Beyond OS Trust Stores: TLS Trust in Russia's Android Ecosystem

William J. Tolley
Hampden-Sydney College
Breakpointing Bad
william@breakpointingbad.com

Everett Morse
Hampden-Sydney College
morsee29@hsc.edu

Jedidiah R. Crandall
Arizona State University
Breakpointing Bad
jed@breakpointingbad.com

Abstract

This paper presents a measurement study of trust-anchor distribution in the Russian Android ecosystem. We analyze applications distributed through RuStore, a domestic marketplace promoted in Russia after the 2022 sanctions, in order to examine how APKs bundle certificate material and related trust-configuration artifacts independently of the operating system trust store. Our analysis combines ecosystem-wide measurement with a weighted top-application view to study how application packaging and software supply chains can expand the effective trust boundary of TLS. We discuss the implications for mobile trust, software supply chains, and the assumption that TLS remains a sufficient fallback against hostile networks.

Keywords

TLS, Android Security, Trust Management, Censorship Infrastructure, Russian Mobile Ecosystem

1 Introduction: TLS Trust Anchors in the Russian Android Ecosystem

Transport Layer Security (TLS) is widely treated as the mechanism that preserves confidentiality and authenticity against on-path adversaries. In discussions of censorship circumvention, mobile security, and Internet freedom technologies, it is often assumed that once application traffic is protected by TLS, network operators can observe metadata but cannot meaningfully terminate or impersonate encrypted connections without triggering certificate warnings [10, 38].

That assumption depends on more than the TLS protocol alone. TLS authentication is only as strong as the certificate authority (CA) relationships accepted by the client. When a client accepts a certificate chain anchored in a trusted root, it implicitly delegates authority over endpoint authentication to the entity controlling that root [1, 17]. If that trust relationship fails, TLS may remain cryptographically valid while providing little protection against an adversary capable of issuing certificates trusted by the client.

This problem becomes especially important in environments where state actors combine network control with influence over trust infrastructure. In such settings, the power to issue certificates from a trusted authority can enable transparent TLS interception if those certificates are accepted by devices or applications [1, 3]. Russia provides a particularly important case study. Following the 2022 sanctions imposed after the full-scale invasion of Ukraine, Russia accelerated the development of domestic digital infrastructure, including a national certificate authority commonly known as the *Russian Trusted Root CA* and a domestic Android distribution channel, RuStore [9]. This raises a broader question with direct relevance to Internet freedom: to what extent has state-associated trust moved beyond the operating system trust store and into the application layer itself?

Prior work has documented weaknesses in TLS trust, including certificate validation failures in deployed software, TLS interception by intermediaries, and the security consequences of custom trust handling in deployed applications [4, 11, 36]. Other work has examined the security and privacy properties of mobile ecosystems in politically distinct national environments, including recent analyses of national app ecosystems [15]. Yet comparatively little attention has been paid to how state-associated trust anchors propagate through mobile application packages themselves, and how application packaging and software dependencies may silently expand the effective trust boundary beyond the operating system.

On Android, applications can alter trust relationships by bundling certificate material inside the APK, defining custom `network_security_config` policies, loading trust stores through application code, or relying on third-party SDKs that introduce their own trust artifacts [27]. As a result, the effective trust configuration experienced by the user may differ substantially from the operating system default, even when the user has not explicitly installed any additional certificate authorities.

In this paper, we study the structural distribution of trust anchors in the Russian Android ecosystem. We conduct a large-scale static analysis of applications distributed through RuStore and complement that ecosystem-wide view with a focused analysis of the most widely used Russian applications weighted by download share. Our goal is to examine how application packaging and software supply chains can expand the effective trust boundary of TLS in a politically distinct mobile ecosystem, including whether repeated certificate artifacts across unrelated applications are consistent with shared dependency or supply-chain propagation [39].

The remainder of this paper proceeds as follows. Section 2 reviews the TLS trust model and prior incidents involving government-controlled certificate authorities. Section 3 describes the adversary model considered in this work. Section 4 presents our dataset construction and analysis pipeline. Section 5 reports empirical measurements of certificate distribution and trust configuration across the RuStore ecosystem and the top-app subset. Section 6 discusses the broader implications for mobile application security, software supply chains, and the limits of treating TLS as a sufficient fallback against state-level network adversaries.

This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license visit https://creativecommons.org/licenses/by/4.0/ or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.
Free and Open Communications on the Internet 2026(2), 12–20
© 2026 Copyright held by the owner/author(s).

Our contributions are:

- A large-scale measurement of certificate bundling and state-associated trust-anchor inclusion in applications distributed through the RuStore Android marketplace.
- Evidence that Russian state-associated trust anchors are concentrated in widely used apps when exposure is weighted by top-app download share.
- Evidence that repeated internal certificate file paths across unrelated applications are consistent with shared-SDK or software-supply-chain propagation rather than isolated manual inclusion.
- Evidence that bundled Russian trust anchors frequently co-occur with explicit trust-configuration artifacts, indicating that application trust in this ecosystem often extends beyond the default operating system trust store.

2 Background

This section reviews the TLS trust assumptions and application-level trust mechanisms that motivate our measurement of bundled certificate material in RuStore applications.

2.1 The TLS Trust Model

TLS provides confidentiality, integrity, and endpoint authentication for communication between a client and a server [30]. In the standard model, the server presents an X.509 certificate, and the client accepts that certificate only if it chains to a trusted root certificate authority (CA) [2]. The security of TLS therefore depends not only on the protocol itself but also on the certificate authorities the client is willing to trust.

