Digital Finance and Data Security

How Private and Secure Is Data Used in Digital Finance?

September 2018

AUTHOR

Patrick Traynor

CENTER *for*
FINANCIAL
INCLUSION | ACCION

Acknowledgements

We gratefully acknowledge the generous support provided by the Center for Financial Inclusion at Accion, without which this work would not have been possible. We would particularly like to thank Sonja Kelly, Director of Research, and Pablo Antón Díaz, Research Manager, for not only helping us to work productively with security stakeholders around the world, but also for their tireless efforts to ensure that these issues are prioritized and addressed.

We also wish to thank Jasmine Bowers, Kevin Butler, and Imani Sherman of the University of Florida, all of whom made significant contributions to the successful completion of this work.

Introduction

Data Privacy and Security Issues in Online Lending

Mobile phones and networks are transforming the world of finance, creating opportunities for widespread financial inclusion, especially among neglected regions and groups. Digital credit offers a particularly important lifeline. Individuals and small businesses in these settings often suffer from the inability to acquire financial assistance. As a consequence, a farmer may not be able to repair the vehicle needed to bring goods to market; a merchant may not be able to fully stock shelves; or an individual may not be able to pay for an unexpected medical procedure.

There are a number of challenges to delivering credit in these scenarios. Efforts to simply reapply existing lending mechanisms have failed as individuals and businesses in these regions and groups often lack the data that lenders look for in more formal settings to make credit decisions, e.g., audited tax forms, pay stubs, property ownership documents, etc. A rapidly-increasing number of companies have created alternative means of measuring creditworthiness based on observing transactions made through mobile applications and other data. Such systems hold the potential to dramatically expand financial inclusion, but if they are poorly designed and executed, they may discourage millions from participating in the modern digital economy.

Around the world, digital lending companies are emerging and growing. They offer digital loans through mobile phone apps or via websites, including those that are optimized for mobile. These digital loan products are varied: many are aimed at consumers; others focus on small enterprises. Many of these companies are reaching out to customers at the base of the economic pyramid. Amounts and loan maturities vary from very short-term “nano” loans of a few dollars to medium-term small business loans of a few hundred or some thousands of dollars. Some companies have grown to substantial—even massive—scale. Others are just starting out. All these companies have at least two things in common: they use online and mobile tools to connect with customers, and they use a range of customer data, obtained electronically, in making their credit decisions.

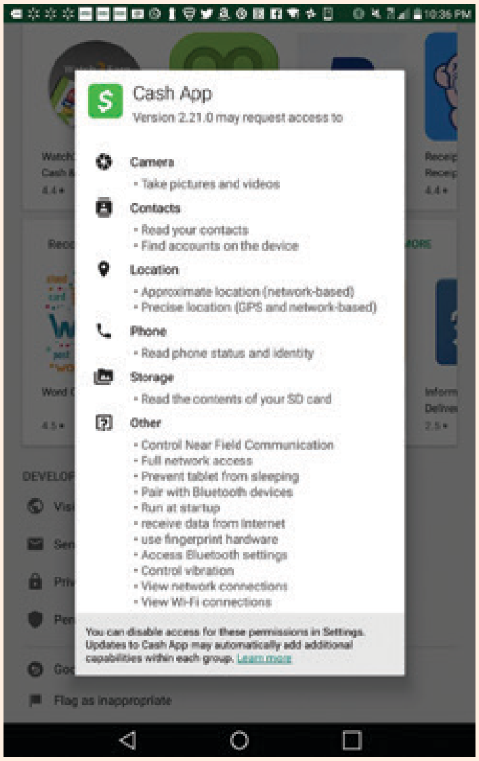
Security and privacy should be among the most important considerations when building digital finance systems. Credit decisions are often based on sensitive information and online finance offerings are no exception. Examples of data used in credit determinations include a person’s history of purchases, movements through time and space (enabled by GPS data), and online activity (such as downloaded applications). The sensitivity of this information gives rise to a series of critical questions for customers:

- To whom am I giving my data? And who else do they allow to access it? For what purposes?
- How do the companies protect data so that people who do not have legitimate access cannot use or steal it?
- What rights and options do I have if something does go wrong?

The goal of this research is to examine how well digital lenders are responding to these questions, and the main sections of the paper are organized accordingly. To address the first question regarding policies for data usage, we performed a thorough analysis

FIGURE 1

Example of a Permission Screen from a Mobile Money App in Android



Cash App

Version 2.21.0 may request access to

Camera

- Take pictures and videos

Contacts

- Read your contacts
- Find accounts on the device

Location

- Approximate location (network-based)
- Precise location (GPS and network-based)

Phone

- Read phone status and identity

Storage

- Read the contents of your SD card

Other

- Control Near Field Communication
- Full network access
- Prevent tablet from sleeping
- Pair with Bluetooth devices
- Run at startup
- receive data from Internet
- use fingerprint hardware
- Access Bluetooth settings
- Control vibration
- View network connections
- View Wi-Fi connections

You can disable access for these permissions in Settings. Updates to Cash App may automatically add additional capabilities within each group. Learn more

Note Cash App was not part of the list of applications that were reviewed in this study.

of the privacy policies of a representative cross-section of 52 lenders in the digital finance industry. We evaluated whether such policies covered important issues such as the conditions under which personal consumer data is shared with third parties, the timeframes for data retention and minimization, and the information on these policies provided to clients. To examine data security, we performed a thorough analysis of the characteristics of the connection between client mobile devices and the digital lender's own servers for 27 selected companies. While it is possible to measure the security of many parts of an online finance system, the most critical piece—and the one evaluated here—is ensuring that the connections between customers' mobile devices and the lender's servers are secure. Finally, we evaluated the terms of service for each of the online finance offerors to characterize the impact on a customer, should fraud occur.

It is critical to characterize the state of security and privacy in this burgeoning space and to identify potential weaknesses and areas for improvement, with the overall goal of protecting system users and strengthening the defense barriers that will enable this new range of online finance providers to become trustworthy, secure, and reliable. Security claims do little to protect real users unless they are backed by strong mechanisms and adequate privacy policies. Independent evaluations—such as the one detailed in this paper—not only help consumers identify the firms currently meeting the bar for best practices, they also push other firms toward the implementation of better industry standards for consumer and asset protection in the digital financial sector.

Thoroughly identifying the security flaws and vulnerabilities of an online system is no easy task, and no single analysis can measure all relevant aspects of security and privacy. Accordingly, this work should be viewed only as a first step. We also recognize that these companies vary widely in scale and maturity: while we hope to see good security practices everywhere, we note that early stage companies may be juggling many issues. We would expect more robust security from larger, more established companies.

It is our hope that companies, consumers and regulators can use this work to plot the path toward improved security standards for online finance. Together, we can ensure the transformative power of these systems while minimizing risks to all parties. With the help of Accion and its investment fund team at Accion Venture Lab, we reached out to all of the companies included in this report prior to its publication to share with them our findings, provide useful guidelines and tips on how to correct the most common vulnerabilities that were uncovered by this study, and offer them a chance to comment on these findings. The next section of the paper contains more details on the results of this consultative process.

Digital Finance Providers Evaluated

Digital finance companies exist in a wide array of markets across the world. While many focus on small businesses, others work with individuals. Our analysis characterizes the security and privacy practices of a wide swath of this industry, including both leading names in the industry (such as Tencent in China and Lending Club in the United States) and smaller efforts (such as Coins in the Philippines and Branch in Kenya). Table 1 lists the 52 digital finance companies (14 US and 37 from other countries) that we analyzed for this project.

We considered multiple factors in our selection. First, all the analyzed companies have a mobile application or version of their website optimized for mobile devices. Second, these companies offer a broad range of geographic coverage, spanning 14 countries across five continents (Figure 2).

This study was not expected to cover every existing digital finance provider, but only a representative sample, both by region and type of product; i.e., small and medium enterprise lending, person-to-person lending, payroll lending, etc. We attempted to balance the above considerations with input from the Center for Financial Inclusion at Accion (CFI), as well as members of the larger financial inclusion community, to pick the most representative sample.

This study and its main findings represent a first step to examine the data security risks of digital finance providers, and we look forward to further case studies into any one of the companies featured here or others not analyzed in this effort that can provide additional insights as well as concrete recommendations.

> Security claims do little to protect real users unless they are backed by strong mechanisms and adequate privacy policies.

FIGURE 2

Countries Represented by Our Selection of Digital Finance Companies

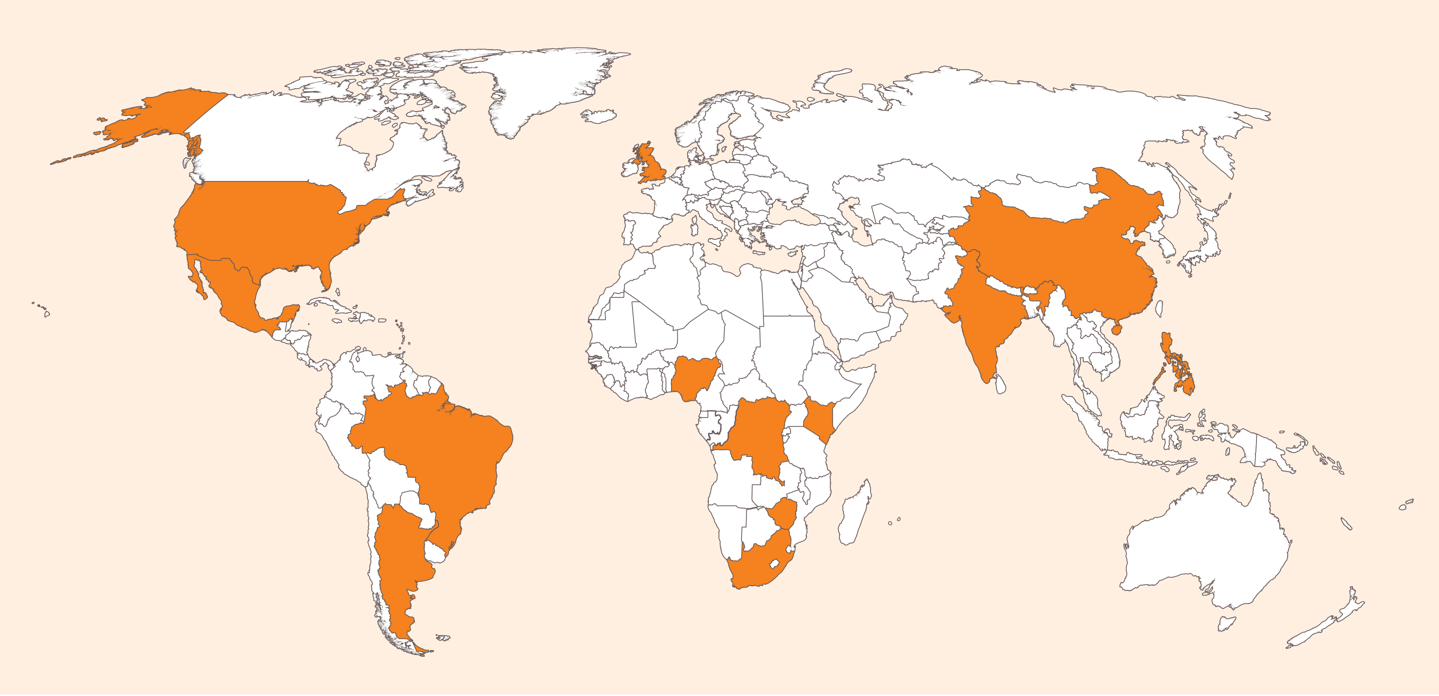


TABLE 1

Digital Lenders Evaluated in This Study

COMPANY	COUNTRY
Airtel	Democratic Republic of Congo
Atom	United Kingdom
Azimo	United Kingdom
BlueVine	United States
Branch	Kenya
C2FO	United States
Coins	Philippines
CommonBond	United States
Creditas	Brazil
CrowdEstates	United Kingdom
EcoCash Loans	Zimbabwe
Equitel	Kenya
Equity Direct	Kenya
Farm Drive	Kenya
FastPay	United States
GetBucks	South Africa
IndiaMART	India
Insikt	United States
InstaPaisa	India
InvoiNet	Argentina
Jimubox	China
Kabbage	United States
KCB	Kenya
Koopkrag	South Africa
Kopo Kopo	Kenya
Kubo Financiero	Mexico
Lending Club	United States
Lulalend	South Africa
M-Pawa	Kenya
MCo-op Cash	Kenya
Micromobile	Kenya
MoneyTap	India
OnDeck	United States
Pay Your Tuition Funds	United States
PaySense	India
PesaPata	Kenya
Prosper	United States
Puddle	United States
RoadLoans	United States
Saida	Kenya
Salud Fácil	Mexico
SMECorner	India
Social Lender	Nigeria
Suregifts Redemption	Nigeria
Tala	Kenya
Taplend	United Kingdom
Tencent	China
Upstart	United States
WeFinance	United States
Yoco	South Africa
Zidisha	Kenya

Privacy Analysis

1

Many observers point to the transition of the information economy as one of the hallmarks of the twenty-first century. In the financial sector—as well as in other industries—"big data" promises to deliver customized services at an accelerated speed, to improve customer satisfaction through personalized product design, and to increase profits through targeted advertisements. However, access to the customer data needed to facilitate these improvements may also have unintended consequences. Personal information that some individuals may prefer to keep private (e.g., pregnancy, sexual orientation, behavior patterns) has become discoverable through the use of modern data analytics techniques. Financial information, affiliations (political, religious, etc.) and other data may also become involved. As such, many consumers look to the acquisition and use of their data with apprehension.

