated cloud services. We consider illicit attackers, but must also include legally compelled or forensic access – government and law enforcement ac-

***Corresponding Author: Maximilian Zinkus:** Johns Hopkins University, zinkus@cs.jhu.edu

Tushar M. Jois, Matthew Green: Johns Hopkins University, {jois, mgreen}@cs.jhu.edu

Table 1. Examples of post-compromise data confidentiality attacks and mitigations. Each mitigation defends against equivalent classes of *illicit* and *lawful* access, unable to distinguish them (§2).

Illicit Access Attack		Lawful Access Attack	Protections
Logical extraction via jailbreak/zero-day	∼	Logical extraction via UFED or GrayKey	FDE, RTFE (§5.1.1)
Cloud data retrieval via provider compromise or insider threat	∼	Cloud data extraction via court order	User-Controlled Encryption (§6.1.1)
Manual extraction via user coercion or blackmail	∼	Manual extraction via user interrogation	Deniable Encryption (§5.1.1), Trusted Hardware (§5.1.2, §6.1.2)
Physical extraction from stolen device	∼	Physical extraction from device in evidence	FDE, RTFE, Secure Deletion, Deniable Encryption (§5.1.1)
On-path protection downgrade by MitM	∼	Selective protection downgrade from compelled provider	Transparency (§6.1.3)
Brute-force password guessing	=	Brute-force password guessing	Trusted Hardware (§5.1.2, §6.1.2)

cess has deeply influenced mobile device and cloud service development, and from a technical perspective the threats to user data confidentiality are indistinguishable (§2).

Purpose of this work. While encryption technology historically emerged from the research community to industry for refinement, this trend has reversed in recent years. Deployment of mobile device encryption has become a priority for manufacturers and vendors [14]. As such, many prevalent techniques and threat models have been developed outside of the research community, and may not have been clearly documented or analyzed by researchers. In this work we attempt to rectify the problem: our goal is to capture and systematize the latest developments in this dynamic area, and to provide the research community with an overview of the current open research areas.

Contributions. In this work we systematize the current state of the art in cryptographic protection for mobile devices and their associated cloud services. Our goal is to integrate the latest developments from industrial security systems with knowledge developed by the research community, to assist the community in identifying the open research problems that exist in this area. Towards this goal we contribute the following:

- We thoroughly examine the research literature in defending mobile device data against data extraction attacks.
- We contextualize our work through extensive analysis of industry literature, as well as government documents and standards, technical documents, articles, and other sources.
- We formalize a novel threat model for mobile devices considering the subtleties of cloud integration, and an emerging defense paradigm for cloud services.
- We systematize *cryptographic confidentiality mechanisms* from the research literature and from industry solutions.
- We identify remaining technical challenges towards achieving mobile device and cloud data privacy against data extraction adversaries, providing motivation and direction for future work grounded in the realities of the modern device ecosystem.

2 Background

Post-compromise security. In this work, we analyze confidentiality mechanisms which mitigate data extraction attacks. These attacks occur in the setting where devices and cloud services may be compromised via software, hardware, or procedural vulnerabilities. Thus, we consider the extent of confidentiality which remains *given* such compromise. Post-compromise cryptography has been considered for data in-transit [15]; we consider an analogous notion for data at-rest on cloud-enabled mobile devices.

The central role of encryption. Encryption is integral to post-compromise data confidentiality. Encryption systems rely on significantly smaller "trusted code bases" (TCBs) than other security systems, and are generally designed transparently as public standards. As a result, common vulnerabilities are often not present in encryption systems, or do not directly expose encrypted data. The post-compromise setting can be viewed as a strengthening of Kerckhoff's principle [16]: confidentiality remains when the enemy not only understands, but *controls* the system, except for the secret key.

Impact on law enforcement. While provider improvements continue to enhance data confidentiality on mobile devices, they have provoked a backlash from the law enforcement community. This reaction is best exemplified by the FBI's "Going Dark" initiative [6] which seeks to increase law enforcement access to encrypted data via legislative and policy initiatives. These concerns have also motivated agencies to invest in acquiring technical means for bypassing smartphone security. This dynamic broke into the public consciousness during the 2016 "Apple v. FBI" controversy [17–19], in which Apple contested an FBI demand to bypass technical security measures. A vigorous debate over these issues continues to this day [20–22]. Since 2015, in the US alone hundreds of thousands of forensic searches of mobile devices have been executed by over 2,000 law enforcement agencies across the country [21]. These agencies use inexpensive industry tools implementing exploits to extract data [21–23]. This tug-of-war between device manufacturers, law enforcement, and forensics vendors has an important consequence for users: at any given moment, it is difficult to know which smartphone security features are operating as intended, and which can be bypassed.

Illicit vs. lawful access. In this work we are focused on threats to user data confidentiality. In the broadest sense, this requires us to consider any adversary that wishes to access user content *without the explicit consent of the user*. Inevitably, such adversaries include both criminal attackers as well as authorized law enforcement agencies who operate with judicial oversight, as well as some agencies that operate in the "gray area" between the two. Numerous works have explored this dynamic [24–26]; in the technical sections of this work we will focus primarily on the techniques used by an attacker, rather than considering the attacker's intent. Nonetheless it is important to capture the distinction.

Illicit access refers to access by a criminal attacker, while *lawful access* refers to the legally-sanctioned extraction of data from a device. Crucially, current data protection mechanisms in the literature cannot distinguish between illicit and lawful access. Table 1 illustrates this relationship. Device providers use cryptography to attempt to prevent anyone other than the user from accessing the device – inadvertently (or otherwise), these protections apply against both illicit and lawful actors. Several proposals have been advanced to bridge this divide by designing mechanisms that provide plaintext only to authorized parties [26–28]. While these "exceptional access" techniques may enable lawful access, they have been criticized as potentially reducing the security of the underlying encryption mechanism [25, 29]. Solving this problem in the general case remains an open area of research [30]. As a result, when purely considering the technical means of extraction, we must consider both types of access.

Table 2. Targeted Data in Device and Cloud Attacks

Data Type
• Complete copies of device contents, if available
Otherwise:
• Communications (messages, emails)
• Metadata (IP addresses, call logs)
• Location data (GPS, IP-based)
• Contacts/address books
• Social media data (messages, posts)
• Media (images, videos, audio)

2.1 Understanding Data Confidentiality

We consider three classes of stakeholders in mobile device data confidentiality: users; providers who manufacture and design devices and services, but also may be targets of attacks or be required to service subpoenas or searches; and adversaries, whose capabilities imply privacy requirements for data confidentiality. Table 2 provides a summary listing the types of data commonly targeted for attack by adversaries and considered sensitive by users and providers.

Adversaries. Due to the breadth and frequency of law enforcement access to mobile devices, extensive evidence exists of *what data* is targeted by law enforcement. On the other hand, illicit activity is far more opaque. As such, we can use law enforcement access as a proxy for the sensitivity of data across both types of access. From a wide array of sources [10, 14, 21, 22, 31–36] we ascertain law enforcement prioritization of data for investigation and surveillance, detailed in Table 2.

Providers. Device and cloud service providers (collectively, "providers") such as Apple and Google make privacy decisions on behalf of their billions of consumers. Notably, a cloud and device provider are often the same entity. These privacy decisions include providing user-controlled ("end-to-end") encryption [37], file encryption [37–39], and encrypted cloud storage and backup [37, 40]. However, these providers have been illicitly accessed [2] and also routinely acquiesce to legal requests for user data [41, 42], and have even allegedly scuttled plans to add provider-inaccessible user-controlled encryption based on law enforcement backlash [43]. There is risk in centralizing such control over privacy decisions into two large companies: providers become a single point of storage for a trove of user data. Smaller, privacy-focused providers exist (e.g. Purism [44]), but have failed to capture markets on the scale of Apple and Google.

Adversary-provider interaction. Three factors combine to create a unique interaction of adversaries and providers: 1) the privileged position of providers as handlers of sensitive data, 2) attacks which access provider servers, and 3) the economic incentives for providers to be perceived as trustworthy. As a result, adversaries and providers together can represent a form of honest-but-curious [45] (or "semi-honest") party, passively observing user data. We note this pattern commonly in practice, where providers turn over or unintentionally leak data [10, 14, 46]. In rarer cases, they can represent covert [47] or even fully-malicious [45] adversaries willing to undertake deliberate measures, while potentially attempting to evade detection or blame.

An emerging defense paradigm. In response to this reality, providers including Apple [48], Google [49], and even apps such as Signal [50] have turned to trusted hardware in the cloud to perform sensitive actions. If correctly configured and deployed, trusted hardware can protect data from even the provider themselves to mitigate these threats. We refer to services adopting this model as *trusted hardware-anchored cloud services*, and discuss the use of trusted hardware extensively throughout this work.

Users. Based on perception studies, users place higher significance on privacy than developers [51, 52] and are largely concerned about the privacy of long-term identifying information such as email addresses and phone numbers, their browsing content and patterns, and about privacy from voice-recognizing personal assistants [53, 54].

Who is responsible for security and privacy? While familiarity with technology correlates with use of proactive privacy safeguards [55], not all users have such expertise, and some have even received recommendations conflicting with best practices [56]. The burden of privacy should not fall most heavily on the users of devices. Protection *by default* avoids the limitations of user interventions [57–59] or providing security advice [56], but can be complex or costly. Default protections also ameliorate the scrutiny opt-in measures may attract [60]. The clear takeaway from usable security research is that adapting technical measures, rather than requiring users to adapt, improves privacy outcomes overall.

2.2 Known Extraction Techniques

In this work, we consider state-of-the-art confidentiality mechanisms in the literature and in industry. To contextualize these mechanisms, we provide an abridged historical summary of known data extraction techniques. We refer the reader to [14] for an exhaustive history.

Almost as soon as smartphones were introduced and popularized, methods for extracting data from them arose [14]. Early methods such as "chip-off" extraction of storage media pushed providers to deploy storage encryption [61], and later to cryptographically enforce user authentication by requiring the passcode to derive decryption keys [39, 62]. As attacks evolved to include exploitation of software and hardware subsystems, providers reacted by including encryption of files at runtime, such as in Data Protection on iOS [37], among other mitigations.

Physical access. Data extraction from devices is a constantly-evolving practice. Hardware-based extraction methods have continued to evolve in response to provider changes, synthesizing with software compromise to bypass hardware protections before launching an extraction exploit. For example, analysis indicates [14, 63, 64] that GrayKey, a proprietary tool made by US company GrayShift [65], compromises device subsystems via the USB Lightning port and/or the wireless interfaces on iPhones to launch a passcode-guessing attack [63, 65]. In general, these techniques require significant investment to develop, but then may be re-used indefinitely, massively impacting users' data confidentiality.

Cloud access. Cloud extraction methods have arisen as a result of increasing data storage on remote servers. Whether by password or server compromise through software exploits, insider threats, or legal requests, unencrypted data on provider servers, and encrypted data where keys are not user-controlled, are susceptible to attack, and indeed compromised in practice using these techniques [2, 9, 10, 12, 43].