Modern operating systems and browsers ship with curated trust stores containing many root CAs. Unless additional constraints such as certificate pinning or name constraints are imposed, any trusted CA may be able to authenticate arbitrary domains. In practice, TLS security depends on both protocol design and trust-store governance.

2.2 Certificate Authorities and Application Trust

On most consumer devices, trust stores are maintained by platform vendors such as Android, iOS, Windows, or macOS. In practice, however, applications do not always rely exclusively on the system trust store. Prior work has shown that deployed software frequently implements its own TLS handling or custom certificate validation behavior [5, 11, 25, 29].

Android applications can alter trust relationships in several ways. They may bundle certificate material directly inside the APK, define custom trust policies through `network_security_config`, load trust stores from application resources, or implement certificate validation logic in code [12, 27]. These mechanisms allow an application to extend or alter the set of certificate authorities it trusts beyond those provided by the operating system.

2.3 TLS Interception and Trusted Roots

TLS interception occurs when an intermediary terminates an encrypted connection and establishes a separate TLS session with the intended server. This can occur without certificate warnings if the intermediary presents a certificate chaining to a root trusted by the client or the application's own trust configuration [4]. Such mechanisms are commonly used in enterprise security products and can also be employed by network operators or state-controlled infrastructure when devices or applications trust certificates issued under their control [4, 18, 28, 31].

2.4 State Network Control and Government-Operated CAs

Many countries have deployed centralized network infrastructure designed to monitor, filter, or regulate internet traffic [7, 9]. In the Russian Federation, legislation adopted in the late 2010s authorized deployment of traffic management equipment commonly referred to as *Technical Means for Countering Threats* (TSPU), which supports centralized filtering and routing enforcement across major operators [8, 14, 26, 37].

Concerns about government-enabled TLS interception are not hypothetical. In 2019, users in Kazakhstan were instructed to install a government-issued root certificate enabling interception of encrypted traffic without browser warnings [28]. Major browser vendors later blocked the certificate even when installed locally [23, 24], illustrating both the feasibility of large-scale TLS interception and the importance of trust-store governance.

More recently, the Russian Federation introduced a domestic certificate authority known as the *Russian Trusted Root CA* [8, 9, 28]. This raises a distinct question for the Android ecosystem: not merely whether trust in such a CA appears at the platform level, but whether trust in that authority has propagated into application packages themselves.

2.5 Mobile Application Ecosystems in Politically Distinct Contexts

Prior work has identified security weaknesses in mobile networking stacks and WebView-based application architectures [6, 22, 32]. Other work has examined how app ecosystems interact with censorship, availability controls, and regional regulatory environments [13, 15, 19, 21, 34].

Most directly relevant to this paper, Konyukhov [20] showed that several widely used Russian Android applications embed the Russian Trusted Root CA and, in case-study testing, accept certificates issued by it during TLS validation. That result motivates the larger ecosystem question we study here: how widely state-associated trust anchors are distributed across Android applications in the RuStore ecosystem, and how application-level trust configuration may expand the effective TLS trust boundary beyond the operating system trust store.

3 Threat Model

We consider a network adversary whose capabilities are consistent with state-operated or nationally coordinated network infrastructure. The adversary's objective is to observe, infer, or manipulate user traffic protected by Transport Layer Security (TLS) without triggering certificate validation errors on the client device.

3.1 Adversary Capabilities

We assume an adversary positioned within national or regional network infrastructure, for example through traffic management equipment deployed at internet service providers, exchange points, or upstream operators [9, 26]. Under this model, the adversary can observe traffic between user devices and remote services, inject or modify packets in transit, and terminate or proxy connections.

In addition, the adversary may be able to issue TLS certificates from a certificate authority trusted by the client device or application [28]. This may occur if a state operates or influences a certificate authority trusted by client software, if interception certificates are installed locally, or if applications include additional certificate authorities within their own trust configuration [27]. When a client accepts certificates issued by such an authority, the adversary can authenticate TLS connections to arbitrary domains without triggering certificate validation errors.

3.2 Adversary Goals

The adversary seeks to use this position to achieve one or more of the following outcomes:

- identify which services or domains a user is accessing;
- inspect application traffic nominally protected by TLS; and
- selectively block, degrade, or disrupt connections to specific services.

Unlike traditional TLS attacks, this model does not rely on cryptographic weaknesses or certificate-validation bugs. Instead, it relies on the interaction between network control and trust relationships accepted by the client.

3.3 Adversary Limitations

We do not assume that the adversary compromises the user's device or operating system. In particular, we assume that the operating system behaves according to its documented trust policies, that TLS implementations correctly perform certificate validation, and that the cryptographic primitives used by TLS remain secure.

Under these assumptions, TLS sessions may remain cryptographically valid even when traffic is intercepted by an intermediary trusted by the client.

3.4 Scope of Measurement

Our empirical analysis focuses on the structural distribution of certificate authorities within Android application packages rather than on full runtime validation behavior. Specifically, we measure whether applications distributed through the RuStore ecosystem bundle certificate material that could expand the set of trust anchors available to the application's TLS validation logic independently of the operating system trust store.

This provides a scalable way to examine how application packages modify the effective TLS trust boundary. While static analysis cannot by itself determine whether a given certificate is exercised at runtime, it reveals which authorities are available to the application and therefore capable of influencing TLS authentication under the adversary model above.

4 Methodology

This section describes the datasets and analysis pipeline used to measure trust-anchor distribution in Android applications distributed through the RuStore ecosystem. Our goal is to determine how frequently applications include certificate material that can expand trust beyond the operating system trust store, whether that material includes certificates associated with the Russian national PKI, and whether such patterns are concentrated in the ecosystem's most widely used applications.