Privacy policies can help alleviate some of this fear. They offer a public explanation of how a company intends to handle user data, the uses to which it may be put, with whom they may share such data, the conclusions that may be drawn, and what rights customers have to correct such information. If a company follows its own privacy policy, the actual risks of improper data use may be reduced. Unfortunately, no two privacy policies are the same and many are vague, unclear, or incomplete.

The first exercise in this study was to characterize privacy policies for digital lenders. We sought to determine whether such policies discuss critical issues, how they compared against traditional financial offerings, and the accessibility or readability of such policies for customers from different backgrounds. An interesting further study question, while not addressed here, is how and when users are exposed to these policies.

Methodology

Our study of privacy policies addressed two high-level questions:

- Do the privacy policies published by digital lenders address the appropriate issues?
- How readable are these policies for their target audiences?

We collected the English-language privacy policies of the 52 selected companies listed above.[1] The policies were available on the webpages or within the applications associated with each company. Instead of inventing our own criteria, we combined guidance by the US Federal Deposit Insurance Corporation (FDIC)[2] and the GSM Association (GSMA)[3] for mobile applications. This approach was valid for many reasons. First, it combined industry and regulatory perspectives on what privacy policies should cover. Second, using these long-published guidelines prevented us from creating arbitrary requirements about which the industry might be unaware. Finally, we previously applied this combination to evaluate the privacy policies of mobile money providers.[4]

The use of FDIC guidelines is not without some debate. Many consider FDIC recommendations as best practices and they are followed by many financial institutions throughout the world (especially if those institutions interact with US-based financial institutions). However, the 38 digital finance providers based elsewhere that were evaluated in this work did not necessarily have any requirement to follow US agency rules. Unfortunately, in this study it was impractical for us to perform an analysis considering each of the 14 possible different national standards. Further insight can be gained into any of the studied countries through follow-on analysis; however, our goal was to develop a relative understanding of performance across the industry.

TABLE 2

Privacy Policy Specifications Considered for This Analysis

BOTH FDIC AND GSMA

User choice and control: Notices should disclose the user's right to opt-out and how users can control the use of their personal information.

Security: Notices should disclose how personal information will be protected and safeguarded.

Data to be shared: Notices should list the personal information of users that may be disclosed.

Data minimization/retention: Information sharing practices of personal information of former customers should be disclosed. Only the minimum amount of user information should be collected, accessed, used, and retained at all times.

GSMA

Purpose of data collection: Policies should disclose the purpose of collecting, accessing, and sharing user data and ensure that each purpose is for legitimate business operations.

Children and adolescents: If applicable to children, the service should guarantee that the child's personal information is properly collected and should abide by all laws related to children's privacy.

Accountability and enforcement: Employees are held accountable for proper use and protection of user data.

FDIC

Collection process: Policies should list the types of personal information that are collected.

Definitions: Policies should define terms concerning collection process, information disclosure, etc.

Examples: Policies should include examples of the collection process, information disclosure, etc.

Third parties: Policies should disclose affiliates with whom the bank shares non-public personal information.

Coding Process

We evaluated privacy policies manually. We created a codebook of words correlated to each of the above policy principles, which is shown in Table 3.

Mention of any of the words contained in the codebook was sufficient to credit the privacy policy with addressing a given principle. This should not be construed as an indication that such policy comprehensively covers a topic; rather, our goal was to determine if a principle was even considered. We took this approach because we sought empirical evidence of compliance with the guidelines mentioned; arguing for or against the quality of coverage beyond this is an important task, but one more appropriately undertaken by more specialized researchers such as policy specialists and consumer rights advocates. Accordingly, the results herein should be viewed as a starting point for discussion and *by no means an endorsement of any of these policies*.

We followed standard methodology used throughout the social sciences for evaluating text. Two individuals or "coders" were tasked with independently evaluating all of the collected documents. If the coder did not find exact keywords in the codebook but did find similar text, the coder was instructed to use best judgment when scoring that principle. If neither the keywords nor similar text were found for a specific principle, the policy received a score of zero.

The results of the independent trials were then compared and mutually reconciled to arrive at the reported result. During the reconciliation process, if the results of the coders diverged (e.g., when judgments about ambiguous text differed), we discussed the instructions and thought process with both coders to determine the final score for each policy and principle. Completion of all the steps in this process was critical to capturing the policy content fully and accurately.

Analysis of Readability

The applied linguistics community has developed several tests to automatically measure the readability of text. These tests produce a “grade level” that is meant to reflect the suggested level of education needed to fully comprehend the body of text. For example, although a gold-standard technique is still debated, it has become common in certain fields to use several tests for one study. Our analysis calculated scores based on the following readability scoring mechanisms: Flesch-Kincaid Grade Level, Gunning Fog Index, Coleman-Liau Index, SMOG Index, and Automated Readability Index. Individual scores were tallied and we calculated an overall average score. In addition, we also calculated the estimated time-to-read and a word count.

Results

Mention of Principles

Figure 3 is a comprehensive representation of our privacy policy content findings. The graph shows three groups: the first two are the selected lenders we reviewed—international and US digital lenders. The third group is US traditional banks, used as a benchmark because they are required to comply with FDIC guidelines.

It is clear from Figure 3 that the policies of the US digital lenders are quite similar in content coverage to those of the traditional US banks. Over three-fourths of the US lenders’ policies covered seven of the 11 categories, while over one-third covered all 11. These results are similar to those of the US traditional banks, where over three-fourths cover eight of the categories and at least one-fifth cover all 11.

By comparison, the international policies covered less content across the board. This may be a result of legal requirements for credit providers in the United States. Among the specific gaps, 65 percent of the international policies failed to include definitions and examples of privacy-related terms. Policies tended to include jargon (e.g., browser cookies) that may not be common knowledge.

TABLE 3

Codebook of Words Used for Each Policy Principle

POLICY SPECIFICATION	KEY WORDS AND PHRASES
Purpose of Data Collection	Reasoning, Enhance User Experience, User Experience
Children and Adolescents	Children, Children’s Privacy
Accountability and Enforcement	Employee, Accountable, Accountability
Collection Process	Collect
Definitions	Means, Is, Are
Examples	Types of Personal Information, Types Of, For Example, Includes
Third Parties	Third Party, Third Parties
User Choice and Control	Disable, Edit, User Can, Change
Security	Security
Sharing Process	Share, Sharing Process
Data Minimization and Retention	Minimization, Termination, Continue To Share, Retention, Retain

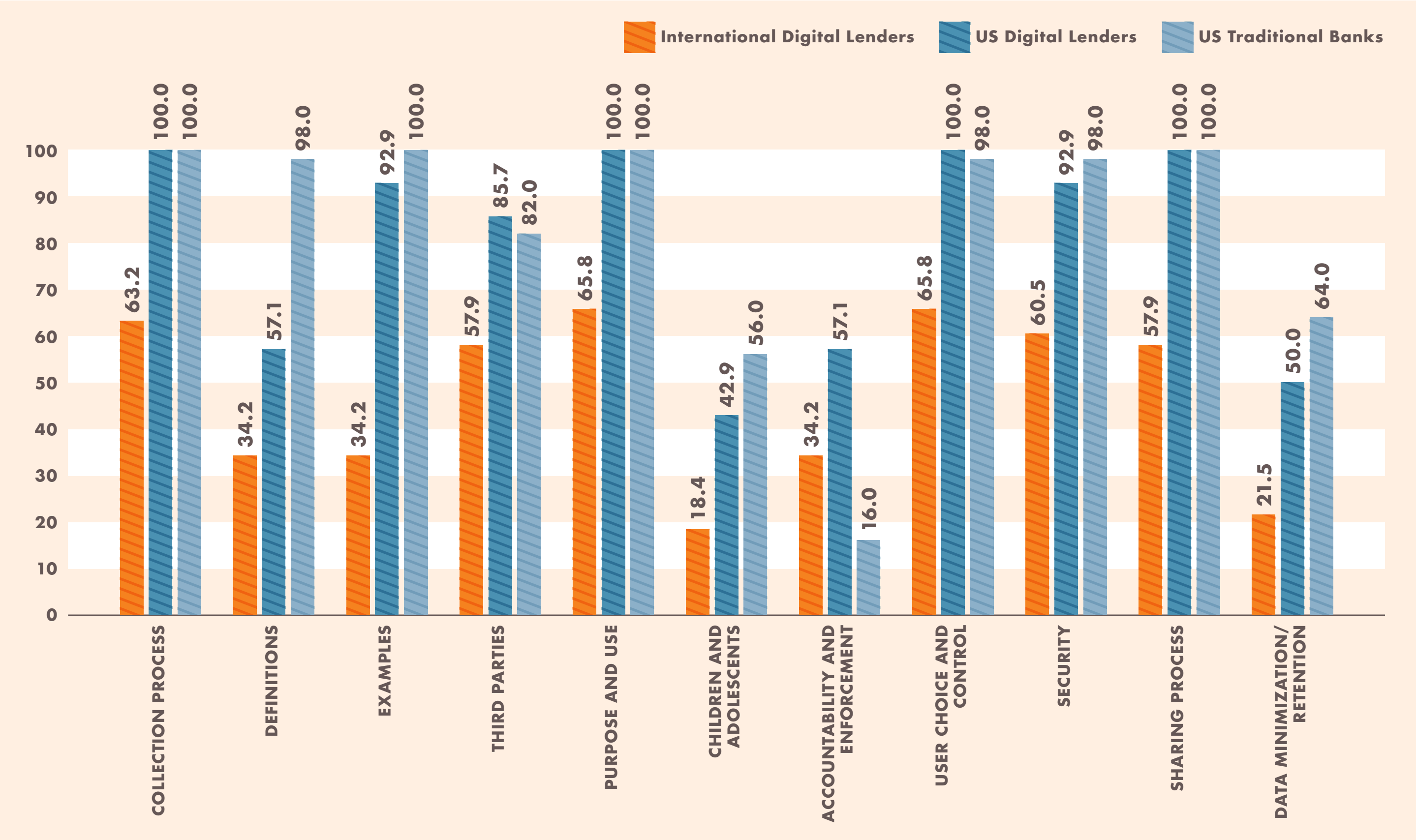
Therefore, to be understood by users, policies need to define terms that relate to user data and privacy.

Another topic frequently omitted was employee accountability. Defining employee accountability gives the user an understanding of what should happen in the event of data mishandling.

Although we in no way assume US bank policies to be the ideal standard, we include the results to indicate the impact of regulation on practice.

FIGURE 3

Content of Privacy Policies (by percentage of total number of companies)



Reading Grade Level Scores

Privacy policy reading grade level statistics represent the average reading grade levels of our three sets of policies (see Figure 4). As a point of reference, it is widely assumed that the average American citizen reads at approximately an eighth grade level. Our measurement showed that the grade levels of the digital lenders' policies were higher than the US traditional bank policies, which we included as a baseline to show impact of regulation. As shown in Table 4, the median reading grade level of the international policies was higher than the median of both sets of US policies. With a median of 13.1 years, customers would need at least one year of higher education to fully grasp the meaning of the policies. In contrast, average school attainment in Nigeria and the Democratic Republic of the Congo, for example, is nine years.[5] Hence, it is important that providers and policy makers aim for content that users of varying educational backgrounds can read and understand.