3 Systematization Methodology

Organization. We divide our analysis between on-device (§5) and cloud-based (§6) approaches, despite thematically similar challenges, due to their significant variation in functional requirements. Few technical mechanisms in the literature holistically approach *post-compromise data confidentiality* either on-device or in the cloud; we therefore develop our taxonomy using interrelated but distinct problem areas. We then evaluate mechanisms via concrete criteria, referencing the directly related adversary capabilities, and highlighting limitations which motivate future work. For each section of our systematization we provide a table visually summarizing the findings of our analysis, and we close each section with motivation and directions for future research.

Criteria for evaluation. Our criteria analyze the extent to which data confidentiality approaches satisfy the privacy requirements of the post-compromise threat model, which are derived directly from the adversarial capabilities discussed in §4.

When using encryption for data confidentiality, two defining characteristics arise: what data is encrypted at a given time, and how are encryption keys stored or derived. As a result, we naturally organize the literature into encryption and authentication solutions. In the cloud setting, an additional requirement arises: transparency, that some claimed behavior, cryptographic key, or identity is correct. Within these categories, we evaluate state-of-the-art approaches in the literature and in industry solutions.

Each confidentiality mechanism may impede or even be mutually exclusive with some functionality; such trade-offs are indeed common [66]. Following [15], we extend our analysis to include usability (for users) and adoptability (for providers). For users, these factors intuitively include performance and ease of use [57–59]. For providers, they include costs of implementation, and for cloud services the costs of maintenance [67].

Informed evaluation. This work draws from both theory and practice to analyze the research literature. Our systematization is informed by consideration of *real, deployed* systems, public record documents, news stories, blog posts, and official device manufacturer and cloud provider documentation. Further, we leverage the recent effort of Zinkus et al. [14], which provides an extensive collection of analyses of industry and practice in mobile device data confidentiality.

Table 3. Classes of *Data Extraction Adversary* Capabilities

Physical (Device) Access (§4.1.1)	Cloud Access (§4.1.2)
Logical extraction	Cloud extraction
Manual extraction	Compelled data retrieval
Physical media access	Selective compromise
Compelled decryption	Compelled omissions

4 Threat Model

Due to system complexity and requirements, software and hardware security remain fundamentally difficult, with extensive vulnerabilities reported in each iteration of devices [14, 68] and in their wired [69–72] and wireless [73, 74] peripherals. Data extraction tools rely on these vulnerabilities [5, 14, 64] and have remained successful in practice [14, 21, 22, 31], therefore our analysis focuses squarely on *what confidentiality remains* when these protections are bypassed.

Defining the model. We propose a novel threat model, the *data extraction adversary*, which captures the practical realities of data extraction and post-compromise confidentiality. Prior models in the literature fall short of the full capabilities of such an adversary, either lacking consideration of the cloud [75], of physical access and software compromise [76, 77], or of the subtlety of trusted providers who may be attacked or subpoenaed [45]. Therefore, we enumerate the following real-world capabilities to elucidate this emerging threat model faced by mobile devices. Table 3 provides a summary which we expand upon in §4.1.1 and §4.1.2 to describe ten capabilities which guide our systematization. We strongly urge consideration of these capabilities in future work.

4.1 The *Data Extraction Adversary*

4.1.1 Physical Access

Physical device access can enable physical or logical extraction [14]. Physical compromise entails analysis of storage hardware, whereas logical compromise relies on exploiting device software, co-opting it to extract data. Passcode guessing attacks can even be launched via server "farms" [14]. This extensive access can undermine critical security assumptions of many confidentiality mechanisms, rendering them ineffective. In this work, we assume adversarial capabilities are eventually limited by budget – that is, we assume some costly attacks such as fully decapsulating (dissolving protective layers with acid) and exploiting running hardware is prohibitively expensive, even for such powerful adversaries. This caveat allows us to consider a wider array of technical approaches which are effective in practice. Data extraction adversaries therefore have the following capabilities relating to physical device access:

1. Logical extraction via device compromise [5, 22, 78]
2. Manual extraction via passcode compromise [64, 79]
3. Manual extraction via interrogation [31]
4. Physical media extraction [61, 80]
5. Compelled decryption via court order [81]

4.1.2 Cloud Access

Unique challenges for cloud data. Cloud data can be broadly categorized into three classes, each with unique requirements: cloud services which compute over user data (potentially in aggregate) and provide functionality to users (e.g. navigation, translation, or search); data synchronization, where devices share live data by communicating with cloud storage; and backup, where device data is stored long-term for potential future recovery. Services which compute over sensitive data naturally risk exfiltration or compromise. Data synchronization requires careful key management for shared, sensitive data. Backups require recoverability even if a device (and any encryption secrets it stored) are lost. We discuss these challenges and the extent to which existing approaches address them in §6.

Providers as targets. Cloud access via compromised or subpoenaed providers or via device or credential compromise represents a powerful capability. Mobile devices are increasingly integrated with cloud functionality for services like messaging, backup, sharing, and storage [14, 82, 83]. As a result, access to cloud services and accounts can expose troves of sensitive data [2, 9, 12, 84]. Subpoenas may target specific accounts [41, 42], or even physical locations for a period of time – including anyone who entered the location during the specified period – via controversial "geofence" warrants [85]. A natural response to these threats would be to simply disable cloud functionality on-device; unfortunately, this is a decreasingly viable solution, as the usability impacts may extend beyond users' expectations [14]. Data extraction adversaries have the following data compromise capabilities relating to cloud services, storage, and backup:

1. Cloud extraction via device compromise [10–12]
2. Compelled data retrieval via court order [42, 46, 81]
3. Cloud extraction via password compromise [11, 12]

Providers as accomplices. Whether considering a provider adversarial or simply in compliance legal requests, one must acknowledge providers' ability to materially modify or omit technical safeguards. Our analysis stops short of allowing arbitrary malicious behavior from providers, as their unique control over core device and cloud functionality leaves little room for tenable mitigations. We therefore add the following capabilities relating to data extraction adversaries and provider access to cloud infrastructure:

4. Selectively modifying protections for individual or groups of users [86]
5. Incentivizing or requiring providers to omit protections [35, 43]

5 On-Device *Post-Compromise Data Confidentiality*

Problem areas. Despite decades of research and industrial advancements, creating, deploying, and maintaining mobile devices which are simultaneously vulnerability-free and enjoy sufficient (i.e. massively marketable) performance and functionality remains out of reach in practice. For example, despite Apple's stated commitment to security and privacy [87] and nearly peerless financial resources, iOS still regularly admits vulnerabilities up to and including jailbreaks [71, 88, 89]. This is not a criticism of Apple, but rather evidence of the fundamental difficulty of securing complex systems.

Under our threat model, strong **storage encryption** with keys unavailable to potentially compromised software is the primary means of maintaining confidentiality. Encryption is deeply interrelated with authentication: as encrypted data must be accessible for correct functionality, cryptographic **user authentication** is required to mediate access to encryption keys. We analyze the research literature and real-world systems within these two problem areas to characterize the extent of protection available against data extraction adversaries, highlight limitations in this protection, and motivate and guide future work.

Table 4. On-Device *Post-Compromise Data Confidentiality*

Problem Area	System	Implemented: iOS	Implemented: Android	Adversary Capabilities: Device Comp.	Adversary Capabilities: Passcode Comp.	Adversary Capabilities: Interrogation	Adversary Capabilities: Physical Media	Adversary Capabilities: Court Order	Usability: Performance	Usability: Ease of Use	Adoptability: Impl. Cost
Storage Enc.	**Full-Disk Encryption §5.1.1**										
	***Data Protection* [37]**	✓	-	○	○	○	●	○	▫	▫	⇑
	`dm-crypt` **[38, 90]**	-	✓	○	○	○	●	○	◆	◆	↑
	Run-Time File Encryption §5.1.1										
	***Data Protection* [37]**	✓	-	◐	○	○	◐	○	▫	▫	⇑
	***File-Based Encryption* [39]**	-	✓	○	○	○	◐	○	▫	▫	⇑
	Secure Deletion §5.1.1										
	***via Trusted Hardware* [37, 91]**	✓	✓	⊙	⊙	⊙	⊙	⊙	▫	✶	⇑
	***via Encryption* [92]**	✗	✗	⊙	⊙	⊙	⊙	⊙	◆	✶	↑
	Deniable Encryption §5.1.1										
	***BurnBox* [93]**	✗	✗	●	●	●	●	○	▫	✶	⇑
	***Filesystem PDE* [94–96]**	✗	✗	●	●	⊙	●	●	◆	✶	⇑
User Auth.	**Biometrics §5.1.2**	✓	✓								
	***Fingerprint* [62, 97]**	✓	✓	○	○	○	●	○	▫	◆	⇑
	***Facial Recognition* [98, 99]**	✓	✓	○	○	○	●	○	▫	◆	⇑
	Passcodes §5.1.2										
	***Numeric Passcodes* [79, 100]**	✓	✓	○	○	○	●	⊙	▫	▫	⇑
	***Patterns* [100]**	✗	✓	○	○	○	●	⊙	▫	▫	⇑
	***Arbitrary Passphrases* [62, 101]**	✓	✓	○	○	○	●	⊙	▫	◆	⇑
	Trusted Hardware §5.1.2										
	***TrustZone* [102–104]**	-	✓	◐	○	○	◐	○	▫	▫	⇑
	***StrongBox Keymaster* [105, 106]**	-	✓	◐	○	○	●	○	▫	▫	⇑
	***SEP* [37, 107]**	✓	-	◐	○	○	●	○	▫	▫	⇑

Implemented: ✓ = by the provider; ✗ = no 1st-party implementation; - = N/A
Adversary Capabilities: Mitigates... ○ = never; ◐ = partially; ⊙ = conditionally; ● = completely
Performance: Impact... ▫ = minimal; ◆ = noticeable; ✶ = significant
Ease of Use: Requires... ▫ = no interaction; ◆ = user opt-in; ✶ = user intervention or configuration
Implementation Cost: Requires... ↑ = specialized software; ⇑ = specialized hardware & software

5.1 Evaluation

5.1.1 Storage Encryption

To protect data on a device from extraction, whether by manual analysis of physical storage media [61, 80], or (more commonly [14]) by logical extraction using a compromised software component or operating system kernel [5, 64, 78, 108, 109], data must be encrypted with keys inaccessible to the extractor. Cryptography is leveraged to create systems with a variety of properties ranging from data indistinguishability to deniability, as discussed in this section.

Full-disk encryption (4.1.1–4). FDE enables a disk to be transparently encrypted and decrypted, providing protection to data at rest when a device is powered off [110]. Cryptographic cipher configurations have emerged and been standardized [111] and implemented [37, 38] by providers to address challenges of efficiency and security [110, 112, 113], most notably a lack of additional space in software-only implementations to store ciphertext-adjacent data such as encrypted keys, nonces, or authentication tags, and other failures in hardware-backed FDE [114]. Legal analysis indicates that encryption keys derived from secrets the user remembers cannot be compelled in some jurisdictions [81].

However, the success of FDE in mobile devices is varied [14, 115] due to the design of decrypting all data after device startup (unlike run-time file encryption), and adaptation by adversaries. Two key adaptations are 1) not allowing mobile devices to discharge [5, 116, 117] and 2) obtaining passcodes/passwords via keyloggers, extortionary actions and court orders, searches, or even imprisonment to compel divulging of passwords [115]. These adaptations essentially negate the protective benefits of FDE, especially considering that modern mobile devices are likely powered-on at almost all times.