4.1 Study Scope

We study the Russian Android application ecosystem as a case study of TLS trust under adversarial PKI conditions. Russia provides a particularly relevant environment because it combines a domestic application distribution ecosystem, a state-controlled certificate authority introduced after the 2022 sanctions, and network infrastructure capable of large-scale traffic control [9, 26].

Rather than surveying multiple countries superficially, we examine one ecosystem in depth and ask whether applications distributed through Russian channels include trust anchors that expand the set of certificate authorities recognized by the application.

4.2 RuStore Corpus Construction

We built a corpus of Android applications distributed through RuStore, a Russian Android application marketplace promoted as a domestic alternative to Google Play following the 2022 sanctions [35]. Our collection pipeline consisted of four stages: package enumeration, payload download, APK extraction, and corpus normalization.

First, we enumerated package identifiers from RuStore using a Playwright-based crawler that visited public catalog and search-result pages, executed dynamically loaded content, scrolled result pages until no new application entries appeared, and extracted package identifiers from application URLs. We deduplicated package identifiers across all crawled pages before attempting downloads. This stage yielded 1,535 unique package identifiers.

Second, we attempted to download the corresponding application payloads from RuStore's backend API. Of the 1,535 enumerated packages, 1,426 downloads succeeded, while the remaining packages could not be retrieved because of missing download links, API-side failures, or persistent transport errors suggesting that the application was no longer available.

Third, we normalized the downloaded artifacts into a consistent APK corpus. Some RuStore payloads were direct APK files, while others were ZIP bundles containing embedded APKs. In addition to extracting APKs from files with standard extensions, we inspected remaining payloads by file signature and archive structure in order to recover valid APKs from APK-structured files that lacked standard extensions. Payloads that could not be parsed as APKs were excluded.

This normalization process initially identified 1,295 APKs from files with standard APK or ZIP structure. Signature-based inspection recovered additional valid APKs from files with nonstandard or missing extensions. The final normalized corpus contained 1,394 analyzable APKs.

4.3 Static APK Analysis Pipeline

Each APK in the normalized corpus was analyzed using a static pipeline designed to identify bundled certificates and related trust-configuration artifacts. For each application, we unpacked the APK and inspected its resources, assets, and certificate-bearing files. The analysis recorded certificate subjects, fingerprints, and file paths within the package. This process was automated using file-extension searches, certificate parsing, string matching over extracted resources and configuration files, and manual spot checks of representative outputs.

Across the 1,394 analyzed applications, this process produced an inventory of 7,445 certificate records, of which 4,608 contained SHA-256 fingerprint information. We use certificate record to refer to a certificate-like artifact observed at a particular path inside an APK. Some records lacked SHA-256 fingerprints because the artifact could not be parsed directly as a complete X.509 certificate, for example because it was contained in a keystore, was malformed or encoded, or was identified through path or configuration evidence rather than direct certificate parsing.

Our analysis focuses on the structural presence of certificate authorities within application packages rather than on full runtime TLS behavior. Specifically, we measure how frequently applications distributed through RuStore include certificate material that could expand or otherwise modify the set of trust anchors available to the application's TLS validation logic.

We therefore focus on four classes of artifacts:

(1) certificates corresponding to the Russian Trusted Root CA or subordinate Russian state certificates;
(2) Huawei Mobile Services (HMS) trust stores and related keystore files;
(3) other bundled certificate files indicating that an application includes additional certificate material within the APK; and
(4) explicit trust-configuration artifacts, including Android network security configuration files, custom trust-anchor definitions, and indicators of certificate pinning logic.

For Russian trust anchors, we combined subject-string matching with canonical fingerprint validation. Certificates whose subject fields contained identifiers such as "Russian Trusted Root CA" or "Russian Trusted Sub CA" were first identified via string matching. These candidates were then validated against known SHA-256 fingerprints observed in the corpus to ensure consistent identification across applications.

4.4 Classification Criteria

Using the extracted certificate inventory, we classified applications according to whether they bundled additional certificate material and whether that material included specific trust anchors or trust-configuration signals of interest.

An application was classified as **bundling any certificate** if the APK contained one or more certificate-bearing files such as `.cer`, `.crt`, `.pem`, `.der`, or BKS keystores.

An application was classified as **bundling the Russian CA** if it included a certificate whose subject or fingerprint matched the Russian Trusted Root CA or its subordinate CA.

An application was classified as **bundling Huawei trust stores** if it contained characteristic HMS keystore artifacts such as `hmsr ootcas.bks`, `hmsincas.bks`, or `grs_sp.bks`. These files are associated with Huawei SDK dependencies and indicate the presence of Huawei trust infrastructure within the application package [16].

An application was classified as having **custom trust configuration** if it included artifacts indicating trust behavior beyond the default operating system trust store, including explicit `network_security_config` policies, custom trust-anchor declarations, or code-level indicators associated with certificate pinning or custom certificate handling. Applications not meeting this criterion were treated as relying on default trust behavior for the purposes of the weighted top-app analysis.

Because bundled certificate material does not by itself demonstrate how an application performs TLS validation at runtime, we treat these categories as structural evidence of additional trust anchors or trust-modifying behavior included within the application package.

4.5 Top-Application Metadata and Weighted Exposure Analysis

The full RuStore corpus allows us to measure ecosystem-wide prevalence, but raw APK counts do not fully capture likely user exposure. We therefore conducted a second analysis focused on the most widely used Russian applications.