FIGURE 4

Average Reading Grade Level of Privacy Policies

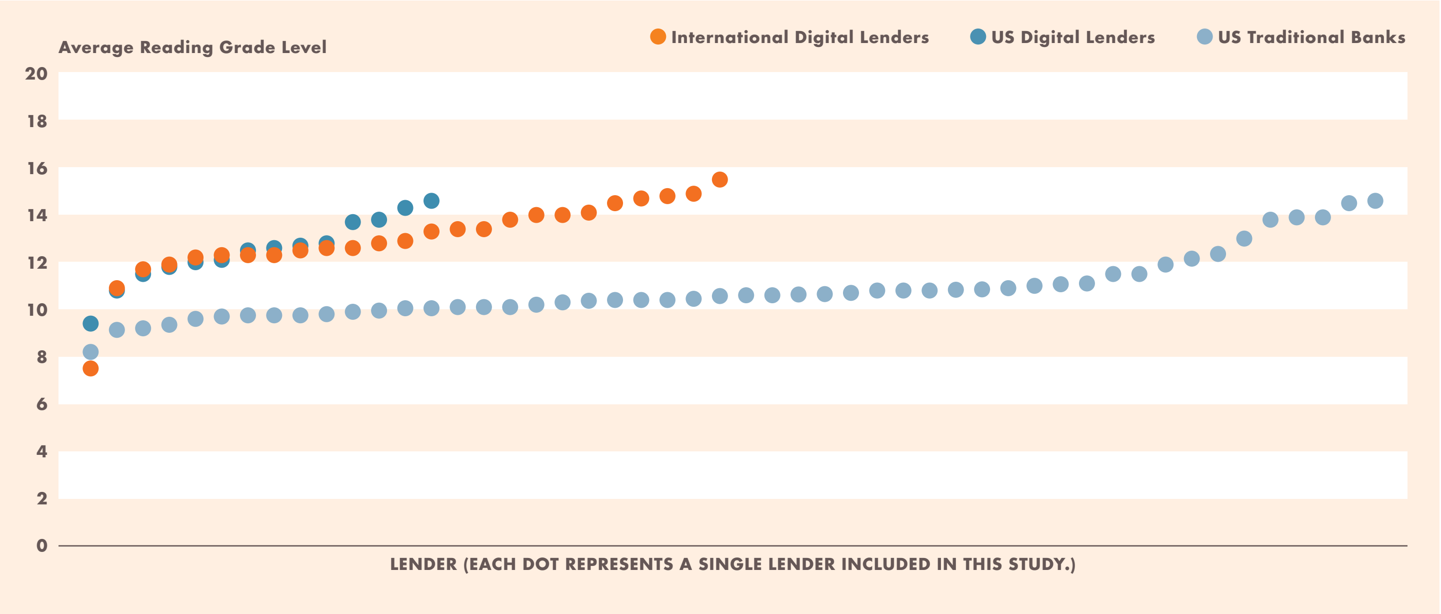


Figure 5 shows the word counts of all three sets of privacy policies. Aside from one outlier policy of over 11,000 words, the word counts ranged between zero and 6,000. As shown in Table 5, the median word counts of each data set were relatively close. However, the minimums were far less similar, with InvoiNet having a word count of 151. In addition, InvoiNet only covered two of the 11 categories of policy content. We found that policies of lower word count are more inclined to cover less user privacy content than those with higher counts. In contrast, we cannot conclusively say that longer policies are fully comprehensive. However, longer policies can reflect effort put forth to fully cover critical policy topics.

TABLE 4

Privacy Policy Reading Grade Level

AVERAGE READING GRADE LEVEL

	INTERNATIONAL DIGITAL LENDERS	US DIGITAL LENDERS	US BANKS
Minimum	7.5 (Atom)	9.4 (RoadLoans)	8.2
Maximum	15.5 (InvoiNet)	14.6 (Fast Pay)	14.6
Median	12.9	12.6	10.6

FIGURE 5

Word Count of Privacy Policies

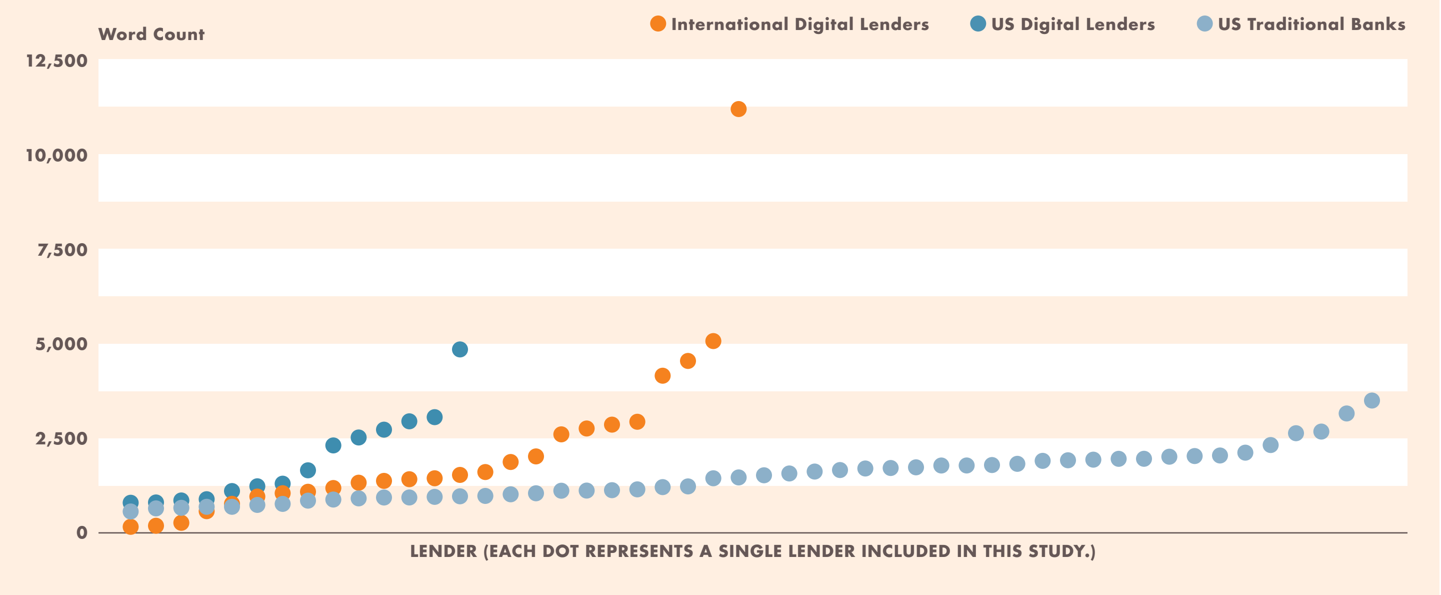


TABLE 5

Privacy Policy Word Count Breakdown

WORD COUNT	INTERNATIONAL DIGITAL LENDERS	US DIGITAL LENDERS	US BANKS
Minimum	151 (InvoiNet)	788 (RoadLoans)	557
Maximum	11,210 (KCB)	4,847 (Prosper)	3,494
Median	1,436	1,470	1,488

Word Count vs. Readability

We also analyzed the correlation between word count and reading grade level for each set of policies (Annex A). We found a direct correlation between the word counts and reading grade levels in the policies of the international lenders: the longer the policies, the higher the reading grade required. In comparison, we did not find a trend in the US lenders' policies. With these findings, we recommend that policy writers assure that their policies, regardless of length, cover the critical best practice topics and attempt to do so in language accessible to the target market.

Conclusions

The international companies' policies had higher average reading grade levels, although they covered less content for nearly almost every principle, as shown in Figure 3. We provide specific policy recommendations at the end of this document.

Security Analysis

2

Security is a critical requirement any time money is involved. Digital financial services offer consumers greater physical security than borrowing, saving, or transacting in cash. This security is in fact one of the important contributions digital financial services can make to people on lower incomes. However, security risks do not disappear when lending moves online; the nature and location of risk moves, too. Adversaries do not simply disappear: as money flows online, adversaries arise that focus their efforts on whatever vulnerabilities providers leave open. Theft and fraud are enabled when data falls into adversarial hands. Moreover, breaches to online lending services may further endanger customers because of the wider range of personal data they may collect (e.g., social networks, GPS information). As such, it is critical that such services provide strong, best-practice protections for their customers.

Before we go on, we will address an important misconception that our analysis simply informs potential adversaries as to where they should attack. We reject this assertion for a number of reasons. First, in the field of information security, this thought process is known as "security by obscurity". The thinking goes that if adversaries do not already know of the existence of a weakness, they never will. Basing the security of a system on the hope that an adversary will never find significant vulnerabilities is problematic because it assumes that an adversary would never bother to look. This is akin to the "key under the doormat" approach to security. Can anyone really pretend to be surprised when the first thing a burglar does when approaching a property is check for a hidden key? Research tells us that nearly all our systems are constantly under an automated barrage of attack traffic and it would be naïve to think that emerging online finance infrastructure would somehow be spared from this onslaught. We do, however, note that larger companies are likely to attract more attention from bad actors.

Second, security by obscurity is entirely unnecessary given that strong solutions to many of these problems exist and at relatively low cost. As discussed below, we are not necessarily recommending that companies invest in expensive anomaly detection systems (although they may wish to consider it); rather, they should deploy and properly configure the strong connection security mechanisms that are already available to them through features of the majority of their computing infrastructure.

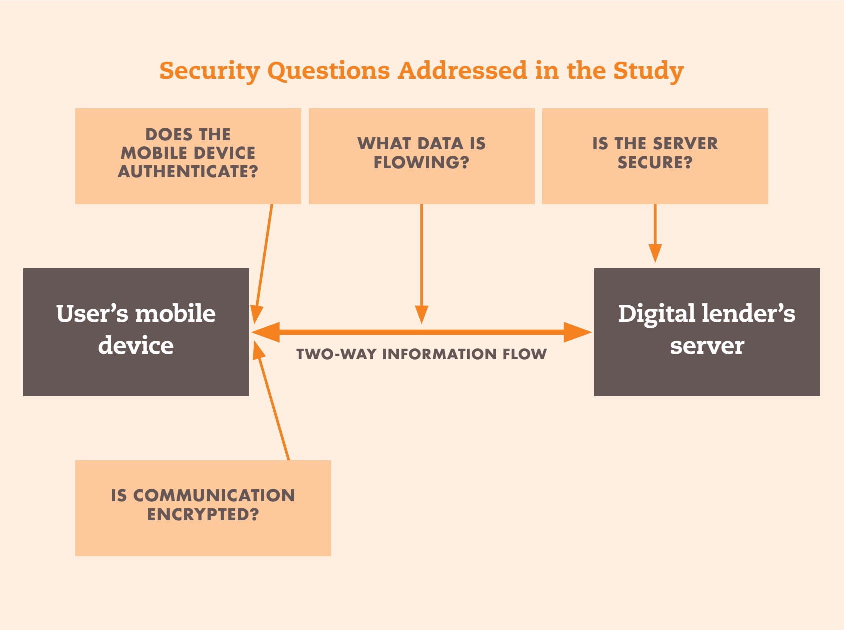
Finally, before this report was published, we reached out to each of the analyzed companies to give them a chance to improve on the discovered problems. Our goal is to ensure that all organizations protect customer data as well as possible, and we are willing to assist them in making the necessary changes.

Methodology

Our study sought to measure the security of the connection between customers' mobile devices and the server within the digital lender's network that processes and stores data. While it is possible to measure the security of many parts of an online finance system, the most critical is ensuring that connections between mobile devices and the service provider's servers are secure. Accordingly, we focused our attention here.

FIGURE 6

Mapping Secure Communications



Measuring the security of connections between users and digital finance providers is the first step in assessing the practices of this industry. If a digital finance provider fails to adequately protect data in this space, an adversary could recover potentially sensitive consumer information with very little effort. Such a breach can obviously entail financial loss. However, it is critical to note that the wide array of data collected by these services could further harm a customer. For instance, GPS data could be used to track specific individuals and target them for extortion or other harm.

Although correctly protecting communications, as discussed in this paper, is a good first step, a positive evaluation here should not be viewed as an endorsement of all security practices of the studied companies; rather, it merely represents that this one aspect is done well. Multiple additional analyses of internal policies and controls, each of which would entail significant additional efforts, remain projects for future research.