From a usability perspective, FDE minimally delays startup time. Cryptographic accelerators [37, 118] are a vital optimization to maintain performance and power efficiency for FDE, and efficient alternatives have been explored for devices lacking such hardware [119]. Regarding adoptability, implementations of FDE are widely available, such as the Linux kernel module `dm-crypt` [90, 120], and providers use these or implement their own for mobile devices [37, 38], although in some cases leave them disabled by default apparently for performance reasons [121]. Implementation is not necessarily straightforward, with misconfigurations exposing immense amounts of data as recently as 2019 [114], and thus implementation complexity cannot be disregarded.

Run-time file encryption (4.1.1–1,4). RTFE brings many advantages of FDE to the powered-on, running system by enabling on-demand decryption and re-encryption of data [37, 122, 123]. The advantages of maintaining data encryption after startup are most clearly reflected in their effect on data extraction: in many cases, RTFE-encrypted data is often protected from extraction in practice, often leaving adversaries only able to access data already decrypted at the time of device seizure [14]; however, this protection depends heavily on RTFE configurations for different types of data. As with FDE, RTFE is seemingly also protected from legal compulsion in some jurisdictions.

RTFE has one key consideration regarding usability: after device lock, any data configured to be re-encrypted is no longer available for use. As a result, apps and lock screen interfaces must be adapted to account for inaccessible data. This issue is largely mitigated by the ease of user authentication (discussed later in this section) and by user interface design. Finally, as with FDE, the advantages of RTFE come with substantial performance costs which must be mitigated by specialized hardware. Cryptographic accelerators are key components of RTFE due to the just-in-time decryption design.

Secure deletion (4.1.1–1,2,3,4,5). Secure deletion is critical for resilient confidentiality, as deleted files may contain as much or more information (by nature of having been deleted) than existing files on a device. Secure deletion is generally achieved via encryption, using cryptographic accelerators, with random keys which are expunged [124]; recent work enables this on mobile devices (which commonly use NAND flash storage [125] which entails specific challenges) including without system-level privileges [92]. Alternatively, functionality of the underlying hardware can provide secure deletion [126]. Apple iOS provides a mechanism referred to as "effaceable storage" for secure deletion of encryption keys [37]. This mechanism uses a combination approach: encryption of data plus hardware functionality to securely delete encryption keys. Android allows secure deletion of keys in trusted hardware via "rollback resistance" [91]: by preventing rollback of a key deletion operation, an encryption key is purged.

Secure deletion can mitigate the impact of data extraction, whether by software, physical, or even passcode compromise, albeit with three significant limitations: first, a user must know a search is coming to preempt it; second, deleted data is lost; and third, it is not always clear when data is securely deleted. The second limitation can be addressed via cloud backup, which itself has risk. Recoverable secure deletion has been recently contributed to the literature [93], but still suffers from the limitation of preemption, and further, creates recovery data to be stored remotely which is susceptible to compromise. Secure deletion is also a critical component of run-time file encryption, as to re-encrypt data, keys and decrypted copies of data must be irrecoverably evicted from storage. Securely deleted data is additionally protected from court compulsion, although this risks charges of destruction of evidence. However, such considerations are beyond the scope of this technical paper.

Once implemented in software, potentially leveraging specialized encryption hardware, secure deletion operates transparently to the user. However, considering the third mentioned limitation, it is not always clear that data has been deleted, and may simply be hidden from the user interface for later deletion [14].

Deniable encryption (4.1.1–1,2,3,4,5). Plausibly deniable encryption (PDE) is a line of research which examines how encrypted storage systems can be made *deniable*: storage which can feasibly be hidden from an adversary performing a search which potentially includes device access and even compelled decryptions. Approaches to PDE generally involve metadata hiding [127]: preventing filesystem metadata from betraying the location or existence of hidden data using encryption or apparently-unused storage areas.

PDE can provide a large degree of protection against all kinds of device compromise and search, although this protection is limited in that a user may be required to successfully deceive an adversary, fabricating that no further data exists. As noted by [93], this can be a significant limitation, especially for users under duress. Such deniability can even extend to legal compulsion, noting risks of perjury. PDE schemes can even allow "dummy" passcode disclosure to revel a subset of protected data to satisfy an authority [124].

Although neither of the major platforms provide PDE, various approaches have been proposed in the literature. Recent work has brought deniable encryption to NAND flash storage systems [94], generally challenging due to wear-leveling systems [125]. Since then, contributions have provided "user-friendly" switching between deniable and regular encrypted storage [95] and methods for plausible deniability of the very presence of a PDE system on a device [96]. However, even with cryptographic hardware, these systems still burden users with noticeable overhead.

Trusted hardware (4.1.1–1,4). Trusted hardware enables storage and usage of secrets such as encryption keys without leaking them to potentially compromised device software and hardware. Particularly, secrets derived from user authentication (discussed later in this section) are protected from the rest of the system, making storage encryption resilient to attack; for this reason we list trusted hardware under user authentication in Table 4 despite its relevance across categories. Trusted hardware requires embedded microcontrollers protected from a wide range of attacks, from physical reverse-engineering to software attacks launched by a compromised kernel. These protections include physical hardening and tamper detection [37], micro-architectural defenses [128, 129], replay protection [37, 102, 130], encrypted RAM [37, 102], and minimal "trusted computing bases" (TCBs) [131] reducing code size to minimize possibility of vulnerability. Trusted hardware can then be leveraged for storage encryption to handle decryption keys.

Trusted hardware has been implemented in iOS and Android via specialized hardware and software, referred to respectively as the Secure Enclave Processor (SEP) and its operating system SEPOS, and StrongBox on Android [37, 105, 132]. Some older Android devices instead use TrustZone implementations such as Trusty [102, 103]. Trusted hardware has also been developed independently in the literature: Keystone [133] is a realization of trusted hardware relying on end-to-end hardware verification on the open RISC-V platform [134].

Once deployed, trusted hardware can operate transparently to the user, and with minimal performance impact. For storage encryption, a critical part of this performance comes from hierarchical key management: trusted hardware generally only stores keys at the root of large hierarchies otherwise stored in main (unprotected) memory. However, these hierarchies are encrypted with the root keys, and thus are protected without overburdening the trusted hardware. The protections trusted hardware relies upon generally scale in terms of performance up to their usage in these embedded platforms, but often not beyond: for example, encrypted RAM is used for embedded systems [135], but the performance cost this entails has precluded more general use [136, 137]. Trusted hardware also has implications for maintaining the integrity of core system components. For example, a device can employ a hardware root of trust to verify the digital signatures of its firmware and bootloader [37, 138]. The device's trusted hardware would stop execution if an unsigned bootloader or firmware update is run, preventing a potentially compromised operating system from accessing user data.

5.1.2 User Authentication

Authentication on-device is the necessarily complement to encryption: when data access is required, secure authentication mechanisms mediate the derivation or release of encryption keys.

Biometrics (4.1.1–4). Biometrics have been studied in the biology, statistics, and criminology literature for decades [139]. However, with the advent of digital biometrics on mobile devices [62, 97–99, 140], their use has rapidly transitioned from exceptional to ubiquitous.

Android and iOS both provide biometric authentication in the forms of fingerprint and facial recognition [37, 62, 97, 99]. Apple claims [62] that biometrics encourage users to select stronger passcodes, but this claim has been questioned in the literature [140]. Recent advances in biometrics have enabled authentication via hand geometry [141], palm print [142], iris [143–145], periocular [146, 147], and even electrocardiographic identification [148]. Despite this diversity, biometrics all rely on the body and so their threat models and achieved security vary only slightly.

Despite inherent (sensing) hardware complexity and the challenge of securely storing fingerprint and facial recognition data [98], modern devices have successfully deployed trusted hardware to implement biometrics which has resulted in more seamless authentication. In combination with trusted hardware [37, 97, 101], biometrics enable protection against logical and physical extraction attacks which lack access to the user. An important caveat to biometric authentication is that while passcodes and other memorized secrets generally cannot be legally compelled in some jurisdictions (e.g., the Fifth Amendment in the US [81, 149]), biometrics generally can be, incurring notable risk.

Passcodes and variants (4.1.1–4,5). Passcodes and similar systems have become the norm for mobile device authentication [37, 100]. When combined with the secure processing, attempt-limiting, and time-delaying functionalities of trusted hardware, passcodes become a convenient and secure mechanism from which encryption keys can be derived to protect data. Further, passcodes and other memorized secrets are often protected from legal compulsion via court order due to the Fifth Amendment or similar laws [81, 149].

iOS and Android support authentication in the form of PINs, pattern-based codes, and arbitrary passcodes [37, 100]. Short passcodes such as historically default 4-digit passcodes on iOS [150] and even 6-digit modern defaults [37] and patterns on Android offer limited protection against brute-force attacks [64, 80, 151, 152], common passcode guessing [79], or other similar attacks [153, 154]. Indeed, when trusted hardware fails to enforce guessing limits and delays, long passcodes are a last line of defense [31, 64]. Unfortunately, choosing weak passcodes is a common user behavior [79, 155].

Trusted hardware (4.1.1–1,4). Specifically for user authentication, trusted hardware can be leveraged to safely handle user authentication data such as a passcode or biometric measurement. Trusted hardware can also facilitate time-based delays (such as using PBKDF2 [156]), enforce guessing limits, and even erase data when brute-forcing is detected [14, 37]. This adds an additional challenge for data extraction, requiring attackers to bypass trusted hardware [64] to perform password guessing. The high performance of these mechanisms is derived from direct integration with biometric measurement hardware [37, 97, 101]. Refer to §5.1.1 for general evaluation of trusted hardware in the on-device setting, which we omit here for brevity.

5.2 Analysis

Extensive engineering and research effort has been undertaken to secure mobile devices. However, there are practical limitations to the *post-compromise data confidentiality* of modern mobile devices, and these limitations highlight potential directions for future research.

Under-utilized run-time file encryption. Although RTFE is a powerful mitigation for *post-compromise data confidentiality*, it is significantly hampered in practice by insecure defaults. iOS Data Protection [37] and Android file-based encryption [39] defaults to decrypting data after the *initial* user authentication since startup, or "after first unlock" (AFU). In practice, data protection classes on iOS are integral to whether or not data is successfully extracted via logical compromise [14]. By placing Data Protection classes in the hands of developers, some apps can opt-in to protect their users, but lacking secure defaults this mitigation does not apply to vast amounts of user data [37]. App instrumentation and file access tracking could dramatically improve RTFE by automatically opting unused data into more encrypted Data Protection classes without impacting performance.

Passcodes as single points of failure. If a device is unlocked, or worse, if the passcode is known, device access is nearly unbounded [157]. There is clearly much work still to be done in understanding user decisions and encouraging stronger passcodes, but the burden should not lie solely on users. Biometrics have had a massive impact on the way people access their devices, but seem not to have improved underlying passcode strength and currently face legal compulsion in the US [79].

Alternative authentication schemes can provide protection for data even after passcode compromise. These schemes rely on relocating secrets to other parties/devices which can be retrieved for authentication.