For this stage, we enriched the APK analysis output with application metadata collected from RuStore, including application name, category, and coarse download-popularity information. We then constructed a top-app subset consisting of the 100 most-downloaded applications according to RuStore's popularity metadata for which both trust-analysis results and popularity metadata were available.

Using this subset, we computed a weighted exposure view of trust-anchor distribution. Rather than counting each APK equally, this analysis weights findings by download share derived from the RuStore popularity metadata for each application. Specifically, we collected the download-count labels displayed on RuStore app pages, converted those counts into numeric estimates, and then normalized them across the top-app subset so that each application's weight reflects its share of total observed downloads in that subset. This allows us to distinguish long-tail prevalence from concentration in applications with the greatest apparent user reach.

We used the same trust classifications in the top-app analysis as in the full-corpus analysis. In particular, we examined the distribution of Russian CA bundling across the top-app subset as a function of whether an application was classified as having custom trust configuration under the definition above. Applications in the complementary category were treated as exhibiting default trust behavior.

This second analysis answers a different question from the full-corpus measurement. The corpus-wide results show how widespread trust-anchor expansion is across the RuStore ecosystem as a whole. The weighted top-app results show whether those patterns are concentrated in the applications most likely to shape real-world user risk.

4.6 Certificate Path Clustering

To distinguish between developer-specific configuration and broader software supply-chain effects, we analyzed the file paths under

which Russian certificates appeared within APKs. When a certificate appears in many applications under the same or closely related internal paths, this suggests propagation through shared SDKs, libraries, or build pipelines rather than independent manual inclusion by each developer [15].

We therefore grouped applications by certificate path and counted the number of unique APKs containing each path. This clustering analysis allowed us to identify recurrent paths such as `res/raw/rootca_ssl_rsa2022.cer`, `res/raw/russian_trusted_root_ca.cer`, and related variants.

4.7 Ecosystem Attribution

To understand how broadly Russian trust anchors are distributed across the mobile ecosystem, we grouped applications by package prefix in order to identify major vendor families whose applications include the Russian CA. This allows us to estimate whether the behavior is confined to isolated applications or distributed across major ecosystem actors.

This attribution step identified Russian CA inclusion across applications associated with major ecosystem actors such as Ozon, Yandex, VK, MTS, Mail.ru, and 2GIS.

4.8 Threat Relevance

Our measurement is motivated by a threat model in which a network adversary can intercept TLS connections if it can present certificates chaining to a certificate authority trusted by the application.

Under standard WebPKI assumptions, TLS security depends on the integrity and independence of the certificate authority ecosystem [2]. In the Russian setting, the relevant question is whether trust anchors associated with the national PKI have propagated into the application layer in ways that expand the set of certificate authorities recognized by mobile applications, especially among the applications most likely to be installed and used at scale.

If applications include such trust anchors and a network adversary is capable of issuing certificates from those authorities, TLS connections may be intercepted through a TLS proxy while remaining cryptographically valid from the client's perspective.

5 Results

We now examine how frequently applications distributed through RuStore bundle additional certificate material and how often those certificates include Russian state trust anchors. We first report results from the full RuStore corpus in order to characterize ecosystem-wide prevalence, distribution, and trust-configuration patterns. We then present a separate top-app weighted analysis to determine whether these patterns are concentrated in the applications most likely to shape real-world user exposure.

5.1 Certificate Bundling Prevalence in the Full Corpus

Across the 1,394 APKs in our corpus, we observe widespread inclusion of certificate material within application packages. As Table 1 shows, nearly half of all analyzed applications bundle at least one certificate or trust store within the APK itself. More notably, over one quarter of the applications bundle the Russian Trusted Root CA or its subordinate certificate among their artifacts.

Table 1: Certificate bundling in RuStore applications

Category	Count	Percent
Total analyzed	1,394	–
Bundling any certificate	691	49.6%
Bundling Russian Trusted Root/Sub CA	399	28.6%
Bundling Huawei HMS trust stores	191	13.7%
Bundling both Russian CA and Huawei HMS	127	9.1%

These results indicate that application packages frequently include certificate material that can extend or otherwise modify the set of certificate authorities available to the application beyond those present in the operating system trust store.

5.2 Russian CA Distribution Across the Ecosystem

To determine whether Russian trust anchors appear only in isolated applications or across recognizable application families, we grouped applications by package prefix. Table 2 reports both app-level prevalence and certificate-record counts. The app-level columns count each APK once, while the certificate-record column counts repeated Russian CA artifacts within APKs. This distinction avoids conflating repeated certificate files with unique applications while preserving the structural signal that some application families package Russian CA material multiple times across resource paths.

Table 2: Russian CA prevalence and repeated certificate artifacts by package prefix

Package Prefix	CA Apps	Total Apps	Percent	CA Records
`ru.yandex`	14	14	100.0%	17
`com.yandex`	13	13	100.0%	18
`ru.ozon`	12	15	80.0%	26
`com.vk`	6	6	100.0%	17
`ru.mts`	4	4	100.0%	13
`ru.mail`	4	4	100.0%	11
`ru.dublgis`	2	2	100.0%	22
`ru.sberins`	1	1	100.0%	12

These package prefixes correspond to major components of the Russian digital ecosystem, including e-commerce platforms, mapping services, telecommunications providers, social networks, and large technology platforms. The presence of Russian trust anchors across these prefixes suggests that the certificates appear in applications associated with widely used services rather than being confined to isolated or obscure packages.

5.3 Supply-Chain Distribution of Certificates

To determine whether the Russian CA is included independently by individual developers or propagated through shared dependencies, we analyzed the internal paths under which the certificates appear

within APK resources. Table 3 lists the most common file paths containing Russian CA certificates.