Our experiments sought to answer four specific questions about the state of secure communications use by digital finance providers (Figure 6). They are:

1. Do mobile devices properly use strong encryption algorithms to protect the confidentiality and integrity of all communications? (Devices)

2. Do mobile devices properly verify that they are communicating with the correct server? (Authentication)

3. Are the servers configured to use strong encryption algorithms to protect the confidentiality and integrity of all communications? (Servers)

4. What data are being collected by digital finance providers over these links? (Permissions)

A failure at any one of these steps can lead to a successful attack on online finance systems. For Question 1, if a mobile device is programmed to use weak or no encryption algorithms, an attacker may be able to recover the data. For Question 2, if the mobile device is not certain it is communicating with the correct server, it may simply be sending its sensitive data to an attacker. For Question 3, even if the other two issues are properly addressed, a poorly-configured server may undo all of these protections and expose user data. Finally, for Question 4, identifying the data a digital finance provider has permission to take allows us to reason about what data may be exposed should any of the above security weaknesses be confirmed.

We break down our methodology with respect to each of these questions below.

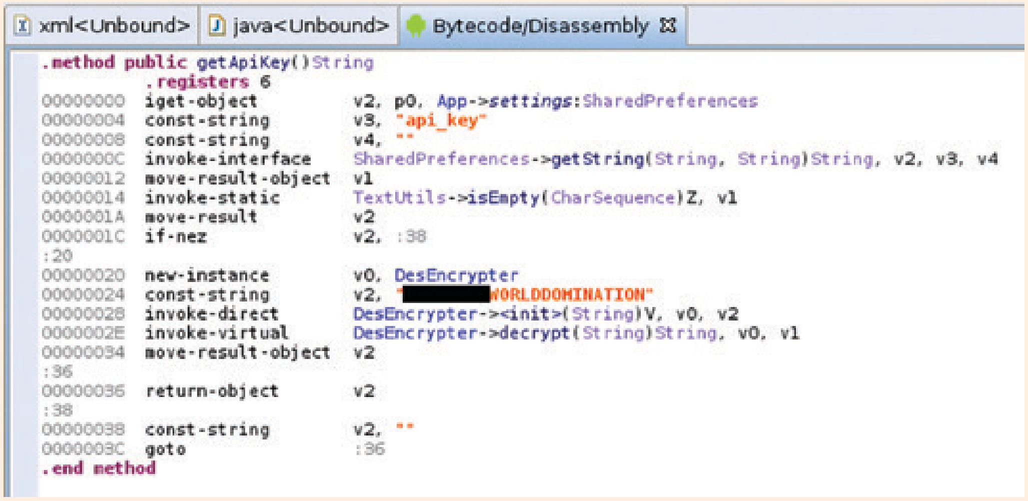
Use of Encryption by Apps on Mobile Devices

1. Do mobile devices properly use strong encryption algorithms to protect the confidentiality and integrity of all communications? (Devices)

Of the 52 digital finance providers in this study, 27 offer a mobile application for Android

FIGURE 7

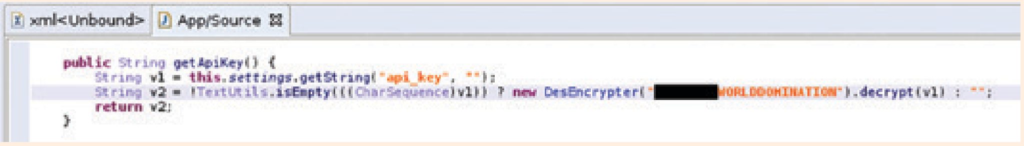
Android application bytecode in the JEB tool. Such code is extremely difficult for an analyst to read and assess.



```
.method public getApiKey()String
            .registers 6
00000000  iget-object          v2, p0, App->settings:SharedPreferences
00000004  const-string         v3, "api_key"
00000008  const-string         v4, ""
0000000C  invoke-interface     SharedPreferences->getString(String, String)String, v2, v3, v4
00000012  move-result-object   v1
00000014  invoke-static        TextUtils->isEmpty(CharSequence)Z, v1
0000001A  move-result          v2
0000001C  if-nez               v2, :38
:20
00000020  new-instance         v0, DesEncrypter
00000024  const-string         v2, "[redacted]WORLDDOMINATION"
00000028  invoke-direct        DesEncrypter-><init>(String)V, v0, v2
0000002E  invoke-virtual       DesEncrypter->decrypt(String)String, v0, v1
00000034  move-result-object   v2
:36
00000036  return-object        v2
:38
00000038  const-string         v2, ""
0000003C  goto                 :36
.end method
```

FIGURE 8

Source code recovered from the previous figure, which is more easily understood by an analyst.



```
    public String getApiKey() {
        String v1 = this.settings.getString("api_key", "");
        String v2 = !TextUtils.isEmpty(((CharSequence)v1)) ? new DesEncrypter("[redacted]WORLDDOMINATION").decrypt(v1) : "";
        return v2;
    }
```

phones.[6] We downloaded a copy of each of these applications to computers in our lab. Android applications are written in the Java programming language and the code for the applications themselves is not included in a download. However, a wide range of tools exist to take such mobile application code and turn it back into the original source code written by its authors. We performed such *reverse engineering* on each application to learn more about precisely how its creators attempted to provide security.

We use the JEB Decompilation tool by PNF Software. JEB takes an application as input and outputs Java source code. Figure 7 shows application code before this de-compilation process.

We manually read this source code (Figure 8) to identify security-sensitive areas and then determined whether or not the application implemented best practices as commonly understood in the computer science community. Reverse engineering tools such as JEB recover code in applications even if it is never used by the application. Our analysis carefully checked to make sure that such "dead code" was not considered. This process was intensive and time-consuming and drew upon our expertise and many years of experience in this space; many hundreds of hours were spent recovering, understanding and documenting this code.

We were able to recover well over 1 million lines of code from the 26 mobile apps. Our in-depth analyses searched for specific categories of operations, including data encryption and password handling. In the case of encryption, we specifically searched for the use of two outdated and now deprecated cryptographic standards, the "Data Encryption Standard" (DES)[7] and "Triple DES" (3DES).[8] The use of either of these ciphers would void any protections application designers attempted to build to protect the confidentiality of data, as these standards can now be routinely breached.

For password handling, we were particularly interested in the use of "salt," which is random data used to make stored passwords stronger. Salts should be random and unique to each user; if they are not, any protection they may provide is defeated.

Authentication by Mobile Apps

2. Do mobile devices properly verify that they are communicating with the correct server? (Authentication)

Authentication is critical because it ensures that both parties know with whom they are talking. Without strong authentication, anyone can pose as a lender or customer without the other side of the transaction knowing.

Correct certificate handling is critical to authentication on the Internet. If you have ever seen a small lock icon in your browser, it means that the Transport Layer Security (TLS)[9] protocol was protecting your communications. TLS is the result of collaboration between cryptographers, systems security experts, and a broad community of policy experts attempting to provide applications with a straightforward means of protecting their communications. As such, everything from online banking in browsers to email generally relies on TLS.

TLS relies on *certificates* for establishing identity. The certificate itself has a) information about the website/server, b) a public key that can be used by the client to begin communications, and c) a digital signature from a trusted third party (e.g., DigiCert, Verisign). Such certificates form much of the basis of security on the Internet because they can be used to positively identify well-known entities. However, certificates can easily be used in incorrect and dangerous ways. For instance, a client receiving a certificate may fail to check to whom that certificate belongs. An attacker could easily send their own certificate, and such clients would accept it as valid (thereby sending all information directly to the attacker). Similarly, a server could fail to have its certificate signed by a trusted third party, opening the door for an adversary to forge a similar certificate (with no way for the client to determine that such a forgery occurred).

We measured the strength of authentication via certificate handling in mobile applications. Unfortunately, apps do not have that helpful "lock" icon anywhere inside them, so we had to dig deeper. We accomplished this through the use of the Mallodroid tool. Mallodroid takes an application as an input and, like the JEB tool, reverse engineers it to produce Java code. Mallodroid then searches that code for certificate handling routines and determines if they are written in potentially dangerous ways. When the tool indicated a possible problem during our study, we manually analyzed the routine in question and determined if a problem indeed existed and whether that routine was called by the application (and was not dead code). While the use of the Mallodroid tool is relatively fast, it provides limited insight into the code and still requires a significant amount of evaluation by a security engineer.

Server Security

3. Are the servers configured to use strong encryption algorithms to protect the confidentiality and integrity of all communications? (Servers)

Even with close attention to detail for security on the mobile device, failure to provide similar attention in the backend servers can similarly expose sensitive user data. Given that the software engineers writing the code for mobile phones are often different than the individuals configuring the servers, it was critical to check both sides of the connection.

We measured the configuration of the server via the Qualys Secure Sockets Layer (SSL) Test.[10] Qualys provides a free service that attempts to connect to a target server using all possible configurations that have historically been approved for TLS/SSL connections. The output of the Qualys SSL Test is a grade level, similar to those used in traditional report cards. Many previously-allowed versions of these protocols and their parameters, while believed to be secure in the past, are now known to be insecure. Such insecurity can lead to an adversary being able to observe, modify, and inject their own traffic between a user and the server. As such, performing this test helped us concretely characterize the security standing of an organization's communications.

The tool's scoring rubric ranged from A+ to T.[11] We denoted all T scores as F* in our results to make it clear to readers that it is a failing grade. The scores were calculated based on three categories: protocol support, key exchange, and cipher strength.

Supported Protocols

TLS and SSL are cryptographic protocols that secure communication on networks. The original version of SSL (SSL 2.0) was created in 1994, and TLS 1.0 was created in 1999. Since then, the protocols have been updated due to serious security flaws that allowed adversaries to intercept or amend secure communication between two parties—a man-in-the-middle attack. Support for old versions of these protocols is therefore dangerous. The Qualys scoring mechanism (Table 6) weighed each URL's protocol based on the scale in Table 7 below.

TABLE 6

Qualys Score Chart (Derived from "SSL Server Rating Guide")

SCORE	GRADE
Exceptional configurations	A+
Score ≥ 80	A
Score ≥ 65	B
Score ≥ 50	C
Score ≥ 35	D
Score ≥ 20	E
Score < 20	F
Untrusted certificates	T (F*)
Mismatched certificate names	M

TABLE 7

Impact of SSL/TLS Version on Qualys Grade

PROTOCOL	SCORE
SSL 2.0	0%
SSL 3.0	80%
TLS 1.0	90%
TLS 1.1	95%
TLS 1.2	100%

TABLE 8

Key Exchange Characteristics and Their Impact on Qualys Score

KEY EXCHANGE CHARACTERISTIC	SCORE
Weak key (Debian OpenSSL flaw)	0%
Anonymous key exchange (no authentication)	0%
Key or DH parameter strength < 512 bits	20%
Exportable key exchange (limited to 512 bits)	40%
Key or DH parameter strength < 1024 bits (e.g., 512)	40%
Key or DH parameter strength < 2048 bits (e.g., 1024)	80%
Key or DH parameter strength < 4096 bits (e.g., 2048)	90%
Key or DH parameter strength ≥ 4096 bits (e.g., 4096)	100%

TABLE 9

Cipher Strength and Its Impact on Qualys Score

CIPHER STRENGTH	SCORE
0 bits (no encryption)	0%
< 128 bits (e.g., 40, 56)	20%
< 256 bits (e.g., 128, 168)	80%
≥ 256 bits (e.g., 256)	100%

Key Exchange and Cipher Strength

Key exchange occurs when encrypted keys are exchanged between parties, providing a secure means to transmit messages. Legitimate key exchange requires two critical factors: authentication and secure key generation. Authentication is required in order to prevent an unauthorized party from accessing or transmitting messages. In addition, the more random the encryption key, the harder it is for an attacker to crack and steal it. Diffie-Hellman key exchange is an example key exchange mechanism. The Qualys scoring mechanism weighed each URL's key exchange based on the scale detailed in Table 8.

Ciphers are algorithms designed to encrypt and decrypt data. As a general principle, ciphers that use longer keys are a greater barrier to an adversary.[12] The Qualys scoring mechanism weighed each URL's cipher strength based on the scale in Table 9.

Both the protocol support and cipher strength scores were calculated by averaging the score of the strongest and weakest cipher.

While we were able to measure the security of an organization's communications, we make no claims as to whether any specific company has leaked data in the past. We are simply pointing out that the "key is under the doormat" and any attacker capable of seeing the communications could potentially compromise them.