For example, secret sharing schemes [158] allow a secret to be divided into individually useless parts, and recombined efficiently. However, the twin problems of distributing shares and the frequency of authentication render such techniques impractical. Users can opt to relocate secrets to secure yet accessible external devices instead, such as cryptographic hardware tokens (also called 2FA tokens, for two-factor authentication) [159]. This approach can protect data in case of passcode compromise, but confers costs of inconvenience due to additional overhead, and potential data loss if the hardware device is lost or fails without a backup. Presently, mobile platforms lack support for 2FA in device unlock (despite supporting it in other cases, e.g. web or app authentication [160]). Any such implementation would likely rely on trusted hardware to perform authentication given the user passcode/biometric in addition to the second factor.

Vulnerabilities in trusted hardware. Trusted hardware is heavily relied upon for *post-compromise data confidentiality*. There is evidence that trusted hardware on mobile devices has been exploited in practice [14, 64] to enable passcode brute-force attacks. Additionally, Apple recently updated the trusted hardware components which mitigate replay attacks [37], and thus we can infer that previous brute-force exploits [64] may have exploited this subsystem. The SEP API has been fully reverse-engineered [107], and therefore may be an appropriate target for formal specification and verification, in an effort to provably mitigate such vulnerabilities in the future.

Formal lower-bound analysis of deniable encryption. The PDE literature includes extensive adversarial formalizations and security analysis. However, future work in lower-bounds proofs may guide PDE design towards optimality. Following the insights of Tyagi et al. [93], analysis of oblivious RAM (ORAM) [161] may offer promising analogues to aid this work.

Continuous authentication. Continuous authentication systems synthesize biometric and other sensor measurements to confirm that the authenticated user is still operating the mobile device. Google Smart Lock [162] uses location, proximity detection through motion, reachability of trusted network devices, and voice recognition to prevent locking the device when the user is continuously identified. User behavior such as touchscreen interaction, walking gait, and app usage can be similarly used for authentication on mobile devices [163]. Future work could apply these mechanisms to run-time file encryption or to authenticate sensitive user actions.

Cryptographic warrant enforcement. Law enforcement search of mobile devices is commonly governed by warrants which approve the extent of access allowed. There are no extant cryptographic methods for efficiently enforcing such a policy, but achieving such a system is an open problem of great promise. Witness encryption [164] could be leveraged to decrypt only data which falls under a set of predicates determined by a warrant. If this or another cryptographic system could be efficiently realized, the need for law enforcement transparency could be significantly reduced. Initial work in the literature explores this possibility [30].

6 Cloud *Post-Compromise Data Confidentiality*

Problem Areas. We analyze cloud protection mechanisms from the research literature and from industry which protect data targeted by extraction adversaries (§2.1) by addressing extraction capabilities (§4.1.2). **User-controlled encryption (UCE)** enables sensitive data to remain protected, as cloud providers lack access to encryption keys. To maintain functionality, data must be decrypt-able, and thus requires mediated access via cryptographic **user-to-cloud authentication (UtCA)**. Finally, cloud services may be used to execute sensitive functionality. *Transparency* enables a user/device to ensure correct execution or minimize the impact of malicious parties. Exploiting device trust in providers, a data extraction attack could e.g. replace an HSM IP address with that of an unprotected server; transparency in the setting of *post-compromise data confidentiality* replaces trust in providers with verifiability, specifically of cryptographic keys and identities which are required to bootstrap further security. In the evaluation which follows, we analyze the research literature and real-world implemented systems within these three problem areas to characterize the extent of protection available against data extraction adversaries, highlight limitations, and motivate and guide future work.

Table 5. Cloud *Post-Compromise Data Confidentiality*

Problem Area	System	Implemented: Apple	Implemented: Google	Adversary Capabilities: Device Comp.	Court Order	Password Comp.	Selective Mod.	Omission	Usability: Performance	Ease of Use	Adoptability: Impl. Cost	Maint. Cost
UCE	**Key Agreement §6.1.1**											
	Apple Handoff **[165]**	✓	-	○	●	⊙	○	○	▫	▫	↑	↓
	Trusted Hardware §6.1.1											
	Titan **[166]**	-	✓	◐	●	○	◐	◐	▫	▫	⇑	↑
	iCloud Keychain **[48]**	✓	-	◐	●	○	◐	◐	▫	▫	⇑	↑
UtCA	**PAKE §6.1.2**											
	OPAQUE **[167]**	✗	✗	○	●	○	◐	◐	▫	▫	↑	↓
	SRP **[37, 168]**	✓	✗	○	●	○	◐	◐	▫	▫	↑	↓
	2FA §6.1.2											
	OTP **[169, 170]**	✗	✓	●	○	●	○	○	▫	◆	↓	↓
	PKCS #11 **[159]**	✗	✗	●	⊙	●	○	○	▫	✶	↑	⇓
	Apple 2FA **[171]**	✓	-	●	⊙	●	○	○	▫	◆	↑	↓
Transparency	**Decentralization §6.1.3**											
	Blockchain **[172]**	✗	✗	◐	◐	○	●	●	✶	✶	↑	⇕
	P2P Networks **[173]**	✗	✗	◐	◐	○	●	●	✶	✶	↑	↕
	Transparency Logging §6.1.3											
	Trillian **[174]**	✗	✗	◐	-	○	◐	●	◆	✶	↑	↑

Implemented: ✓ = by the provider; ✗ = no 1st-party implementation; - = N/A
Adversary Capabilities: Mitigates... ○ = never; ◐ = partially; ⊙ = conditionally; ● = completely
Performance: Impact... ▫ = minimal; ◆ = noticeable; ✶ = significant
Ease of Use: Requires... ▫ = no interaction; ◆ = user opt-in; ✶ = user intervention or configuration
Implementation Cost: Requires... ↑ = specialized software; ⇑ = specialized hardware & software; ↓ = neither
Maintenance Cost: ⇓ = negligible; ↓ = low; ↑ = high; ↕ = variable; ⇕ = highly variable

6.1 Evaluation

6.1.1 User-Controlled Encryption for Cloud Data

Encryption technologies, such as symmetric ciphers [175], are well-understood and robustly implemented. As a result, the problem of user-controlled encryption in the cloud reduces to one of encryption key management. Provider servers may be compromised by data extraction adversaries, thus encryption keys must be kept exclusively on-device or within trusted hardware. For cloud services which compute over user data, risk must either simply be accepted or mitigated through complex techniques such as homomorphic encryption [176]. For data synchronization between user devices or backup services, approaches discussed in this section offer trade-offs between confidentiality and complexity, ultimately providing at best partial protection from extraction attacks.

Key agreement (4.1.2–2,3). For scenarios where cloud data is stored only for purposes of synchronizing multiple live devices, it is possible to derive high-entropy keys between devices using cryptographic key agreement techniques, such as those already used for end-to-end encrypted messaging systems. Indeed, Apple already supports the derivation of pairwise keys as part of its Handoff service [37], which is a system that synchronizes data between devices via Bluetooth and

WiFi. The advantage of using this approach to deriving keys for cloud services is that the cloud provider never learns the shared device keys. This assumes that the provider does not selectively modify or omit the key agreement protocol, e.g. by tampering with the identity and key distribution services. See §6.1.3 for more on preventing such attacks. The disadvantage of this approach is that it is useful primarily for synchronizing data between functioning devices: this approach does not support device backup in the event that all user devices (and hence keys) become unavailable.

Trusted hardware (4.1.2–1,2,4,5). Trusted cloud hardware allows users to compute privately even when they do not fully trust a cloud provider [177, 178]. In practice, trusted cloud hardware takes the form of hardware security modules (HSMs) deployed by providers, which generally execute pre-determined functionalities such as encryption or password verification [37, 49, 166, 179]. Trusted hardware enables user-controlled encryption of cloud-stored data without requiring users to memorize or store high-entropy secrets, either through *password strengthening* [180–183], cryptographically mixing the user password (or a derivation thereof) with high-entropy secrets resulting in a secure encryption key unknown to the cloud provider, or via password-authenticated key exchange (§6.1.2). A key feature of these devices is that they can enforce *guessing limits* to prevent dictionary attacks. This approach enables *user-controlled* encryption while mitigating the need for users to memorize or store high-entropy secrets, which may be an untenable UX requirement [57].

In Apple iCloud, trusted hardware is also used to manage a list of "trusted devices" per-account, a mechanism which can be used to share or revoke access to encryption keys among user devices without disclosing them to Apple [37, 48]. Further, HSMs can cryptographically authenticate their code. Therefore, if the user trusts that Apple correctly implements iCloud Keychain functionality [179], they can rely on this authentication to prevent modifications to that functionality. However, the user has no way to verify they are communicating with an Apple HSM, and thus must implicitly rely on the provider. This caveat has particular relevance in light of Apple's agreement with the Chinese government to move iCloud encryption keys to Chinese servers [86]. For Google Mobile Services (GMS), the Titan HSM system [49, 166] is used to protect some mobile device backups [40] using the user's device authentication credential in a similar entropy-stretching design.

The use of trusted cloud hardware poses many challenges for providers. These include the need to prove to users that the hardware is correctly implemented and cannot be re-programmed, as well as many additional deployment challenges around replication and availability (see §6.2 for further discussion).

6.1.2 User-to-Cloud Authentication

Password-authenticated key exchange (4.1.2–2,4,5). PAKE is a cryptographic protocol in which communicating parties with a shared, low-entropy secret (a password) are able to securely derive a shared high-entropy secret key. In the cloud setting, asymmetric PAKE (aPAKE) is of particular relevance: the user authenticates with their password, which has been previously registered with the cloud (ideally in trusted hardware, per §6.1.1) without revealing it, while the cloud is authenticated either implicitly or through public-key infrastructure. Recent cryptographic results in (a)PAKE have achieved seemingly optimal communication complexity [184], and even pre-computation attack resistance in the OPAQUE protocol [167].

Apple has implemented one such aPAKE in iCloud [37]. iOS users execute the Secure Remote Password protocol (SRP) [168] rather than traditional password authentication to authenticate to iCloud HSMs. As compared with OPAQUE, SRP is not pre-computation attack resistant, and lacks a formal proof of security with standard assumptions. The SRP interaction is transparent to users and confers negligible performance differences, but fully hides the user's iCloud password from Apple servers. Implementing SRP required up-front investment from Apple, but no significant additional upkeep. Although in theory users could notice if Apple servers discontinued use of SRP, the technical knowledge required implies that Apple could, if compelled, selectively remove this feature covertly. However, this mechanism partially maintains security against server-side modifications or omissions as it does not leak the user password.

Two-factor authentication (4.1.2–1,2,3). 2FA for cloud services can maintain security in the event of password or device compromise. 2FA via cryptographic hardware tokens has been standardized, and the research literature contains extensive formalization and analysis of these standards [159, 185, 186].

Google has implemented 2FA in multiple forms, allowing for hardware tokens, and time- and HMAC-based one-time passwords (OTP) [187]. These implementations provide users opt-in measures to increase the security of their cloud accounts, however, OTP secrets are likely accessible to providers by design [169, 170]. These mechanisms require little upkeep, and open-source implementations are readily available. Apple's implementation of 2FA stands out, in that it interacts with their trusted device infrastructure [171] and therefore *may* rely on secret stored in user devices rather than in the cloud, however, this functionality is not documented.