As Table 3 shows, the most common path, `res/raw/rootca_ssl_rsa2022.cer`, appears in 81 distinct applications. The recurrence of identical paths across many different applications suggests that the certificate is introduced through shared SDK dependencies, libraries, or build pipelines rather than through independent developer decisions.

Table 3: Most common file paths containing Russian CA certificates

Path	Apps
res/raw/rootca_ssl_rsa2022.cer	81
res/Y0.cer	54
res/raw/russian_trusted_root_ca.cer	53
res/raw/russian_trusted_sub_ca.cer	51
res/WN.cer	51

This pattern is consistent with software supply-chain propagation, where trust anchors introduced by a dependency become embedded across many downstream applications [15].

5.4 Secondary Trust-Store Artifacts: HMS

We also examine the overlap between applications bundling Russian trust anchors and those including Huawei Mobile Services (HMS) trust-store artifacts. We include HMS because it is a common, recognizable example of SDK-associated trust material in the corpus, not because we treat it as equivalent to the Russian Trusted Root CA or as carrying the same threat-model implications.

We identified 191 applications containing Huawei trust-store artifacts such as `hmsrootcas.bks`. Among these, 127 applications also include the Russian Trusted Root CA. This overlap indicates that part of the RuStore application ecosystem includes artifacts associated with both Russian PKI infrastructure and Huawei mobile service components within the same application packages.

We treat this as a secondary structural observation about layered trust configuration rather than as the central result. It does not imply a unified trust policy or coordinated operational relationship between these infrastructures. Instead, it illustrates the compositional nature of application-level trust configuration, where regional platforms, SDKs, and packaged certificate stores can coexist inside the same APK.

5.5 Trust Configuration Artifacts

While certificate bundling provides one indication of expanded trust boundaries, Android applications may also modify TLS trust behavior through explicit security configuration mechanisms such as `network_security_config`, certificate pinning, or policies governing acceptance of user-installed certificate authorities.

Table 4 summarizes these artifacts among applications bundling the Russian CA.

To better understand how bundled certificates interact with application trust configuration, we examined additional trust-related artifacts within the APK corpus. Certificate bundling alone does not reveal how an application handles trust decisions in practice, so we looked for additional trust-configuration mechanisms that clarify how bundled trust anchors fit into broader application trust behavior.

Table 4: Trust artifacts in apps bundling the Russian CA

Artifact	Apps
Apps bundling Russian CA	399
Apps with `network_security_config`	304
Apps defining custom trust anchors	130
Apps with certificate pinning indicators	294
Apps with pinning + network policy	226

First, we identified applications that define an explicit `network_security_config` policy. Among the 399 applications bundling the Russian Trusted Root or Subordinate CA, 304 include a `network_security_config` file defining custom trust configuration. Of these, 130 applications explicitly define custom certificate trust anchors within the configuration. This indicates that bundled Russian trust anchors often appear alongside application-level trust policies rather than as isolated embedded files. Applications that bundle the Russian CA but do not expose an explicit `network_security_config` may still load certificate material through application code or embedded libraries, or the certificate may be present as an unused or inherited artifact.

Second, we searched for indicators of certificate pinning logic within application code, including references to libraries such as OkHttp's `CertificatePinner` [33]. Among the applications bundling the Russian CA, 294 include code references associated with certificate pinning mechanisms. Of these, 226 simultaneously define a `network_security_config` policy. The co-occurrence of pinning indicators and explicit network security configuration suggests that many of these applications combine multiple trust mechanisms rather than relying on a single source of trust policy.

Finally, we examined applications that explicitly trust user-installed certificate authorities. Across the full corpus, 271 applications include policies allowing certificates installed by the user to act as trusted roots. Among these, 134 applications also bundle the Russian Trusted Root CA.

Overall, these artifacts show that trust relationships within the RuStore ecosystem often differ from the default operating system trust store. Applications commonly combine bundled certificates, custom network security policies, certificate pinning logic, and user certificate trust, producing trust configurations that differ from standard WebPKI assumptions.

We now ask whether this pattern is concentrated in the applications most likely to matter in practice.

5.6 Top-Application Exposure Weighted by Download Share

Raw APK counts do not fully capture user exposure. We therefore weight the top-app analysis by download share to distinguish long-tail prevalence from concentration in applications most likely to shape exposure at scale.

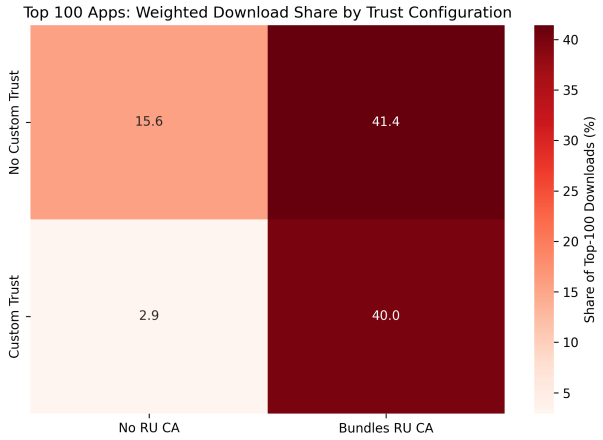

Figure 1: Weighted distribution of top-app download share by Russian CA inclusion and whether an application was classified as having custom trust configuration. Russian CA bundling is concentrated in the most widely used applications, especially among apps in the custom-trust-configuration category.