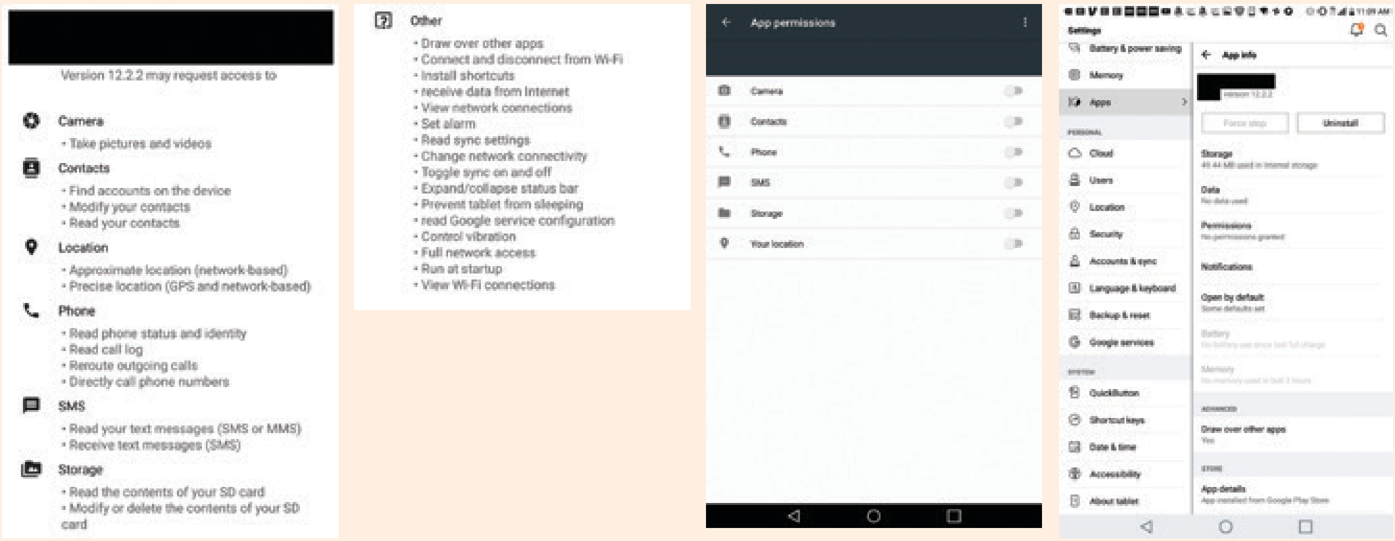
Permissions for Use of Data

4. What data are being collected by digital finance providers over these links? (Permissions)

Mobile applications have access to a significant amount of potentially sensitive data (see Figure 9). Such data includes a mobile user's contact list (and therefore insight into their social network), current location (via GPS data), and potentially much more. It is understood that digital finance providers potentially rely on these kinds of data in order to make their credit decisions; however, they rarely reveal (not even

FIGURE 9

An Example of Permissions Collected by One Digital Finance Provider Mobile App



[redacted]

Version 12.2.2 may request access to

Camera
- Take pictures and videos

Contacts
- Find accounts on the device
- Modify your contacts
- Read your contacts

Location
- Approximate location (network-based)
- Precise location (GPS and network-based)

Phone
- Read phone status and identity
- Read call log
- Reroute outgoing calls
- Directly call phone numbers

SMS
- Read your text messages (SMS or MMS)
- Receive text messages (SMS)

Storage
- Read the contents of your SD card
- Modify or delete the contents of your SD card

Other
- Draw over other apps
- Connect and disconnect from Wi-Fi
- Install shortcuts
- receive data from Internet
- View network connections
- Set alarm
- Read sync settings
- Change network connectivity
- Toggle sync on and off
- Expand/collapse status bar
- Prevent tablet from sleeping
- read Google service configuration
- Control vibration
- Full network access
- Run at startup
- View Wi-Fi connections

App permissions

Camera

Contacts

Phone

SMS

Storage

Your location

App info

[redacted]

version 12.2.2

Storage
49.44 MB used in internal storage

Data
No data used

Permissions
No permissions granted

Notifications

Open by default
Some defaults set

Battery
No battery use since last full charge

Memory
No memory used in last 3 hours

ADVANCED

Draw over other apps
Yes

STORE

App details
App installed from Google Play Store

in privacy policies) those data they actually use or how such data factors into credit decisions.

Our analysis looked at the Android *manifest file*, which is used at the time an application is installed to request access to specific types of sensitive data. This mechanism enables users to be presented with a data list at the time of installation before an application is installed so that they can determine whether or not they wish to grant such access. Accordingly, scanning this file allowed us to characterize the kinds of data these applications seek to collect.

As such data is sent from a mobile device to the server for processing, our analysis could not uncover how each piece of data is used. Such insight would require each company to disclose its algorithms and many claim that these constitute their competitive advantage and intellectual property and are therefore unwilling to share them. In the interest of transparency, credit decision-making data and algorithms must be made available in traditional lending in the United States; regulation, however, has not yet reached the digital lending industry. Regardless, we believe that further study into such algorithms, including their strengths, weaknesses and biases, is a worthwhile task.

Results

Mobile Application Security (Encryption and Authentication)

1. Do mobile devices properly use strong encryption algorithms to protect the confidentiality and integrity of all communications? (Devices)

In order to find weak uses of cryptography, our reverse engineering started by first searching for instances of DES/3DES or static encryption strings. In Table 10 we show our findings in the 27 US and international apps. Most critically, 17 out of 27 apps offered demonstrably dangerous ciphering options. Over half of the apps use DES, an algorithm that has been deprecated for over a decade—well beyond the creation date of these apps. In many of these apps, we also found disastrously incorrect cryptographic use, leading to substantial risk to consumers. In this section, we explore our findings through examples of the problems we found.

As an example of static parameters causing cryptographic problems, in Figure 10 we show where one international company's app generated a secret DES key with a static

TABLE 10

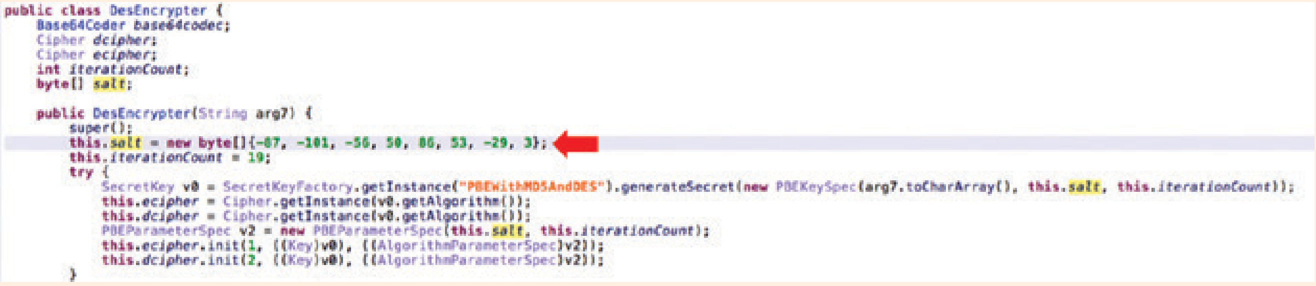
Applications with weak cryptographic parameters. Disturbingly, 17 of 27 applications offered demonstrably bad ciphering options.

APPLICATION	DES	3DES	STATIC VALUES USED FOR ENCRYPTION
Company 1	✓	✓	✓
Company 2	✓	✓	–
Company 3	✓	✓	–
Company 4	✓	✓	–
Company 5	–	–	–
Company 6	✓	✓	✓
Company 7	–	–	–
Company 8	–	–	–
Company 9	–	–	–
Company 10	–	–	✓
Company 11	–	–	–
Company 12	✓	✓	–
Company 13	Obfuscated DES code	–	–
Company 14	–	–	–
Company 15	✓	–	–
Company 16	✓	–	✓
Company 17	–	–	–
Company 18	✓	✓	–
Company 19	✓	✓	–
Company 20	–	–	–
Company 21	✓	✓	✓
Company 22	–	–	–
Company 23	✓	✓	–
Company 24	✓	✓	–
Company 25	✓	✓	–
Company 26	✓	✓	–
Company 27	–	–	–

Note Company names have been anonymized from this table.

FIGURE 10

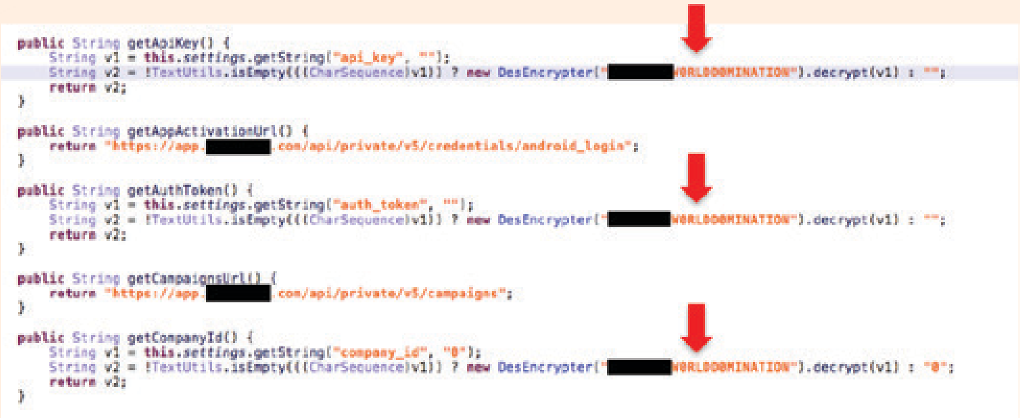
An Example of a Hardcoded "Salt" Value



```
public class DesEncrypter {
    Base64Coder base64codec;
    Cipher dcipher;
    Cipher ecipher;
    int iterationCount;
    byte[] salt;

    public DesEncrypter(String arg7) {
        super();
        this.salt = new byte[]{-87, -101, -56, 50, 86, 53, -29, 3};
        this.iterationCount = 19;
        try {
            SecretKey v0 = SecretKeyFactory.getInstance("PBEWithMD5AndDES").generateSecret(new PBEKeySpec(arg7.toCharArray(), this.salt, this.iterationCount));
            this.ecipher = Cipher.getInstance(v0.getAlgorithm());
            this.dcipher = Cipher.getInstance(v0.getAlgorithm());
            PBEParameterSpec v2 = new PBEParameterSpec(this.salt, this.iterationCount);
            this.ecipher.init(1, ((Key)v0), ((AlgorithmParameterSpec)v2));
            this.dcipher.init(2, ((Key)v0), ((AlgorithmParameterSpec)v2));
    }
```

FIGURE 11

An Example of an App That Used a Static String to Generate Keys



```
public String getApiKey() {
    String v1 = this.settings.getString("api_key", "");
    String v2 = !TextUtils.isEmpty(((CharSequence)v1)) ? new DesEncrypter("[redacted]W0RLDD0MINATION").decrypt(v1) : "";
    return v2;
}

public String getAppActivationUrl() {
    return "https://app.[redacted].com/api/private/v5/credentials/android_login";
}

public String getAuthToken() {
    String v1 = this.settings.getString("auth_token", "");
    String v2 = !TextUtils.isEmpty(((CharSequence)v1)) ? new DesEncrypter("[redacted]W0RLDD0MINATION").decrypt(v1) : "";
    return v2;
}

public String getCampaignsUrl() {
    return "https://app.[redacted].com/api/private/v5/campaigns";
}

public String getCompanyId() {
    String v1 = this.settings.getString("company_id", "0");
    String v2 = !TextUtils.isEmpty(((CharSequence)v1)) ? new DesEncrypter("[redacted]W0RLDD0MINATION").decrypt(v1) : "0";
    return v2;
}
```

salt—that is, one that is always the same. As mentioned previously, salts were designed to add randomness; an adversary who knows this value will have a substantially easier time recovering secret data.

This is not the only place where this weakness occurred in the app, however. Figure 11 is another snapshot of the code found in the app. As denoted by the red arrows, the same string was used as a passphrase to create a secret key. Due to this, all installations of this application would generate the same key, and any adversary with this string could likewise generate the key. Accordingly, communications from this app could be intercepted and decrypted.

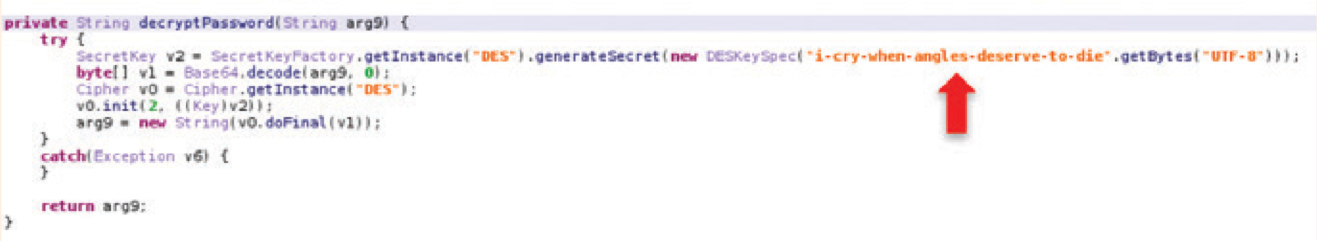
Likewise, Figure 12 is a snapshot of a similar use of cryptography in a US-based lender's app. As shown in the figure, the static string "i-cry-when-angles-deserve-to-die"[13] was used to generate a secret key, causing the same vulnerability as in the previous example.