Trusted hardware (4.1.2–1,4,5). Specific to user-to-cloud authentication, cloud HSMs enable secure storage of user registration information. However, novel and compelling challenges emerge when considering questions of scale: multi-HSM consistency over user authentication data, for example to support load balancing amongst HSMs, has only recently received treatment in practice, and has received almost no formal analysis. Load balancing among HSMs requires secure sharing of not only (static) user authentication data but also (likely dynamic) state associated with the user, such as remaining login attempts. Signal, the end-to-end encrypted messaging platform, recently applied distributed consensus [188, 189] to achieve HSM consistency for encrypted backups [50]. Apple pre-provisions HSM clusters rather than dynamically scaling [179]. Refer to §6.1.1 for general evaluation of trusted cloud hardware which we omit here for brevity.

6.1.3 Transparency of Keys and Identities

Transparency logs (4.1.2–4,5). Providers are trusted to deploy HSMs and manage encryption keys on behalf of their users. Therefore, data extraction attacks launched by or with access to provider systems may exploit this trust. For example, a device could be given the address of a malicious server in place of an HSM. Transparency logs are designed to enforce honest provider behavior by requiring validation against a public log. In the HSM example, a device would be able to validate the identity of the server against such a log. This enforcement has limitations, as new log entries can not necessarily be validated in real-time or with high degrees of certainty, but the approach helps users ensure that their view of cloud systems is consistent. These systems do not alone explicitly prevent surreptitious behavior, but combined with device-side verification of public transparency data, transparency logs can mitigate covert attacks including those against key management, software update delivery, and HSM provisioning.

Public ledgers [190] can provide resilient, distributed storage for transparency logs, and facilitate enforcement of transparency. This model has seen success in Google's Certificate Transparency (CT) [191], a system for auditing the distribution of TLS certificates. Google has since generalized their implementation to support general verifiable data structures [174]. Transparency logs are also used to validate the provenance of keys [192, 193] and software packages [194]. Transparency log systems require initial implementation investment, but can provide verification by default and can rely on decentralized networks to avoid impacting performance for users or cloud providers.

Decentralization (4.1.2–1,2,4,5). Another approach to enforcing transparency is to outsource services to decentralized peer networks. With sufficient decentralization, individual server compromise is ineffective, and court orders become infeasible to coordinate and enforce. However, decentralization can incur significant performance overhead, e.g. by collecting shards of user data from nodes potentially across the world. Decentralized networks often require consensus, implying further overhead.

Due to complexity and requiring coordination, decentralization is rarely used to replace cloud services. Apple has implemented a peer-to-peer service for finding lost devices via Bluetooth broadcasts while maintaining a degree of location privacy [195, 196]. Current work is largely focused on blockchains, in which append-only public ledgers are stored using consensus protocols [190, 197]. Whether blockchain-based [172] or not [173], decentralized data storage systems have potential to mitigate the privacy concerns of provider-controlled cloud servers while relieving providers of maintenance costs.

Trusted hardware (4.1.2–2,4,5). Specific to transparency, trusted hardware facilitates confidentiality by cryptographically verifying code. Providers leverage this feature to remove their own ability to surreptitiously access data or modify functionality: Apple, for example, claims to have destroyed the code signing keys for iCloud Keychain HSMs [179]. However, users generally cannot determine that they are communicating with a correct HSM in practice: thus, this design only provides partial mitigation against many data extraction capabilities. HSMs rely on user authentication to provide data records, and therefore password compromise completely negates their benefit. Refer to §6.1.1 for general evaluation of trusted cloud hardware which we omit here for brevity.

6.2 Analysis

Cloud data confidentiality mechanisms center around removing implicit trust in providers. Due to the practical realities of data extraction adversaries, remaining trust creates disconnects in data confidentiality which imply promising directions for future work.

Open challenges for cloud data. Sensitive data in the cloud faces numerous challenges pertaining to confidentiality. Cloud services which compute over user data to provide functionality represent direct risk to confidentiality unless complex approaches such as fully-homomorphic encryption [176] are adopted by providers. Data synchronization services must authenticate devices and establish encryption keys from low-entropy user credentials via aPAKE or password strengthening, and must safely store these keys (generally, via trusted hardware). Implementing the key agreement model of Firefox Sync [198] is a promising engineering solution to mitigate a subset of extraction attacks at relatively low cost. Cloud backups, a prevalent target for subpoena [14, 21], must also be safely stored while being recoverable even if encryption secrets, devices, and/or passwords are forgotten or lost. Trusted hardware, in combination with other mitigations, has significantly improved these open problems in practice, but creates new challenges: HSM reprovisioning and scaling have been initially explored but lack formal analysis. As a result, maintaining data confidentiality in the cloud in the post-compromise setting remains a promising direction for impactful future work.

Data recoverability and backups. The realities of portable devices and human nature create risk of accidental data loss. As a result, providers offer cloud backup services, encrypting these backups with keys they control [82, 199]. Ostensibly this is to mitigate loss due to forgotten passcodes, but additional pressure from law enforcement is also allegedly a factor [10, 43]. Recoverability seems to imply a dilemma for cloud backup encryption: backups cannot be simultaneously fully recoverable (including upon passcode loss) yet provider-inaccessible (through user-controlled encryption). Biometric-derived encryption keys may hold promise in resolving this dilemma, and a number of user interface solutions might mitigate its impact. Due to the high frequency of subpoena [10, 21] cryptographic tools such as functional encryption [200] may also hold promise in improving privacy while enabling lawful search.

Trusted hardware. In practice, trusted hardware plays a vital role in addressing the duality of trust in providers. By deploying trusted hardware, a provider can remove their own ability to circumvent privacy mechanisms. As such, trusted hardware is thoroughly examined in this systematization, and we identify and formalize the *trusted hardware-anchored cloud services* mitigation paradigm as an emerging pattern. Analyzing the extent and limitations of this paradigm is itself an opportunity for future work. Instantiations of this model in practice still implicitly rely on providers: users generally cannot verify HSM instances and cannot distinguish honest reprovisioning from an attack. Closing these gaps represents multiple lines of promising and impactful future work.

Remaining trust in providers. Finally, our threat model calls into question services in which the provider acts as an identity broker, such as with Apple iMessage, Apple FaceTime, and Google Duo, and of security features such as iCloud Keychain. Existing designs leverage the provider to provide efficient distribution of cryptographic material between peers. However, centralized designs, while relatively performant, require trust in the provider. In some cases, security features are intentionally omitted to enable functionality: iCloud Backup [82] and Android Auto-Backup [199] are stored encrypted with keys held *by the providers*, which allows them to restore user data even if a user forgets their passcode. Worse still, trusted hardware attestation keys have leaked [129]. Myriad opportunities exist for further reducing unneeded trust in providers: verifying peer identity, validating server behavior or committing it to transparency logs, deploying user-controlled encryption, or decentralizing, even if only among groups of providers. Eliminating this trust will significantly improve resilience of device and cloud systems against even the strongest covert and malicious adversaries.

7 Conclusion

Modern mobile devices, with their storage, sensing, and connectivity capabilities, are of unparalleled value to adversaries seeking sensitive personal data. Software security techniques in industry and the literature continue to provide complex and comprehensive protections against many attacks, and yet vulnerability-free mobile operating systems remain out of reach. *Post-compromise data confidentiality* has therefore risen to paramount importance.

We contribute a novel threat model, informed by the realities of the mobile device ecosystem and formalize an emerging defense paradigm for cloud service. We systematize research, providing a thorough evaluation of the current state of both the research literature and engineering results, as well as motivation and directions toward open questions. The open questions we suggest relate to concretely improving privacy and security through technical measures across various fields of research.

It is our hope that this work facilitates collaboration across academia, industry, and policymaking, and promotes not only research but tangible impact towards *post-compromise data confidentiality* for users of mobile devices.

For researchers. In applying cryptographic, security, and privacy research to the mobile setting, researchers should consider the numerous and subtle effects of cloud integration and trusted hardware. As we have enumerated, there are extensive opportunities for novel work across the theory and practice of these emerging settings, and the cloud can provide both significant benefits and unforeseen risks to applied research.

For providers. Users rely on providers for safety and peace of mind. In many regards, providers rise to this mantle by improving security and privacy in devices and services, and contributing to research. In this work we identify a number of gaps between providers' implementations and *post-compromise data confidentiality*. Working to address these will create substantial privacy and security benefits for users.

For policymakers. Device and cloud protections, while increasingly robust, are bypassed in practice by lawful and illicit actors alike. Historically, government standards and practices have improved security and privacy, but in recent years we observe a concerning reversal of this trend. A wide range of user data is readily extractable upon legal request. Rather than weakening or banning strong, user-controlled encryption, policymakers should encourage providers to bolster existing defenses and work with partners in academia and industry to find solutions to incorporate lawful access with strong cryptography.

8 Acknowledgments

The authors would like to thank Dr. Mike Rushanan, and Emma Weil and Dr. Harlan Yu of Upturn, for their insightful feedback.

The authors received support from the National Science Foundation under awards CNS-1653110, CNS-1801479, CNS-1955172, and from a Google Security & Privacy Award as well as an ONR award. Additionally, this material is based upon work supported by DARPA under Contract No. HR001120C0084. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA.

References

[1] S. O'Dea. Number of smartphone users worldwide from 2016 to 2021. https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/, 9 2019.

[2] Feliks Garcia. iCloud celebrity nude leak. *Independent*, 2016.

[3] Paul Ruggiero and Jon Foote. Cyber Threats to Mobile Phones. https://us-cert.cisa.gov/sites/default/files/publications/cyber_threats_to_mobile_phones.pdf, 2011.

[4] DHS. Study on Mobile Device Security. https://www.dhs.gov/sites/default/files/publications/DHS%20Study%20on%20Mobile%20Device%20Security%20-%20April%202017-FINAL.pdf, 2017.

[5] Vladimir Katalov. The Art of iPhone Acquisition. https://blog.elcomsoft.com/2019/07/the-art-of-iphone-acquisition/, 7 2019. Accessed 2020-08-04.

[6] James Comey. Going Dark. https://www.fbi.gov/news/speeches/going-dark-are-technology-privacy-and-public-safety-on-a-collision-course, 10 2014. Accessed: 2020-07-19.

[7] Craig Timberg, Drew Harwell, and Reed Albergotti. Update your Apple devices now. New Pegasus hack prompts company to issue new software to fix iMessage vulnerability. https://www.washingtonpost.com/technology/2021/09/13/pegasus-spyware-new-exploit-apple/, 9 2021.

[8] The Wire Staff. Spyware Like Pegasus Is 'Incompatible With Human Rights': UN's Michelle Bachelet. https://thewire.in/world/spyware-pegasus-incompatible-human-rights-un-michelle-bachelet, 9 2021.

[9] Tobias Matzner. Why privacy is not enough privacy in the context of "ubiquitous computing" and "big data". *Journal of Information, Communication and Ethics in Society*, 2014.

[10] Privacy International. Cloud extraction technology. https://privacyinternational.org/long-read/3300/cloud-extraction-technology-secret-tech-lets-government-agencies-collect-masses-data, 1 2020.

[11] Oleg Alfonin. Accessing iCloud With and Without a Password in 2019. https://blog.elcomsoft.com/2019/07/accessing-icloud-with-and-without-a-password-in-2019/, 7 2019. Accessed 2020-09-10.