As Figure 1 shows, Russian state-associated trust anchors are heavily concentrated in the most widely used applications. Across the top-app subset, 81.4% of weighted download share is associated with applications that bundle the Russian Trusted Root or Subordinate CA, compared with 18.5% for applications that do not. This concentration is especially pronounced among applications classified as having custom trust configuration: within that group, 93.2% of weighted download share is associated with apps bundling the Russian CA, compared with 6.8% for apps in the same category that do not. Even among applications not classified as having custom trust configuration, the weighted distribution still favors Russian CA inclusion: 72.6% of weighted share in that group comes from apps bundling the Russian CA.

These results show that Russian trust-anchor propagation is not merely a long-tail property of the RuStore ecosystem. It is concentrated in the applications most likely to shape user exposure at scale, and is especially pronounced among apps classified as having custom trust configuration.

6 Discussion

This section discusses what our measurements imply for application-level TLS trust, software supply-chain propagation, and the limits of treating TLS as a fallback defense against hostile network infrastructure.

6.1 Application-Level Trust Expansion

Modern mobile operating systems maintain curated trust stores that define which certificate authorities may authenticate TLS connections. These stores are typically governed by platform vendors and subject to external scrutiny. In practice, however, applications may include additional certificate authorities within their own packages, define custom trust policies, or dynamically load trust stores during runtime [12].

Our results show that this is not marginal in the RuStore ecosystem. In the full corpus, nearly half of the analyzed applications bundle certificate material, and more than one quarter include the Russian Trusted Root CA or its subordinate certificate. In many cases, these trust anchors co-occur with additional trust-configuration artifacts such as network security policies or certificate pinning indicators. The top-app weighted analysis sharpens this point further: Russian CA bundling is not merely common, but concentrated in the applications most likely to shape user exposure at scale.

When applications include additional trust anchors, the effective set of certificate authorities available to the application may differ from those present in the operating system trust store. As a result, the trust relationships experienced by users are shaped not only by platform governance but also by application packaging, embedded resources, and dependency choices. In this sense, the practical trust boundary of TLS extends beyond the operating system and into the application layer itself.

6.2 Software Supply-Chain Propagation

The clustering of identical certificate artifacts across many unrelated applications points to shared software components rather than independent developer decisions.

Several certificate paths appear repeatedly across dozens of applications, indicating that the Russian CA is likely distributed through SDKs, shared libraries, or build tooling integrated into application development pipelines. This resembles other forms of supply-chain propagation observed in mobile ecosystems, where a dependency introduced by one component can appear across large numbers of downstream applications [15].

This changes how trust enters the ecosystem. If certificates were being added independently by each developer, the resulting trust relationships might be interpreted as a series of isolated local decisions. When trust anchors propagate through shared dependencies, they can spread broadly across the application layer with limited visibility and little end-user awareness. Decisions made upstream in one library or build pipeline can therefore reshape trust across many downstream applications.

6.3 Interactions with Alternative Mobile Infrastructure

Our measurements also reveal overlap between applications bundling Russian trust anchors and those incorporating Huawei Mobile Services (HMS) trust stores. In particular, 127 applications in the corpus contain artifacts associated with both ecosystems.

We do not claim that this overlap alone establishes a unified trust architecture or a shared operational policy. However, it shows that parts of the RuStore ecosystem are assembled from components associated with multiple alternative mobile infrastructures. As software distribution ecosystems diversify beyond the traditional Google Play model, application trust relationships may increasingly reflect combinations of regional platforms, SDKs, and service dependencies rather than a single globally governed stack.

For this paper, the Huawei overlap is a secondary structural observation: it shows that application trust configuration in the

RuStore ecosystem is not only expanded, but layered and compositional.

6.4 Implications for TLS Security Assumptions

TLS is often invoked as the reason encrypted application traffic remains secure against on-path adversaries. That intuition is only partially correct. The security guarantees of TLS depend not only on cryptographic protocol design but also on which authorities the client is willing to trust [2].

Our results show that these trust relationships are shaped not only by operating system vendors, but also by application packaging practices and software supply chains. When certificate authorities associated with national infrastructure propagate into application packages, they expand the set of authorities capable of authenticating TLS connections within those applications. In such settings, the familiar fallback claim—that TLS still protects users even if the network is hostile—becomes less stable, because the relevant trust boundary may already have moved into the APK.

This point is especially important in environments where network operators or state-linked infrastructure possess the ability to issue certificates from such authorities. In those cases, encrypted connections may remain cryptographically valid even when traffic terminates at an intermediary trusted by the client. The issue is therefore not a break in TLS as a protocol, but a redistribution of trust that changes who can successfully impersonate the endpoint without triggering certificate errors.

6.5 Limitations

This study has several limitations.

First, our analysis relies primarily on static inspection of APK files and does not by itself determine whether bundled certificates are actively exercised during TLS validation at runtime. Applications may include certificate material for purposes unrelated to TLS authentication, and additional dynamic analysis would be required to determine how these artifacts influence specific connections in practice.

Second, our dataset is limited to applications distributed through the RuStore ecosystem. While RuStore is an important distribution channel within the Russian mobile ecosystem, applications obtained through other marketplaces may exhibit different trust configurations. At the same time, this focus is intentional: our goal is not to characterize all Android applications everywhere, but to study one politically and infrastructurally meaningful ecosystem in depth.

We also do not compare RuStore packages against matched versions of the same applications from Google Play, other Android marketplaces, or iOS application stores. Such a comparison would help separate trust material that is specific to RuStore-distributed builds from certificate and trust-configuration practices already present in upstream or cross-marketplace releases. We leave this matched-build comparison to future work, including the question of whether Russian trust anchors are introduced in RuStore-specific builds, inherited from vendor-controlled release processes, or distributed across multiple application channels.