Another example, an MNO app, contained the same vulnerability, as shown in Figure 13. The arrow points to the *actual* key used in the application. Worse still, we searched the web for the list of numbers comprising this key and found them in an unrelated, open-source project.[14] The secrecy of this key (and, as a result, the security of this cryptosystem) is completely non-existent. Not only has the company failed to produce a random value for key generation, the key itself is both static and hardcoded. *No additional work is required for an adversary to use this key.* For other uses, Figure 14 shows the name of the company being used to encrypt and decrypt. This application represents the most egregious misuse of cryptography we examined among all applications.

In a different international example, the application specifically allowed both DES and 3DES, as shown in Figure 15. Allowing these deprecated algorithms could allow an adversary to recover sensitive communications in what is known as a "downgrade attack." Such attacks are simple to configure and execute while the victim is unaware that their data is being compromised. More critically, all of the above options were demonstrably weak—they allowed both null MD5 and null SHA1,[15] neither of which provide encryption. All eight of the ciphers have been deemed broken and hence should not be used for encryption.

FIGURE 12

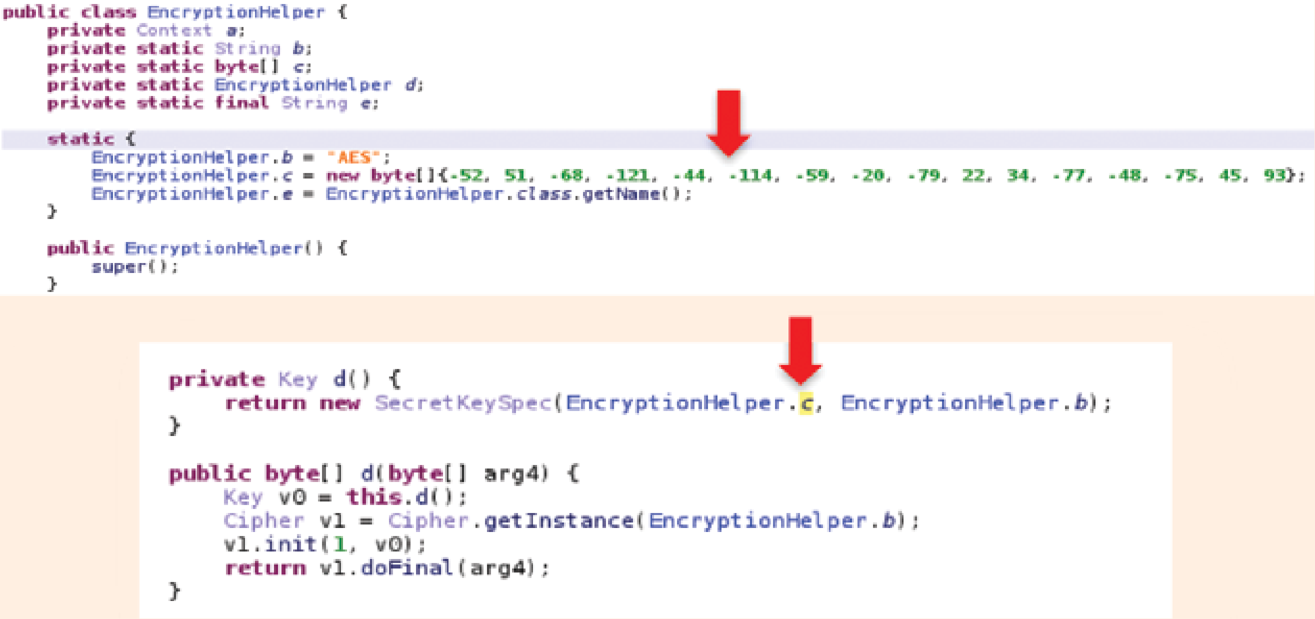
This Company Also Used a Static String to Generate Its Cryptographic Key



```
private String decryptPassword(String arg9) {
    try {
        SecretKey v2 = SecretKeyFactory.getInstance("DES").generateSecret(new DESKeySpec("i-cry-when-angles-deserve-to-die".getBytes("UTF-8")));
        byte[] v1 = Base64.decode(arg9, 0);
        Cipher v0 = Cipher.getInstance("DES");
        v0.init(2, ((Key)v2));
        arg9 = new String(v0.doFinal(v1));
    }
    catch(Exception v6) {
    }

    return arg9;
}
```

FIGURE 13

An MNO App Used a Static Cryptographic Key Across All Users

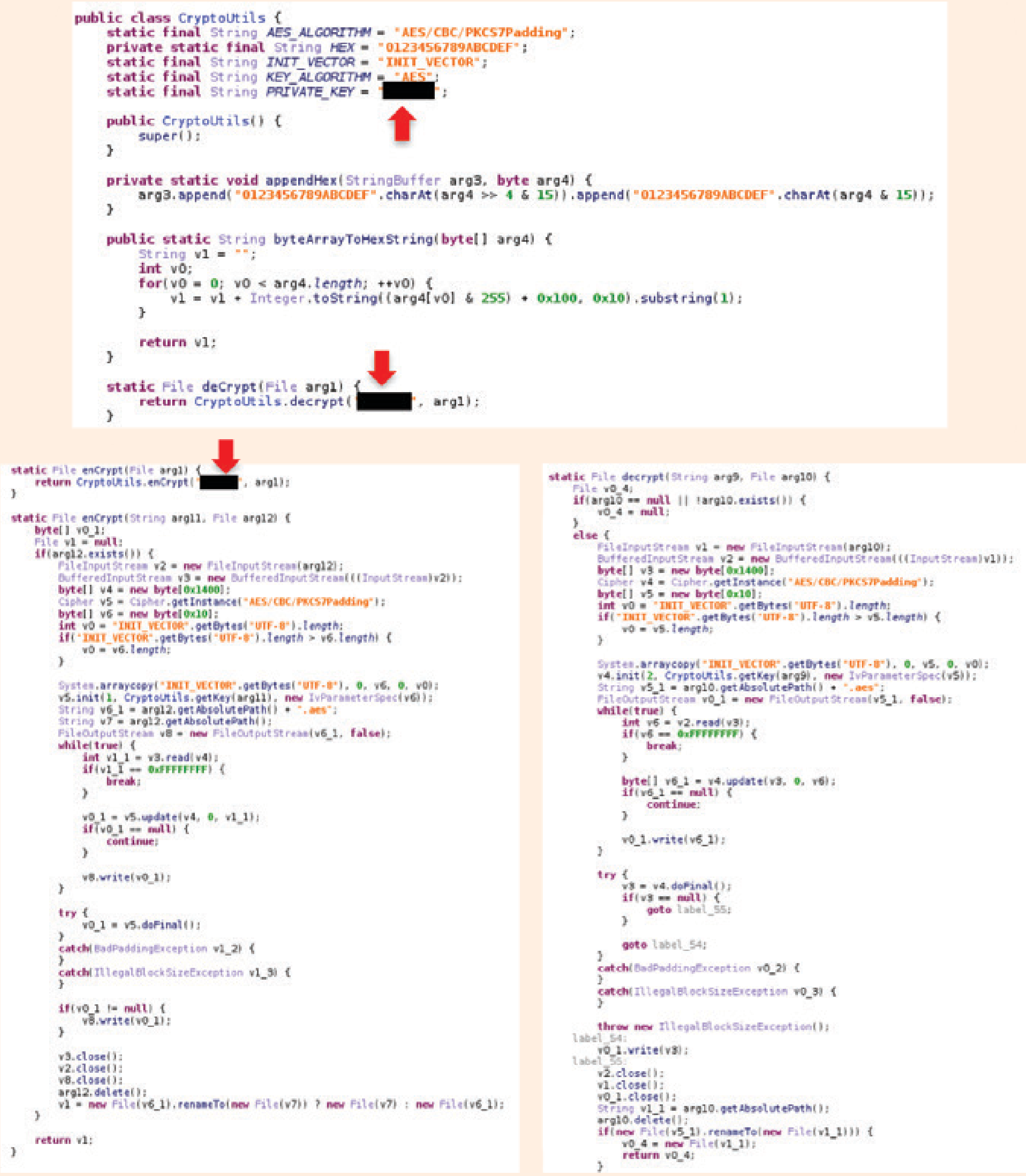


```
public class EncryptionHelper {
    private Context a;
    private static String b;
    private static byte[] c;
    private static EncryptionHelper d;
    private static final String e;

    static {
        EncryptionHelper.b = "AES";
        EncryptionHelper.c = new byte[]{-52, 51, -68, -121, -44, -114, -59, -20, -79, 22, 34, -77, -48, -75, 45, 93};
        EncryptionHelper.e = EncryptionHelper.class.getName();
    }

    public EncryptionHelper() {
        super();
    }
```

```
private Key d() {
    return new SecretKeySpec(EncryptionHelper.c, EncryptionHelper.b);
}

public byte[] d(byte[] arg4) {
    Key v0 = this.d();
    Cipher v1 = Cipher.getInstance(EncryptionHelper.b);
    v1.init(1, v0);
    return v1.doFinal(arg4);
}
```

FIGURE 14

This Company Used a Static Key to Encrypt and Decrypt



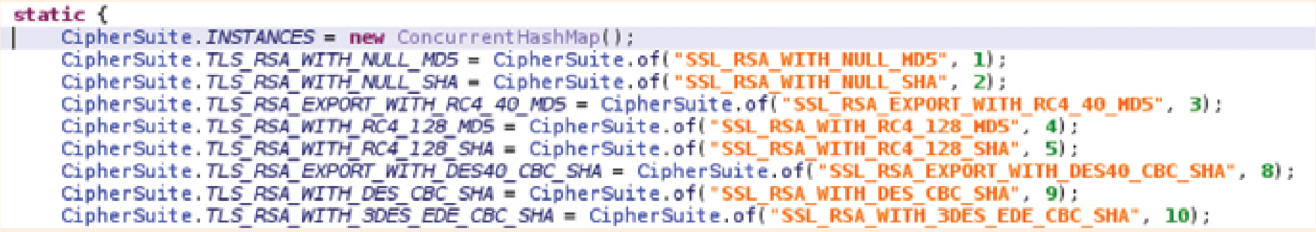
```
public class CryptoUtils {
    static final String AES_ALGORITHM = "AES/CBC/PKCS7Padding";
    private static final String HEX = "0123456789ABCDEF";
    static final String INIT_VECTOR = "INIT_VECTOR";
    static final String KEY_ALGORITHM = "AES";
    static final String PRIVATE_KEY = "[redacted]";

    public CryptoUtils() {
        super();
    }

    private static void appendHex(StringBuffer arg3, byte arg4) {
        arg3.append("0123456789ABCDEF".charAt(arg4 >> 4 & 15)).append("0123456789ABCDEF".charAt(arg4 & 15));
    }

    public static String byteArrayToHexString(byte[] arg4) {
        String v1 = "";
        int v0;
        for(v0 = 0; v0 < arg4.length; ++v0) {
            v1 = v1 + Integer.toString((arg4[v0] & 255) + 0x100, 0x10).substring(1);
        }

        return v1;
    }

    static File deCrypt(File arg1) {
        return CryptoUtils.decrypt("[redacted]", arg1);
    }
```

```
static File enCrypt(File arg1) {
    return CryptoUtils.enCrypt("[redacted]", arg1);
}

static File enCrypt(String arg11, File arg12) {
    byte[] v0_1;
    File v1 = null;
    if(arg12.exists()) {
        FileInputStream v2 = new FileInputStream(arg12);
        BufferedInputStream v3 = new BufferedInputStream(((InputStream)v2));
        byte[] v4 = new byte[0x1400];
        Cipher v5 = Cipher.getInstance("AES/CBC/PKCS7Padding");
        byte[] v6 = new byte[0x10];
        int v0 = "INIT_VECTOR".getBytes("UTF-8").length;
        if("INIT_VECTOR".getBytes("UTF-8").length > v6.length) {
            v0 = v6.length;
        }

        System.arraycopy("INIT_VECTOR".getBytes("UTF-8"), 0, v6, 0, v0);
        v5.init(1, CryptoUtils.getKey(arg11), new IvParameterSpec(v6));
        String v6_1 = arg12.getAbsolutePath() + ".aes";
        String v7 = arg12.getAbsolutePath();
        FileOutputStream v8 = new FileOutputStream(v6_1, false);
        while(true) {
            int v1_1 = v3.read(v4);
            if(v1_1 == 0xFFFFFFFF) {
                break;
            }

            v0_1 = v5.update(v4, 0, v1_1);
            if(v0_1 == null) {
                continue;
            }

            v8.write(v0_1);
        }

        try {
            v0_1 = v5.doFinal();
        }
        catch(BadPaddingException v1_2) {
        }
        catch(IllegalBlockSizeException v1_3) {
        }

        if(v0_1 != null) {
            v8.write(v0_1);
        }

        v3.close();
        v2.close();
        v8.close();
        arg12.delete();
        v1 = new File(v6_1).renameTo(new File(v7)) ? new File(v7) : new File(v6_1);
    }

    return v1;
}
```

```
static File decrypt(String arg9, File arg10) {
    File v0_4;
    if(arg10 == null || !arg10.exists()) {
        v0_4 = null;
    }
    else {
        FileInputStream v1 = new FileInputStream(arg10);
        BufferedInputStream v2 = new BufferedInputStream(((InputStream)v1));
        byte[] v3 = new byte[0x1400];
        Cipher v4 = Cipher.getInstance("AES/CBC/PKCS7Padding");
        byte[] v5 = new byte[0x10];
        int v0 = "INIT_VECTOR".getBytes("UTF-8").length;
        if("INIT_VECTOR".getBytes("UTF-8").length > v5.length) {
            v0 = v5.length;
        }

        System.arraycopy("INIT_VECTOR".getBytes("UTF-8"), 0, v5, 0, v0);
        v4.init(2, CryptoUtils.getKey(arg9), new IvParameterSpec(v5));
        String v5_1 = arg10.getAbsolutePath() + ".aes";
        FileOutputStream v0_1 = new FileOutputStream(v5_1, false);
        while(true) {
            int v6 = v2.read(v3);
            if(v6 == 0xFFFFFFFF) {
                break;
            }

            byte[] v6_1 = v4.update(v3, 0, v6);
            if(v6_1 == null) {
                continue;
            }

            v0_1.write(v6_1);
        }

        try {
            v3 = v4.doFinal();
            if(v3 == null) {
                goto label_55;
            }

            goto label_54;
        }
        catch(BadPaddingException v0_2) {
        }
        catch(IllegalBlockSizeException v0_3) {
        }

        throw new IllegalBlockSizeException();
    label_54:
        v0_1.write(v3);
    label_55:
        v2.close();
        v1.close();
        v0_1.close();
        String v1_1 = arg10.getAbsolutePath();
        arg10.delete();
        if(new File(v5_1).renameTo(new File(v1_1))) {
            v0_4 = new File(v1_1);
            return v0_4;
        }
```