[12] Cellebrite. Unlock cloud-based evidence to solve the case sooner. https://www.cellebrite.com/en/ufed-cloud/, 9 2020. Accessed 2020-09-10.

[13] Chad Spensky, Jeffrey Stewart, Arkady Yerukhimovich, Richard Shay, Ari Trachtenberg, Rick Housley, and Robert K Cunningham. Sok: Privacy on mobile devices–it's complicated. *Proceedings on Privacy Enhancing Technologies*, 2016(3):96–116, 2016.

[14] Maximilian Zinkus, Tushar M. Jois, and Matthew Green. Data security on mobile devices. https://arxiv.org/abs/2105.12613, 2021.

[15] Nik Unger, Sergej Dechand, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, and Matthew Smith. SoK: Secure Messaging. In *IEEE S&P '15*. IEEE, 2015.

[16] Claude E Shannon. Communication theory of secrecy systems. *The Bell system technical journal*, 28(4):656–715, 1949.

[17] Apple Inc. Answers to your questions about Apple and security. https://www.apple.com/customer-letter/answers/, 2016. Accessed 2020-09-22.

[18] Apple Inc. A Message to Our Customers. https://www.apple.com/customer-letter/, 2 2016.

[19] James Comey. FBI Director Comments on San Bernardino Matter. https://www.fbi.gov/news/pressrel/press-releases/fbi-director-comments-on-san-bernardino-matter, 2 2016.

[20] Encryption Working Group. Moving the Encryption Policy Conversation Forward. Technical report, Carnegie Endowment for International Peace, 9 2019.

[21] Logan Koepke, Emma Weil, Urmila Janardan, Tinuola Dada, and Harlan Yu. Mass Extraction. https://www.upturn.org/reports/2020/mass-extraction/, 10 2020. Accessed 2020-10-25.

[22] Privacy International. A technical look at Phone Extraction. https://privacyinternational.org/sites/default/files/2019-10/A%20technical%20look%20at%20Phone%20Extraction%20FINAL.pdf, 10 2019. Accessed 2020-09-22.

[23] Joseph Cox. We Built a Database of Over 500 iPhones Cops Have Tried to Unlock. https://www.vice.com/en_us/article/4ag5yj/unlock-apple-iphone-database-for-police, 3 2020. Accessed 2020-09-22.

[24] Steven M Bellovin, Matt Blaze, Sandy Clark, and Susan Landau. Going bright: Wiretapping without weakening communications infrastructure. *IEEE Security & Privacy*, 11(1):62–72, 2012.

[25] Harold Abelson, Ross Anderson, Steven M Bellovin, Josh Benaloh, Matt Blaze, Whitfield Diffie, John Gilmore, Matthew Green, Susan Landau, Peter G Neumann, et al. Keys under doormats: mandating insecurity by requiring government access to all data and communications. *Journal of Cybersecurity*, 1(1):69–79, 2015.

[26] Stefan Savage. Lawful device access without mass surveillance risk: A technical design discussion. In *ACM CCS '18*, 2018.

[27] Raymond Edward Ozzie. Providing low risk exceptional access, December 10 2019. US Patent 10,505,734.

[28] Charles Wright. Crypto Crumple Zones: Protecting Encryption in a Time of Political Uncertainty. In *Enigma '18*. USENIX, 2018.

[29] Matthew Green. A few thoughts on Ray Ozzie's "Clear" proposal. https://blog.cryptographyengineering.com/2018/04/26/a-few-thoughts-on-ray-ozzies-clear-proposal/, 4 2018. Accessed May 6, 2021.

[30] Matthew Green, Gabriel Kaptchuk, and Gijs Van Laer. Abuse resistant law enforcement access systems. Cryptology ePrint Archive, Report 2021/321, 2021. https://eprint.iacr.org/2021/321.

[31] Joseph Cox and Izzie Ramirez. iPhone Warrant Database 2019. https://docs.google.com/spreadsheets/d/1Xmh1QEXYJmVPFlqAdEIVGemvbkoZmk_WyAPGC4eY-eE/edit#gid=0, 3 2020.

[32] Thomas Brewster. Apple Helps FBI Track Down George Floyd Protester Accused Of Firebombing Cop Cars. https://www.forbes.com/sites/thomasbrewster/2020/09/16/apple-helps-fbi-track-down-george-floyd-protester-accused-of-firebombing-cop-cars/, 9 2020. Accessed 2020-09-21.

[33] NIST. Mobile Device Forensic Tool Specification. https://www.nist.gov/system/files/documents/2019/07/11/mobile_device_forensic_tool_test_spec_v_3.0.pdf, 5 2019. Accessed 2020-08-04.

[34] DHS. Test Results for Mobile Device Acquisition. https://www.dhs.gov/publication/st-mobile-device-acquisition, 10 2019. Accessed 2020-08-04.

[35] S.3398 - EARN IT Act of 2020. https://www.congress.gov/bill/116th-congress/senate-bill/3398, 3 2020. Accessed 2020-09-22.

[36] Patrick Siewert. Apple iPhone Forensics: Significant Locations. https://www.forensicfocus.com/articles/apple-iphone-forensics-significant-locations/, 5 2018. Accessed 2020-09-22.

[37] Apple Inc. Apple Platform Security. https://github.com/maxzinkus/PhoneEncryptionDocumentArchive, 2019–2020. Archived.

[38] Android Open Source Project. Full-Disk Encryption. https://source.android.com/security/encryption/full-disk, 9 2020. Accessed 2020-09-09.

[39] Android Open Source Project. File-Based Encryption. https://source.android.com/security/encryption/file-based, 9 2020. Accessed 2020-09-09.

[40] Troy Kensinger. Google and Android have your back by protecting your backups. https://security.googleblog.com/2018/10/google-and-android-have-your-back-by.html, 10 2018. Accessed 2020-09-20.

[41] Apple Inc. Transparency Report. https://www.apple.com/legal/transparency/, 9 2020. Accessed 2020-09-21.

[42] Google LLC. Global requests for user information. https://transparencyreport.google.com/user-data/overview, 2019. Accessed 2020-09-25.

[43] Joseph Menn. Exclusive: Apple dropped plan for encrypting backups after FBI complained - sources. *Reuters*, 1 2020. Accessed 2020-09-13.

[44] Purism. https://puri.sm/, 2021. Accessed 05-24-2021.

[45] Oded Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[46] Apple Inc. Legal Process Guidelines. https://www.apple.com/legal/privacy/law-enforcement-guidelines-us.pdf, 12

2018. Accessed 2020-09-21.
[47] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In *TCC '07*, pages 137–156. Springer, 2007.
[48] Apple Inc. iCloud Security Overview. https://support.apple.com/en-us/HT202303, 7 2020. Accessed 2020-07-28.
[49] Xiaowen Xin. Titan M makes Pixel 3 our most secure phone yet. https://www.blog.google/products/pixel/titan-m-makes-pixel-3-our-most-secure-phone-yet/, 10 2018. Accessed 2020-09-09.
[50] Joshua Lund. Technology Preview for secure value recovery. https://signal.org/blog/secure-value-recovery/, 12 2019.
[51] Awanthika R Senarath and Nalin Asanka Gamagedara Arachchilage. Understanding user privacy expectations: A software developer's perspective. *Telematics and Informatics*, 35(7):1845–1862, 2018.
[52] Majid Hatamian, Jetzabel Serna, and Kai Rannenberg. Revealing the unrevealed: Mining smartphone users privacy perception on app markets. *Computers & Security*, 83: 332–353, 2019.
[53] Paul Van Schaik, Jurjen Jansen, Joseph Onibokun, Jean Camp, and Petko Kusev. Security and privacy in online social networking: Risk perceptions and precautionary behaviour. *Computers in Human Behavior*, 78:283–297, 2018.
[54] Josephine Lau, Benjamin Zimmerman, and Florian Schaub. Alexa, are you listening? privacy perceptions, concerns and privacy-seeking behaviors with smart speakers. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW): 1–31, 2018.
[55] Dirk Van Bruggen. *Studying the impact of security awareness efforts on user behavior*. PhD thesis, University of Notre Dame, 2014.
[56] Elissa M Redmiles, Noel Warford, Amritha Jayanti, Aravind Koneru, Sean Kross, Miraida Morales, Rock Stevens, and Michelle L Mazurek. A comprehensive quality evaluation of security and privacy advice on the web. In *USENIX Security '20*, pages 89–108, 2020.
[57] Mary Ellen Zurko and Richard T Simon. User-centered security. In *Proceedings of the 1996 workshop on New security paradigms*, pages 27–33, 1996.
[58] Anne Adams and Martina Angela Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):40–46, 1999.
[59] Alma Whitten and J Doug Tygar. Why johnny can't encrypt: A usability evaluation of pgp 5.0. In *USENIX Security '99*, 1999.
[60] A. Gibson et al. NSA targets the privacy-conscious. https://daserste.ndr.de/panorama/aktuell/NSA-targets-the-privacy-conscious,nsa230.html, 3 2014.
[61] Aya Fukami, Saugata Ghose, Yixin Luo, Yu Cai, and Onur Mutlu. Improving the reliability of chip-off forensic analysis of nand flash memory devices. *Digital Investigation*, 20: S1–S11, 2017.
[62] Apple Inc. Touch ID, Face ID, passcodes, and passwords. https://support.apple.com/guide/security/touch-id-face-id-passcodes-and-passwords-sec9479035f1/web, 2020. Accessed 2020-11-22.
[63] Lorenzo Franceschi-Bicchierai and Joseph Cox. Here Are Detailed Photos of iPhone Unlocking Tech GrayKey. https://www.vice.com/en_us/article/v7gkpx/graykey-grayshift-photos-iphone-unlocking-tech, 9 2020. Accessed 2020-09-20.
[64] Thomas Reed. GrayKey iPhone unlocker poses serious security concerns. *MalwareBytes SecurityWorld*, 3 2018. Accessed 2020-09-19.
[65] Robert Palazzo. FCC ID 2AV7EGK01. https://fccid.io/2AV7EGK01, 7 2020. Published by the FCC, accessed via unofficial viewer. Images archived.
[66] Shuzhe Yang and Gökhan Bal. Balancing security and usability of local security mechanisms for mobile devices. In Dimitris Gritzalis, Steven Furnell, and Marianthi Theoharidou, editors, *Information Security and Privacy Research*, pages 327–338, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-30436-1.
[67] Matthew Green and Matthew Smith. Developers are not the enemy!: The need for usable security apis. *IEEE Security & Privacy*, 14(5):40–46, 2016.
[68] Apple Inc. Apple Security Updates. https://support.apple.com/en-us/HT201222, 2003–2020. Accessed 2020-06 through 2020-07.
[69] Hui Lu, Xiaohan Helu, Chengjie Jin, Yanbin Sun, Man Zhang, and Zhihong Tian. Salaxy: Enabling usb debugging mode automatically to control android devices. *IEEE Access*, 7:178321–178330, 2019.
[70] Tielei Wang, Hao Xu, and Xiaobo Chen. Pangu 9 Internals. https://papers.put.as/papers/ios/2016/us-16-Pangu9-Internals.pdf, 8 2016. Accessed 2020-08-11.
[71] a1exdandy. Technical analysis of the checkm8 exploit. https://habr.com/en/company/dsec/blog/472762/, 10 2019.
[72] Roee Hay and Noam Hadad. Exploiting Qualcomm EDL Programmers (1): Gaining Access & PBL Internals. https://alephsecurity.com/2018/01/22/qualcomm-edl-1/, 1 2018. Accessed 2020-09-25.
[73] Nitay Artenstein. Broadpwn. *Black Hat USA*, 2017.
[74] Fenghao Xu, Wenrui Diao, Zhou Li, Jiongyi Chen, and Kehuan Zhang. Badbluetooth: Breaking android security mechanisms via malicious bluetooth peripherals. In *NDSS '19*, 2019.
[75] Timothy Vidas, Daniel Votipka, and Nicolas Christin. All your droid are belong to us: A survey of current android attacks. In *Woot*, pages 81–90, 2011.
[76] Danny Dolev and Andrew Yao. On the security of public key protocols. *IEEE Transactions on information theory*, 29 (2):198–208, 1983.
[77] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *Annual international cryptology conference*, pages 232–249. Springer, 1993.
[78] Cellebrite. What Happens When You Press that Button? https://smarterforensics.com/wp-content/uploads/2014/06/Explaining-Cellebrite-UFED-Data-Extraction-Processes-final.pdf, 6 2014. Accessed 2020-09-26.
[79] Philipp Markert, Daniel V Bailey, Maximilian Golla, Markus Dürmuth, and Adam J Aviv. This pin can be easily guessed. *arXiv preprint arXiv:2003.04868*, 2020.
[80] Sergei Skorobogatov. The bumpy road towards iphone 5c nand mirroring. *arXiv preprint arXiv:1609.04327*, 2016.
[81] Sarah Scheffler and Mayank Varia. Protecting cryptography against compelled self-incrimination. *Usenix Security 2021*, 2021.