Finally, our study does not attempt to evaluate the operational policies of the certificate authorities identified in the dataset. The presence of a certificate within an application package does not by itself imply active traffic interception or any particular issuance practice.

Despite these limitations, the measurements presented here show that certificate authorities associated with national infrastructure can propagate widely through application ecosystems, and that this propagation is especially pronounced among high-exposure applications. These findings highlight the importance of examining trust relationships not only at the level of the operating system trust store, but also within application packaging, dependency chains, and software supply networks.

7 Conclusion

In this paper, we examined how certificate authorities associated with national infrastructure propagate into a mobile application ecosystem through bundled trust anchors and related trust configuration artifacts. Using a corpus of 1,394 Android applications distributed through the RuStore marketplace, we conducted a static analysis of certificate material embedded in APKs and measured how frequently applications extend trust beyond the operating system trust store.

Our results show that application-level trust expansion is common in the RuStore ecosystem. Nearly half of the analyzed applications bundle certificate material, and more than one quarter include the Russian Trusted Root CA or its subordinate certificate among their bundled artifacts. Recurrent certificate file paths across many unrelated applications indicate propagation through shared SDKs, libraries, or build pipelines rather than isolated developer decisions. We also find that Russian trust anchors frequently co-occur with explicit trust-configuration artifacts, and that this pattern is concentrated in the applications most likely to shape user exposure at scale.

These findings highlight a broader structural property of TLS security. TLS is often treated as the mechanism that preserves security against hostile networks, but its effective guarantees depend not only on protocol design or operating system trust stores, but also on which authorities applications choose to trust. When applications bundle additional trust anchors, the practical boundary of TLS trust moves outward from platform governance into APK contents, dependency chains, and software supply networks.

As states increasingly develop domestic digital infrastructure, including national certificate authorities, application marketplaces, and alternative mobile service ecosystems, the interaction between software supply chains and trust-anchor distribution may play an increasingly important role in shaping the security properties of mobile environments. Future work should investigate how bundled trust anchors influence runtime TLS validation behavior and how platform governance mechanisms might better account for trust relationships introduced at the application layer.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. CNS-2452885. We thank the anonymous reviewers for their constructive feedback, which improved the clarity and quality of this work.

References

[1] Paul Black and Robert Layton. Be careful who you trust: Issues with the public key infrastructure. *Proceedings - 5th Cybercrime and Trustworthy Computing Conference, CTC 2014*, pages 12–21, 04 2015.

[2] Jeremy Clark and Paul Oorschot. Sok: Ssl and https: Revisiting past challenges and evaluating certificate trust model enhancements. pages 511–525, 05 2013.

[3] Xavier de Carné de Carnavalet and Paul C. van Oorschot. A survey and analysis of tls interception mechanisms and motivations: Exploring how end-to-end tls is made "end-to-me" for web traffic. 55(13s), July 2023.

[4] Zakir Durumeric, Zane Ma, Drew Springall, Richard Barnes, Nick Sullivan, Elie Bursztein, Michael Bailey, J. Alex Halderman, and Vern Paxson. The security impact of https interception. In *Proceedings of the Network and Distributed System Security Symposium*, 2017.

[5] Daniel Díaz-Sánchez, Andrés Marín-Lopez, Florina Almenárez Mendoza, Patricia Arias Cabarcos, and R. Simon Sherratt. Tls/pki challenges and certificate pinning techniques for iot and m2m secure communications. *IEEE Communications Surveys & Tutorials*, 21(4):3502–3531, 2019.

[6] Mohamed A. El-Zawawy, Eleonora Losiouk, and Mauro Conti. Vulnerabilities in android webview objects: Still not the end! *Computers & Security*, 109:102395, 2021.

[7] Roya Ensafi, Philipp Winter, Abdullah Mueen, and Jedidiah R Crandall. Analyzing the great firewall of china over space and time. *Proceedings on privacy enhancing technologies*, 2015.

[8] Alena Epifanova. Deciphering russia's "sovereign internet law": Tightening control and accelerating the splinternet. Technical Report 2, German Council on Foreign Relations (DGAP), January 2020.

[9] Freedom House. Russia: Freedom on the net 2023 country report. https://freedomhouse.org/country/russia/freedom-net/2023, 2023. Accessed: 2026-03-09.

[10] Sergey Frolov and Eric Wustrow. The use of tls in censorship circumvention. *Proceedings 2019 Network and Distributed System Security Symposium*, 2019.

[11] Martin Georgiev, Subodh Iyengar, Suman Jana, Rishita Anubhai, Dan Boneh, and Vitaly Shmatikov. The most dangerous code in the world: validating ssl certificates in non-browser software. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, page 38–49, New York, NY, USA, 2012. Association for Computing Machinery.

[12] Google. Network security configuration, 2024.

[13] Devashish Gosain, Kartikey Singh, Rishi Sharma, Jithin S, and Sambuddho Chakravaty. Out in the open: On the implementation of mobile app filtering in India. In *Passive and Active Measurement Conference*. Springer, 2024.

[14] Daria Gritsenko, Mariëlle Wijermars, and Mikhail Kopotev, editors. *The Palgrave Handbook of Digital Russia Studies*. Springer International Publishing, 2021.

[15] Sana Habib, Mohammad Taha Khan, and Jedidiah R. Crandall. Examining leading pakistani mobile apps. *Proceedings on Privacy Enhancing Technologies (PoPETS)*, 2025(1):24–41, 2025.

[16] Huawei. Huawei mobile services core documentation, 2024.