FIGURE 15

An App That Allowed for the Use of Export DES, DES, and 3DES, All of Which Are Considered Weak Ciphers



```
static {
    CipherSuite.INSTANCES = new ConcurrentHashMap();
    CipherSuite.TLS_RSA_WITH_NULL_MD5 = CipherSuite.of("SSL_RSA_WITH_NULL_MD5", 1);
    CipherSuite.TLS_RSA_WITH_NULL_SHA = CipherSuite.of("SSL_RSA_WITH_NULL_SHA", 2);
    CipherSuite.TLS_RSA_EXPORT_WITH_RC4_40_MD5 = CipherSuite.of("SSL_RSA_EXPORT_WITH_RC4_40_MD5", 3);
    CipherSuite.TLS_RSA_WITH_RC4_128_MD5 = CipherSuite.of("SSL_RSA_WITH_RC4_128_MD5", 4);
    CipherSuite.TLS_RSA_WITH_RC4_128_SHA = CipherSuite.of("SSL_RSA_WITH_RC4_128_SHA", 5);
    CipherSuite.TLS_RSA_EXPORT_WITH_DES40_CBC_SHA = CipherSuite.of("SSL_RSA_EXPORT_WITH_DES40_CBC_SHA", 8);
    CipherSuite.TLS_RSA_WITH_DES_CBC_SHA = CipherSuite.of("SSL_RSA_WITH_DES_CBC_SHA", 9);
    CipherSuite.TLS_RSA_WITH_3DES_EDE_CBC_SHA = CipherSuite.of("SSL_RSA_WITH_3DES_EDE_CBC_SHA", 10);
```

TABLE 11

Mallodroid found four applications that failed to properly handle certificates.

APPLICATION	BROKEN TRUST MANAGER	BROKEN HOST NAME VERIFIER
Company 1	✓	✓
Company 2	✓	✓
Company 3	✓	✓
Company 4	No Error Reported	✓

Note Company names have been anonymized from this table.

Authentication

2. Do mobile devices properly verify that they are communicating with the correct server? (Authentication)

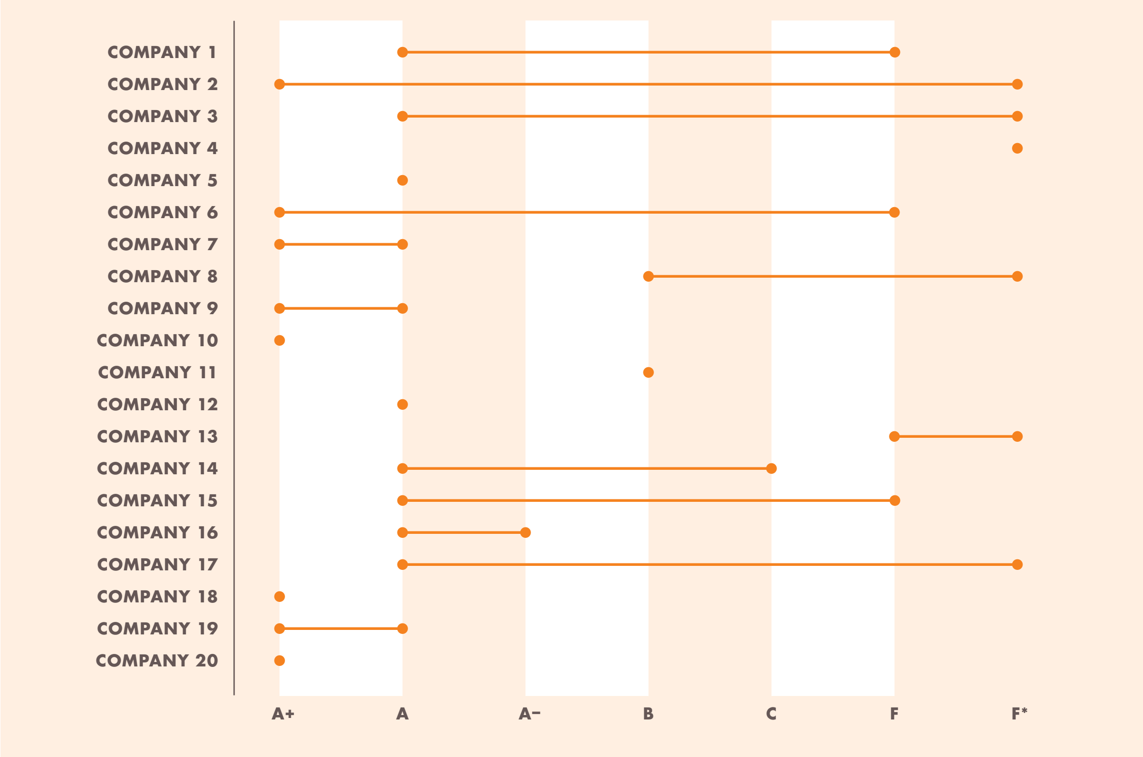
We ran Mallodroid on each of the 27 Android apps. Mallodroid detects broken trust managers and host name verifiers. These functions are critically important to ensuring the security of a TLS session but, when broken, often defeat security in subtle but catastrophic ways.

Trust managers accept or reject presented credentials and manage trust material in order to make trust decisions. Host Name Verifiers determine if a URL's hostname matches its respective server's hostname. If they do not, the verifier takes necessary steps to determine if the connection should be allowed. As shown in Table 11, three of the apps had broken trust managers and four had broken host name verifiers. This means that 3 of the reviewed companies had apps that failed to make trustworthy decisions, and 4 of them failed to verify if connections should actually be allowed.

The consequences of these broken functions are severe; adversaries that can impersonate the provider's service to the app (e.g., on a coffee shop Wi-Fi network) can intercept the sensitive communications and record and tamper with messages between the app and the provider's legitimate service. These apps perform no or weak verification of the service and in many cases will blindly accept a connection to any server that answers. Without this verification, the user is never notified of a problem and the app appears to work normally.

FIGURE 16

Best and Worst Qualys Scores



Note All valid links found in the apps. Companies in this figure have been anonymized.

Server Configuration Security

3. Are the servers configured to use strong encryption algorithms to protect the confidentiality and integrity of all communications? (Servers)

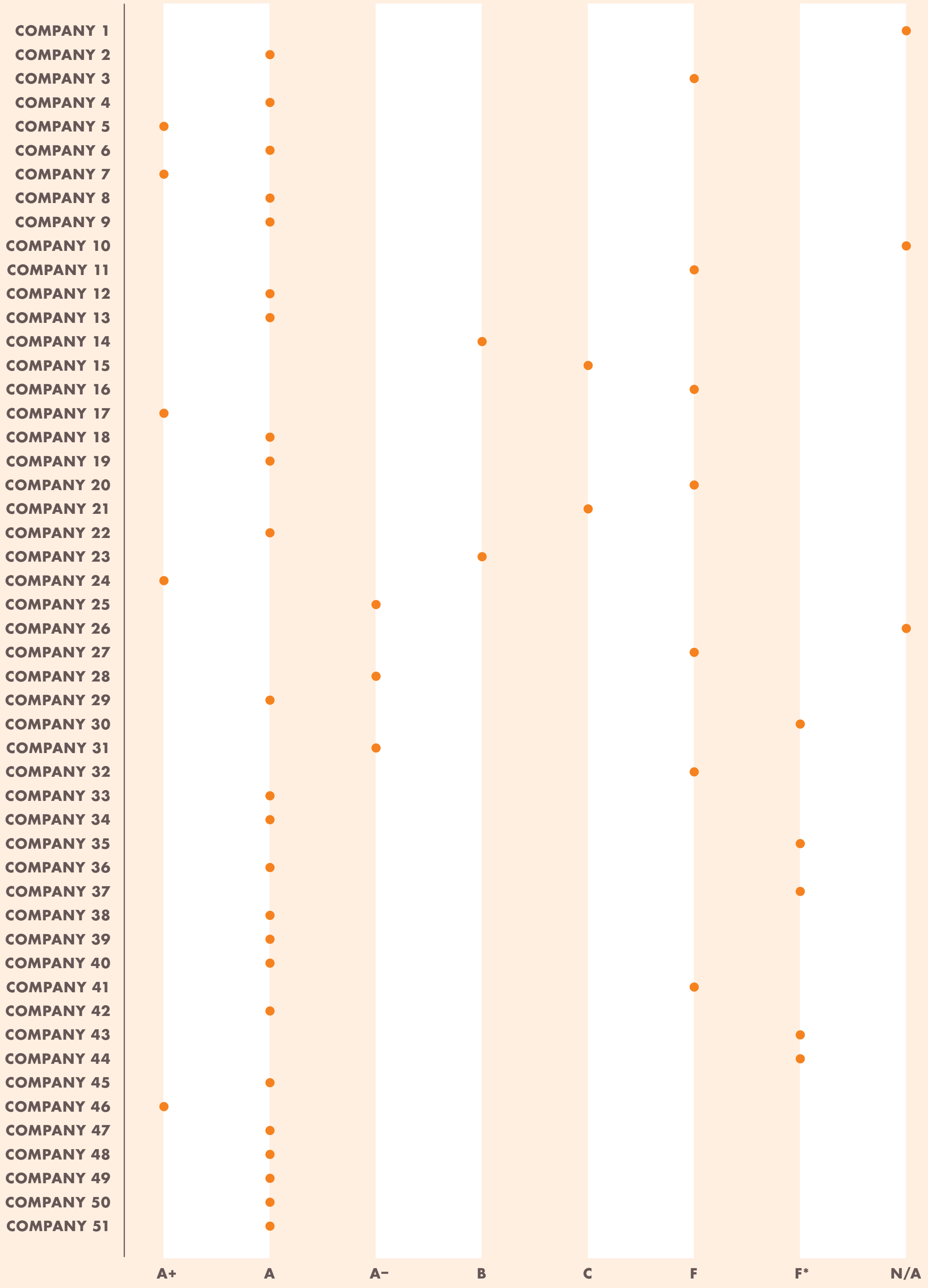
Apps often contact many different servers. Each server may be configured differently, so we first extracted all of the URLs and other server information from each app. We then characterized the range of these configurations using the Qualys SSL scanner, as described above.