[82] Apple Inc. What does iCloud back up? https://support.apple.com/en-us/HT207428, 9 2020. Accessed 2020-09-09.
[83] Google LLC. Google Mobile Services. https://www.android.com/gms/, 2020.
[84] Alex Hernandez. Man steals over 600K iCloud photos searching for nudes. https://techaeris.com/2021/09/11/man-steals-over-600k-icloud-photos-searching-for-nudes/, 9 2021.
[85] Russell Brandom. Police are filing warrants for Android's vast store of location data. https://www.theverge.com/2016/6/1/11824118/google-android-location-data-police-warrants, 6 2016. Accessed 2020-09-25.
[86] Apple Inc. Learn more about iCloud in China mainland. https://support.apple.com/en-us/HT208351, 5 2020. Accessed 2020-12-03.
[87] Apple Inc. Privacy. https://www.apple.com/privacy/, 9 2020. Accessed 2020-09-25.
[88] Lewis Leong. Chinese developers release untethered iOS 7.1.X jailbreak to much controversy. https://en.softonic.com/articles/pangu-ios-7-1-x-jailbreak, 6 2014. Accessed 2020-07-29.
[89] unc0ver jailbreak. https://unc0ver.dev/, 2 2021. Accessed 2021-02-27.
[90] Milan Broz. DMCrypt. https://gitlab.com/cryptsetup/cryptsetup/-/wikis/DMCrypt, 9 2020. Accessed 2020-12-02. dm-crypt documentation.
[91] Android Open Source Project. Rollback Resistance. https://source.android.com/security/keystore/implementer-ref#rollback_resistance, 9 2020. Accessed 2021-02-28.
[92] Li Yang, Teng Wei, Fengwei Zhang, and Jianfeng Ma. Sadus: Secure data deletion in user space for mobile devices. *Computers & Security*, 77:612 – 626, 2018. ISSN 0167-4048. https://doi.org/10.1016/j.cose.2018.05.013.
[93] Nirvan Tyagi, Muhammad Haris Mughees, Thomas Ristenpart, and Ian Miers. Burnbox: Self-revocable encryption in a world of compelled access. In *USENIX Security '18*, pages 445–461, 2018.
[94] Shijie Jia, Luning Xia, Bo Chen, and Peng Liu. Deftl: Implementing plausibly deniable encryption in flash translation layer. In *ACM CCS '17*, pages 2217–2229, 2017.
[95] Bing Chang, Yao Cheng, Bo Chen, Fengwei Zhang, Wen-Tao Zhu, Yingjiu Li, and Zhan Wang. User-friendly deniable storage for mobile devices. *computers & security*, 72: 163–174, 2018.
[96] Chen Chen, Anrin Chakraborti, and Radu Sion. Infuse: Invisible plausibly-deniable file system for nand flash. *Proceedings on Privacy Enhancing Technologies*, 2020(4): 239–254, 2020.
[97] Android Open Source Project. Fingerprint HIDL. https://source.android.com/security/authentication/fingerprint-hal, 9 2020. Accessed 2020-09-09.
[98] Apple Inc. FaceID Security. https://github.com/maxzinkus/PhoneEncryptionDocumentArchive, 11 2017. Archived.
[99] Android Open Source Project. Face Authentication HIDL. https://source.android.com/security/biometric/face-authentication, 9 2020.
[100] Android Open Source Project. Authentication. https://source.android.com/security/authentication, 9 2020. Accessed 2020-09-09.
[101] Android Open Source Project. Gatekeeper. https://source.android.com/security/authentication/gatekeeper, 9 2020.
[102] ARM Holdings. Arm TrustZone Technology. https://developer.arm.com/ip-products/security-ip/trustzone, 9 2020. Accessed 2020-09-09.
[103] Android Open Source Project. Trusty TEE. https://source.android.com/security/trusty, 9 2020. Accessed 2020-09-09.
[104] Liang Kai. Guard your data with the Qualcomm Snapdragon Mobile Platform. https://github.com/maxzinkus/PhoneEncryptionDocumentArchive, 4 2019. Accessed 2020-09-09. Archived.
[105] Google LLC. Android keystore system. https://developer.android.com/training/articles/keystore, 10 2020. Accessed 2021-02-28.
[106] Google LLC. Behavior changes: all apps. https://developer.android.com/about/versions/pie/android-9.0-changes-all, 12 2019. Documentation for Android 9, accessed 2020-09-09.
[107] Tarjei Mandt, Mathew Solnik, and David Wang. Demystifying the secure enclave processor. *Black Hat Las Vegas*, 2016.
[108] Elcomsoft. iOS Forensic Toolkit 6.50: jailbreak-free extraction without an Apple Developer Account. https://www.elcomsoft.com/news/762.html, 9 2020. Accessed 2020-09-22.
[109] Cellebrite. Cellebrite Advanced Services. https://cf-media.cellebrite.com/wp-content/uploads/2020/09/SolutionOverview_CAS_2020.pdf, 9 2020.
[110] Clemens Fruhwirth. *New methods in hard disk encryption*. na, 2005.
[111] CPSC. IEEE Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices. *IEEE Std. 1619-2018*, 1 2019.
[112] Moses Liskov, Ronald L Rivest, and David Wagner. Tweakable block ciphers. In *Annual International Cryptology Conference*, pages 31–46. Springer, 2002.
[113] Luther Martin. Xts: A mode of aes for encrypting hard disks. *IEEE Security & Privacy*, 8(3):68–69, 2010.
[114] Carlo Meijer and Bernard Van Gastel. Self-encrypting deception: weaknesses in the encryption of solid state drives. In *IEEE S&P '19*. IEEE, 2019.
[115] Eoghan Casey and Gerasimos J Stellatos. The impact of full disk encryption on digital forensics. *ACM SIGOPS Operating Systems Review*, 42(3):93–98, 2008.
[116] Oleg Afonin. This $39 Device Can Defeat iOS USB Restricted Mode. https://blog.elcomsoft.com/2018/07/this-9-device-can-defeat-ios-usb-restricted-mode/, 7 2018. Accessed 2020-09-23.
[117] Vladimir Katalov. Working Around the iPhone USB Restricted Mode. https://blog.elcomsoft.com/2020/05/iphone-usb-restricted-mode-workaround/, 5 2020. Accessed 2020-11-07.
[118] Kanad Basu, Deepraj Soni, Mohammed Nabeel, and Ramesh Karri. Nist post-quantum cryptography-a hardware evaluation study. *IACR Cryptol. ePrint Arch.*, 2019: 47, 2019.
[119] Paul Crowley and Eric Biggers. Adiantum: length-preserving encryption for entry-level processors. *IACR Transactions on Symmetric Cryptology*, pages 39–61, 2018.

[120] Levent Demir, Mathieu Thiery, Vincent Roca, Jean-Michel Tenkes, and Jean-Louis Roch. Optimizing dm-crypt for xts-aes: Getting the best of atmel cryptographic co-processors (long version). In *SECRYPT '20*, 2020.

[121] Oleg Afonin. Smartphone Encryption: Why Only 10 Per Cent of Android Smartphones Are Encrypted. https://blog.elcomsoft.com/2016/03/smartphone-encryption-why-only-10-per-cent-of-android-smartphones-are-encrypted/, 3 2016.

[122] Matt Blaze. A cryptographic file system for unix. In *ACM CCS '93*, 1993.

[123] Michael Austin Halcrow. ecryptfs: An enterprise-class encrypted filesystem for linux. In *Proceedings of the 2005 Linux Symposium*, volume 1, pages 201–218, 2005.

[124] Timothy M Peters, Mark A Gondree, and Zachary NJ Peterson. Defy: A deniable, encrypted file system for log-structured storage. In *NDSS '15*, 2 2015.

[125] Aviad Zuck, Yue Li, Jehoshua Bruck, Donald E. Porter, and Dan Tsafrir. Stash in a flash. In *FAST '18*. USENIX, 2018.

[126] Joel Reardon, David Basin, and Srdjan Capkun. Sok: Secure data deletion. In *IEEE S&P '13*. IEEE, 2013.

[127] Ross Anderson, Roger Needham, and Adi Shamir. The steganographic file system. In *International Workshop on Information Hiding*, pages 73–82. Springer, 1998.

[128] Johannes Götzfried, Moritz Eckert, Sebastian Schinzel, and Tilo Müller. Cache attacks on intel sgx. In *Proceedings of the 10th European Workshop on Systems Security*, pages 1–6, 2017.

[129] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution. In *USENIX Security '18*, pages 991–1008, 2018.

[130] J Taylor. Security for the next generation of safe real-time systems. In *Proceedings of Embedded World Conference*, 2016.

[131] Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, et al. seL4: Formal verification of an OS kernel. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 207–220, 2009.

[132] Apple Inc. Apple Pay security and privacy overview. https://support.apple.com/en-us/HT203027, 7 2020. Accessed 2020-07-30.

[133] Dayeol Lee, David Kohlbrenner, Shweta Shinde, Krste Asanović, and Dawn Song. Keystone: An open framework for architecting trusted execution environments. In *EuroSys '20*. ACM, 2020.

[134] Krste Asanović and David A Patterson. Instruction sets should be free: The case for risc-v. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146*, 2014.

[135] Michael Henson and Stephen Taylor. Beyond full disk encryption: protection on security-enhanced commodity processors. In *International Conference on Applied Cryptography and Network Security*, pages 307–321. Springer, 2013.

[136] P. A. H. Peterson. Cryptkeeper: Improving security with encrypted ram. In *IEEE HST '10*, 2010.