[17] Salabat Khan, Fei Luo, Zijian Zhang, Farhan Ullah, Farhan Amin, Syed Furqan Qadri, Md Belal Bin Heyat, Rukhsana Ruby, Lu Wang, Shamsher Ullah, Meng Li, Victor C. M. Leung, and Kaishun Wu. A survey on x.509 public-key infrastructure, certificate revocation, and their modern implementation on blockchain and ledger technologies. *IEEE Communications Surveys & Tutorials*, 25:2529–2568, 2023.

[18] Jongkil Kim. Studies on inspecting encrypted data: Trends and challenges. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 14(1):189–199, March 2023.

[19] Jeffrey Knockel, Lotus Ruan, and Masashi Crete-Nishihata. Measuring decentralization of Chinese keyword censorship via mobile games. In *Free and Open Communications on the Internet*. USENIX, 2017.

[20] Vitaliy Konyukhov. Analysis of russian mobile applications accepting the russian trusted root certificate authority. Master's thesis, Arizona State University, 2024.

[21] Beau Kujath, Jeffrey Knockel, Paul Aguilar, Diego Morabito, Masashi Crete-Nishihata, and Jedidiah R. Crandall. Analyzing prominent mobile apps in Latin America. In *Free and Open Communications on the Internet*, 2024.

[22] Tongxin Li, Xueqiang Wang, Mingming Zha, Kai Chen, XiaoFeng Wang, Luyi Xing, Xiaolong Bai, Nan Zhang, and Xinhui Han. Unleashing the walking dead: Understanding cross-app remote infections on mobile webviews. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 829–844, New York, NY, USA, 2017. Association for Computing Machinery.

[23] Mozilla. Certificate cannot be trusted warning in kazakhstan. https://support.mozilla.org/en-US/kb/certificate-cannot-be-trusted-warning-kazakhstan, 2019. Accessed: 2026.

[24] Mozilla. Bug 1680927: Mitm in kazakhstan. https://bugzilla.mozilla.org/show_bug.cgi?id=1680927, 2020. Mozilla Bugzilla.

[25] Marten Oltrogge, Nicolas Huaman, Sabrina Klivan, Yasemin Acar, Michael Backes, and Sascha Fahl. Why eve and mallory still love android: Revisiting TLS (In)Security in android applications. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 4347–4364. USENIX Association, August 2021.

[26] Aaron Ortwein, Kevin Bock, and Dave Levin. Towards a comprehensive understanding of Russian transit censorship. In *Free and Open Communications on the Internet*, 2023.

[27] Amogh Pradeep, Muhammad Talha Paracha, Protick Bhowmick, Ali Davanian, Abbas Razaghpanah, Taejoong Chung, Martina Lindorfer, Narseo Vallina-Rodriguez, Dave Levin, and David Choffnes. A comparative analysis of certificate pinning in android & ios. IMC '22, page 605–618, New York, NY, USA, 2022. Association for Computing Machinery.

[28] Ram Sundara Raman, Leonid Evdokimov, Eric Wustrow, J. Alex Halderman, and Roya Ensafi. Investigating large scale https interception in kazakhstan. In *Proceedings of the ACM Internet Measurement Conference*, IMC '20, page 125–132, New York, NY, USA, 2020. Association for Computing Machinery.

[29] Francisco José Ramírez-López, Ángel Jesús Varela-Vaca, Jorge Ropero, Joaquín Luque, and Alejandro Carrasco. A framework to secure the development and auditing of ssl pinning in mobile applications: The case of android devices. *Entropy*, 21(12), 2019.

[30] Eric Rescorla. The transport layer security (tls) protocol version 1.3. RFC 8446, 2018.

[31] Danil Shamsimukhametov, Anton Kurapov, Mikhail Liubogoshchev, and Evgeny Khorov. Is encrypted clienthello a challenge for traffic classification? *IEEE Access*, 10:77883–77897, 2022.

[32] Wei Song, Qingqing Huang, and Jeff Huang. Understanding javascript vulnerabilities in large real-world android applications. *IEEE Transactions on Dependable and Secure Computing*, 17(5):1063–1078, 2020.

[33] Square. Okhttp: Http client for android and java, 2024.

[34] Vasilis Ververis, Marios Isaakidis, Valentin Weber, and Benjamin Fabian. Shedding light on mobile app store censorship. In *User Modeling, Adaptation and Personalization*. ACM, 2019.

[35] VK. Rustore: Russian android application marketplace. https://www.rustore.ru/, 2024. Accessed: 2026.

[36] Yingjie Wang, Guangquan Xu, Xing Liu, Weixuan Mao, Chengxiang Si, Witold Pedrycz, and Wei Wang. Identifying vulnerabilities of ssl/tls certificate verification in android apps with static and dynamic analysis. *Journal of Systems and Software*, 167:110609, 2020.

[37] Diwen Xue, Benjamin Mixon-Baca, S. S. Valdik, Anna Ablove, Beau Kujath, Jedidiah R. Crandall, and Roya Ensafi. TSPU: Russia's decentralized censorship system. In *Proceedings of the 22nd ACM Internet Measurement Conference (IMC '22)*. Association for Computing Machinery, 2022.

[38] Pamela Zave and Jennifer Rexford. Patterns and interactions in network security. *ACM Comput. Surv.*, 53(6), December 2020.

[39] Lei Zhang, Zhibo Zhang, Ancong Liu, Yinzhi Cao, Xiaohan Zhang, Yanjun Chen, Yuan Zhang, Guangliang Yang, and Min Yang. Identity confusion in WebView-based mobile app-in-app ecosystems. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1597–1613, Boston, MA, August 2022. USENIX Association.