[137] Alexander Würstlein, Michael Gernoth, Johannes Götzfried, and Tilo Müller. Exzess: Hardware-based ram encryption against physical memory disclosure. In *International Conference on Architecture of Computing Systems*, pages 60–71. Springer, 2016.

[138] Android Open Source Project. Verified Boot. https://source.android.com/security/verifiedboot, 9 2020. Accessed 2020-09-09.

[139] Debnath Bhattacharyya, Rahul Ranjan, Farkhod Alisherov, Minkyu Choi, et al. Biometric authentication: A review. *International Journal of u-and e-Service, Science and Technology*, 2(3):13–28, 2009.

[140] Ivan Cherapau, Ildar Muslukhov, Nalin Asanka, and Konstantin Beznosov. On the Impact of TouchID on iPhone Passcodes. In *SOUPS '15)*, 2015.

[141] Silvio Barra, Maria De Marsico, Michele Nappi, Fabio Narducci, and Daniel Riccio. A hand-based biometric system in visible light for mobile environments. *Information Sciences*, 479:472–485, 2019.

[142] Adrian-Stefan Ungureanu, Shejin Thavalengal, Timothée E Cognard, Claudia Costache, and Peter Corcoran. Unconstrained palmprint as a smartphone biometric. *IEEE Transactions on Consumer Electronics*, 63(3):334–342, 2017.

[143] Ajita Rattani and Reza Derakhshani. Online co-training in mobile ocular biometric recognition. In *IEEE HST '17)*. IEEE, 2017.

[144] Chiara Galdi and Jean-Luc Dugelay. Fire: fast iris recognition on mobile phones by combining colour and texture features. *Pattern Recognition Letters*, 91:44–51, 2017.

[145] Andrea F Abate, Silvio Barra, Luigi Gallo, and Fabio Narducci. Kurtosis and skewness at pixel level as input for som networks to iris recognition on mobile devices. *Pattern Recognition Letters*, 91:37–43, 2017.

[146] Karan Ahuja, Rahul Islam, Ferdous A Barbhuiya, and Kuntal Dey. Convolutional neural networks for ocular smartphone-based biometrics. *Pattern Recognition Letters*, 91:17–26, 2017.

[147] Fernando Alonso-Fernandez, Kiran B Raja, Christoph Busch, and Josef Bigun. Log-likelihood score level fusion for improved cross-sensor smartphone periocular recognition. In *EUSIPCO '17*. IEEE, 2017.

[148] Robin Tan and Marek Perkowski. Toward improving electrocardiogram (ecg) biometric verification using mobile sensors: A two-stage classifier approach. *Sensors*, 17(2): 410, 2017.

[149] Andrew Crocker. Victory: Pennsylvania Supreme Court Rules Police Can't Force You to Tell Them Your Password. https://www.eff.org/deeplinks/2019/11/victory-pennsylvania-supreme-court-rules-police-cant-force-you-tell-them-your, 11 2019. Accessed 2020-12-03.

[150] Apple Inc. iOS Security. https://github.com/maxzinkus/PhoneEncryptionDocumentArchive, 2012–2019. iOS Security Guides. Archived.

[151] Adam J Aviv, Devon Budzitowski, and Ravi Kuber. Is bigger better? comparing user-generated passwords on 3x3 vs. 4x4 grid sizes for android's pattern unlock. In *Proceedings of the 31st Annual Computer Security Applications Conference*, pages 301–310, 2015.

[152] Russell Brandom. A new hack could let thieves bypass the iPhone's lockscreen. https://www.theverge.com/2015/3/30/8311835/iphone-lockscreen-hack-theft-find-my-iphone, 3 2015. Accessed 2020-09-09.

[153] Adam J Aviv, Katherine L Gibson, Evan Mossop, Matt Blaze, and Jonathan M Smith. Smudge attacks on smartphone touch screens. *Woot*, 10:1–7, 2010.

[154] Man Zhou, Qian Wang, Jingxiao Yang, Qi Li, Feng Xiao, Zhibo Wang, and Xiaofeng Chen. Patternlistener: Cracking android pattern lock using acoustic signals. In *ACM CCS '18*, pages 1775–1787, 2018.

[155] Sebastian Uellenbeck, Markus Dürmuth, Christopher Wolf, and Thorsten Holz. Quantifying the security of graphical passwords: the case of android unlock patterns. In *ACM CCS '13*, pages 161–172, 2013.

[156] Burt Kaliski. Pkcs# 5: Password-based cryptography specification version 2.0. Technical report, RFC 2898, september, 2000.

[157] Oleg Afonin. Protecting Your Data and Apple Account If They Know Your iPhone Passcode. https://blog.elcomsoft.com/2018/06/protecting-your-data-and-apple-account-if-they-know-your-iphone-passcode/, 6 2018. Accessed 2020-09-22.

[158] Adi Shamir. How to share a secret. *Commun. ACM*, 22 (11), November 1979.

[159] Stéphanie Delaune, Steve Kremer, and Graham Steel. Formal analysis of pkcs# 11. In *2008 21st IEEE Computer Security Foundations Symposium*, pages 331–344. IEEE, 2008.

[160] Yubico. YubiKey 5 NFC. https://www.yubico.com/se/product/yubikey-5-nfc/, 9 2021.

[161] Kasper Green Larsen and Jesper Buus Nielsen. Yes, there is an oblivious ram lower bound! In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO '18*, 2018.

[162] Google LLC. Security. https://support.google.com/android/answer/9075927, 2020. Accessed 2020-09-18.

[163] Ahmed Mahfouz, Tarek M. Mahmoud, and Ahmed Sharaf Eldin. A survey on behavioral biometric authentication on smartphones. *Journal of Information Security and Applications*, 37, 2017.

[164] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *STOC '13*, 2013.

[165] Apple Inc. Use Handoff to continue tasks on your other devices. https://support.apple.com/en-us/HT209455, 2021. Accessed 2021-05-31.

[166] Uday Savagaonkar, Nelly Porter, Nadim Taha, Benjamin Serebrin, and Neal Mueller. Titan in depth: Security in plaintext. https://cloud.google.com/blog/products/gcp/titan-in-depth-security-in-plaintext, 8 2017. Accessed 2020-09-25.

[167] Stanislaw Jarecki, Hugo Krawczyk, and Jiayu Xu. Opaque: an asymmetric pake protocol secure against pre-computation attacks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 456–486. Springer, 2018.

[168] Thomas D et al Wu. The secure remote password protocol. In *NDSS*, volume 98, pages 97–111. Citeseer, 1998.

[169] David M'Raihi, Salah Machani, Mingliang Pei, and Johan Rydell. Totp: Time-based one-time password algorithm. *Internet Request for Comments*, 2011.

[170] David M'Raihi, Mihir Bellare, Frank Hoornaert, David Naccache, and Ohad Ranen. Hotp: An hmac-based one-time password algorithm. *The Internet Society, Network Working Group. RFC4226*, 2005.

[171] Apple Inc. Two-factor authentication for Apple ID. https://support.apple.com/en-us/HT204915, 7 2020. Accessed 2020-07-28.

[172] Juan Benet and Nicola Greco. Filecoin: A decentralized storage network. *Protoc. Labs*, pages 1–36, 2018.

[173] Juan Benet. IPFS: Content addressed, versioned, P2P file system. *arXiv preprint arXiv:1407.3561*, 2014.

[174] Adam Eijdenberg, Ben Laurie, and Al Cutter. Verifiable Data Structures. https://continusec.com/static/VerifiableDataStructures.pdf, 11 2015.

[175] Joan Daemen and Vincent Rijmen. The block cipher rijndael. In *International Conference on Smart Card Research and Advanced Applications*, pages 277–284. Springer, 1998.

[176] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC '09*, 2009.

[177] Sean W Smith and Vernon Austel. Trusting trusted hardware: Towards a formal model for programmable secure coprocessors. In *USENIX Workshop on Electronic Commerce*, 1998.

[178] Cynthia E Irvine and Karl Levitt. Trusted hardware: Can it be trustworthy? In *2007 44th ACM/IEEE Design Automation Conference*, pages 1–4. IEEE, 2007.

[179] Ivan Krstic. Behind the Scenes with iOS Security. https://www.blackhat.com/docs/us-16/materials/us-16-Krstic.pdf, 8 2016. Accessed 2020-09-07.

[180] Udi Manber. A simple scheme to make passwords based on one-way functions much harder to crack. *Computers & Security*, 15(2):171–176, 1996.

[181] Martın Abadi, T Mark A Lomas, and Roger Needham. Strengthening passwords. Technical report, Citeseer, 1997.

[182] John Kelsey, Bruce Schneier, Chris Hall, and David Wagner. Secure applications of low-entropy keys. In *International Workshop on Information Security*, pages 121–134. Springer, 1997.

[183] J Alex Halderman, Brent Waters, and Edward W Felten. A convenient method for securely managing passwords. In *WWW '05*, pages 471–479, 2005.

[184] Ian McQuoid, Mike Rosulek, and Lawrence Roy. Minimal symmetric pake and 1-out-of-n ot from programmable-once public functions. In *ACM CCS '20*, pages 425–442, 2020.

[185] Jolyon Clulow. On the security of pkcs# 11. In *CHES '03*. Springer, 2003.

[186] Matteo Bortolozzo, Matteo Centenaro, Riccardo Focardi, and Graham Steel. Attacking and fixing pkcs# 11 security tokens. In *ACM CCS '10*, 2010.

[187] Google LLC. Stronger security for your Google Account. https://www.google.com/landing/2step/, 2021. Accessed 2021-02-28.

[188] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[189] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *USENIX ATC '14*, 2014.

[190] Gabriel Kaptchuk. *New Applications of Public Ledgers*. PhD thesis, The Johns Hopkins University, 2020.
[191] Ben Laurie. Certificate transparency. *Commun. ACM*, 57(10):40–46, September 2014. ISSN 0001-0782. URL https://doi.org/10.1145/2659897.
[192] Marcela S. Melara, Aaron Blankstein, Joseph Bonneau, Edward W. Felten, and Michael J. Freedman. CONIKS: Bringing key transparency to end users. In *USENIX Security '15*. USENIX, 2015.
[193] Google. Key Transparency. https://github.com/google/keytransparency/, 11 2020.
[194] Russ Cox and Filippo Valsorda. Proposal: Secure the Public Go Module Ecosystem. https://go.googlesource.com/proposal/+/master/design/25530-sumdb.md, 4 2019.
[195] Andy Greenberg. The Clever Cryptography Behind Apple's 'Find My' Feature. https://www.wired.com/story/apple-find-my-cryptography-bluetooth/, 6 2019. Accessed 2020-07-19.
[196] Alexander Heinrich, Milan Stute, Tim Kornhuber, and Matthias Hollick. Who can find my devices? security and privacy of apple's crowd-sourced bluetooth location tracking system. *arXiv preprint arXiv:2103.02282*, 2021.
[197] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, Manubot, 2019.
[198] Tom Ritter. Private by Design: How we built Firefox Sync. https://hacks.mozilla.org/2018/11/firefox-sync-privacy/, 11 2018. Accessed 2021-05-30.
[199] Google LLC. Back up user data with Auto Backup. https://developer.android.com/guide/topics/data/autobackup, 1 2020. Accessed 2020-09-25.
[200] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC '11*, pages 253–273. Springer, 2011.