Figure 16 shows the best and worst Qualys score of all URLs found in the assessed apps. Coins showed the widest range of configuration failures, from A+ to F*. The lack of consistency among providers was particularly concerning as it signaled that the provider might not have had a configuration management process, which may prevent (or alert engineers to) errors such as weak Diffie-Hellman key exchange parameters, acceptance of deprecated algorithms, and certificate-not-trusted errors.

We also evaluated the international company web sites from our original list of 52 companies for TLS/SSL configuration errors. As shown in Figure 17, a much higher percentage of company websites had properly configured TLS/SSL connections. However, there were also a number of important negative observations. First, 12 of the websites had demonstrably vulnerable configurations (e.g., used broken encryption ciphers, were susceptible to interception, etc.).

FIGURE 17

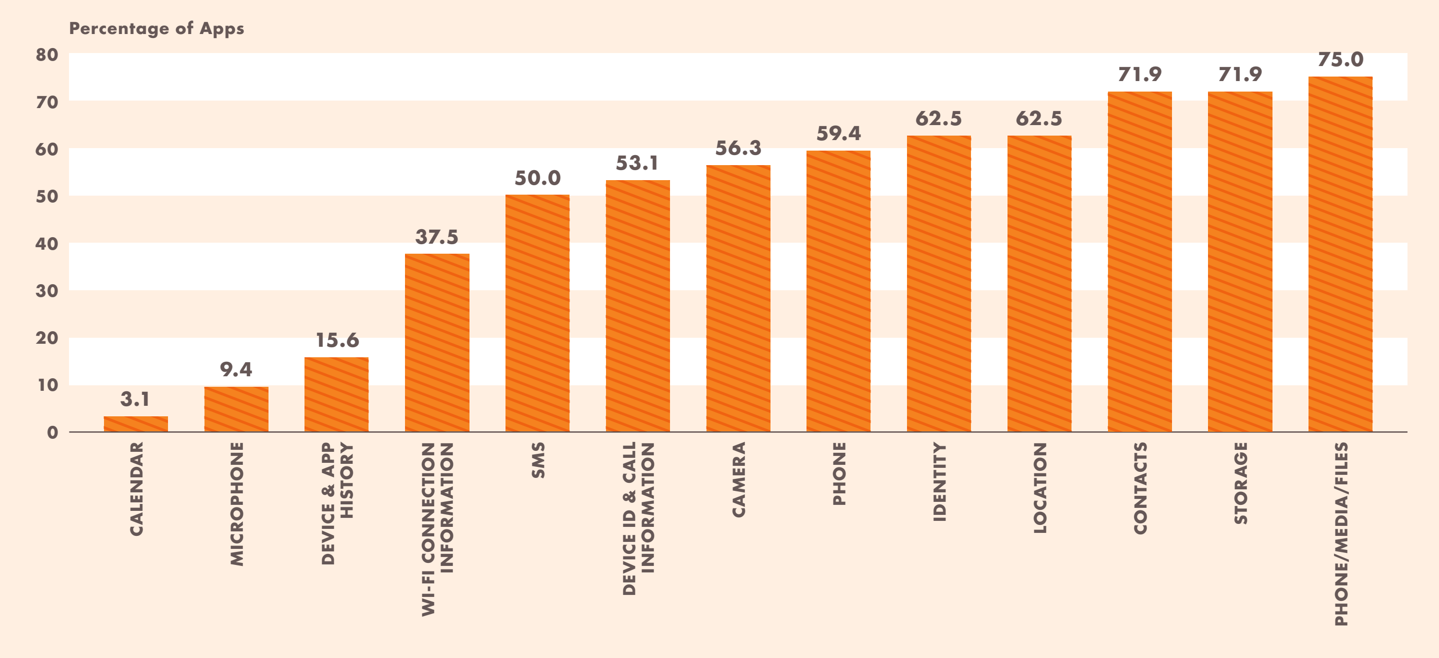
Qualys Test Results for Digital Finance Provider Websites; 11 of 53 Websites Received a Failing Grade



Note Companies in this figure have been anonymized.

FIGURE 18

Permissions Requested by Digital Finance Applications



Permissions

4. What data are being collected by digital finance providers over these links? (Permissions)

We examined the permissions required by apps. Figure 18 shows the frequency of each permission type. Most apps required access to the device's identity, location, phone call capability, photos/media/files, storage, and device ID and call information. Though such permissions may be requested for legitimate reasons (e.g., phone call capability for automatically dialing customer service), all of these could be misused to leak private information about the customer.

Requests for a number of other permissions were more concerning. For instance, 72 percent of the apps requested access to a user's contacts. It was unclear how contacts access would be used on the device; we found nothing in their privacy policies outlining the apps' use of this access. Further study of the apps' code may reveal why this access is required and how the data is stored and processed. Additionally, the reasons for access to "microphone" (9.4 percent), "device and app history"[16] (15.6 percent) and the camera (56.3 percent) were neither discussed in the privacy policy nor obvious in their intentions.

Conclusions

Overall, we found numerous egregious security errors in over half of the apps we examined, including misuse of cryptography, use of weak cryptography, and excessive permission requirements. Many of these issues have been known by both the academic community and industry for years, and these problems put both consumers and providers at severe risk of compromise. As we will show in the next section, these problems can become even worse for the consumer when the provider absolves itself of risk, forcing the consumer to accept the risk of the apps' flaws.

3

Terms of Service Analysis

We reviewed each app's terms of service document to examine how companies were distributing liability (for the consequences of kinds of security breaches described here) between themselves and users. Our intention was to identify how the providers' terms of service described the user liability in case of compromise. Our codebook for this analysis was: "liable, liability, harmless, responsible, fraud," and we followed the same coding process discussed previously in our privacy analysis. Our results are shown in Table 12.

Only eight terms of service agreements addressed fraud by the user and only eight addressed fraud against the user. Those that did address fraud against the user noted that the provider is not liable. This places users in an untenable position. To use an app—many of which we have shown to be vulnerable—they must accept these terms. If a fraud occurs, the user must bear the consequences. Such terms can harm consumer trust of these apps and may ultimately constrain the expansion of these critical services.

TABLE 12

Terms of Service: Fraud

COMPANIES WITH APPS	MENTIONS FRAUD BY USER	MENTIONS FRAUD AGAINST USER
Company 1	–	–
Company 2	–	–
Company 3	–	–
Company 4	✓	"exclusion of liability"
Company 5	✓	–
Company 6	✓	–
Company 7	✓	"keep the Bank and Finserve free and harmless"
Company 8	–	"Limitation of Liability exception"
Company 9	✓	"will not be liable for"
Company 10	–	–
Company 11	–	–
Company 12	✓	"fully assume the risks of liability"
Company 13	–	"no manner responsible for any claim"
Company 14	–	–
Company 15	–	"exclusion of liability"
Company 16	✓	–
Company 17	–	–
Company 18	–	–
Company 19	–	–
Company 20	–	–
Company 21	–	–
Company 22	–	–
Company 23	✓	"exclusion of liability"
Company 24	–	–
Company 25	–	–
Company 26	–	–
Company 27	–	–
Company 28	–	–
Company 29	–	–
Company 30	–	–
Company 31	–	–
Company 32	–	–
Company 33	–	–

Note Company names in this table have been anonymized.

4 Conclusions and Recommendations

We believe in the potentially transformative power of digital finance. These powerful financial instruments, their providers, and their users deserve the best possible protections.

Credit can be a tremendous tool to empower consumers or help build businesses. For many people, digital finance systems make it possible to access credit for the first time, while for many others they increase the convenience and flexibility of the credit they can use. As with any powerful tool—a chainsaw, an automobile, a medicine—it is essential for users to be confident that it will perform safely when used wisely. When the security of a credit tool can be compromised, however, consumers are exposed to potentially very serious risks; and these risks could also affect the viability of the company providing the service.

Digital finance often relies on access to non-traditional data sources, especially data held on mobile devices. As such, it is critical to determine how data is treated and protected in such systems, and to measure the policy and infrastructure that is deployed to minimize potential harm. In a world where even large and seemingly well-protected financial companies are being breached, such protections have never been more critical.

Privacy Policy Recommendations

Our analysis of the privacy and security of digital finance providers characterized the areas of coverage offered by each policy, followed by an analysis of readability. In general, we discovered that the policies of internationally-based digital finance providers were longer and more difficult to read. Virtually all were less understandable than the privacy policies of traditional US-based financial institutions. Moreover, as our technical analysis confirmed, none of the policies provide specifics regarding the types of data that will be collected or how it will be handled. Given that many of these applications request access to information such as GPS location, contact lists, cameras, microphones, and calendars, we believe that these companies must significantly improve their actions in regards to sensitive user data, starting with improving the quality of privacy policies.

All players in the industry would do well to adopt either the FDIC or GSMA recommendations—or indeed, both. Policies should also explicitly identify the sensitive user data they intend to collect so that users can make informed decisions when selecting a lender. Beyond this, the industry should strive for further clarity in the creation of these policies, including writing policies that match the reading level and language of their intended markets. Conformance to this suggestion could easily be tested using the tools we applied in our evaluation. Adoption of this recommendation would not only provide greater clarity for users, but would also make digital lenders better at communicating privacy policies than traditional banks, furthering their value proposition as a more convenient and beneficial alternative.

Technical Security Recommendations

Our security analysis focused on security of applications running on mobile devices, proper authentication, and proper configuration of servers. We found widespread misuse of cryptographic algorithms, including the use of algorithms that have long been publicly deprecated (i.e., listed as beyond their useful lifetime and therefore inherently dangerous). We discovered that 17 of the 27 apps offered such dangerous options. Our results for authentication were better, but four of the 27 apps still exhibited significant vulnerabilities that would allow an adversary to impersonate the server and potentially trick clients into exposing their data. Finally, we measured server TLS/SSL configuration and demonstrated a wide range of security (even among different servers involved in a single app). At least eight of the apps had entirely vulnerable configurations that represent a significant threat to their users. Poor security configuration also impacted digital finance provider websites, with approximately one-quarter also receiving a failing grade.

At the time of this report's publication, the strongest possible relevant standard for secure communications is TLS 1.2. Digital finance providers should eliminate the use of all other versions as soon as possible. Factors that may slow down such changes include legacy devices; however, a measurement study of TLS versions available to users would easily allow for a reasonable timeline to be developed. The use of TLS alone is not enough—it must be correctly parameterized. Applications should use strong encryption ciphers and hashing algorithms (e.g., "AES_256_CBC_SHA256"). Such algorithms can be efficiently implemented in both client and server with negligible impact to performance. Support for weak ciphers (e.g., DES, 3DES) and weak hashing algorithms (e.g., MD5) must be eliminated immediately from the servers of all digital finance providers, as they provide a false sense of security. Such configuration should be uniformly applied across servers and mobile devices. Finally, digital financial applications should not rely on third-party advertising or metrics libraries, some of which—our analysis shows—are guilty of poor information security practices. All such functionality should be developed and protected by the digital lenders themselves.

Terms of Service Recommendations

Finally, we explored the terms of service associated with each app. Unfortunately, only a small number of these providers had available terms of service. Financial institutions in the US have a legally-mandated maximum value of fraud for which consumers are liable (e.g., US $50 for credit cards). Similar standards should be enacted for the digital finance space. While challenging, due to the wide range of jurisdictions involved, such strong consumer protection laws have the positive side-effect of encouraging financial companies to invest in protecting consumers so as to reduce their own losses to fraud.

In an effort to promote greater transparency and collaboration within the sector, CFI (with the collaboration of the Venture Lab team within Accion) carried out a consultative process for all the companies included in the sample for this research project. An effort to reach all 53 companies via email was made, and only one company could not be contacted because no contact information for that company was ever found. The remaining 52 institutions were all sent a brief report containing their individual scores for each test, together with a document that briefly described what each test entailed and a letter of invitation to participate in a private discussion with the research team in order to answer any questions they might have about the findings of the study or any next steps. We also shared with them a guidebook that was developed by the research team to help start-up fintech companies solve many of the most common data security vulnerabilities uncovered by this study. Out of the 52 companies that were contacted, only seven companies responded and participated in this discussion. A private link with the recording of this session was later shared with all companies via email.

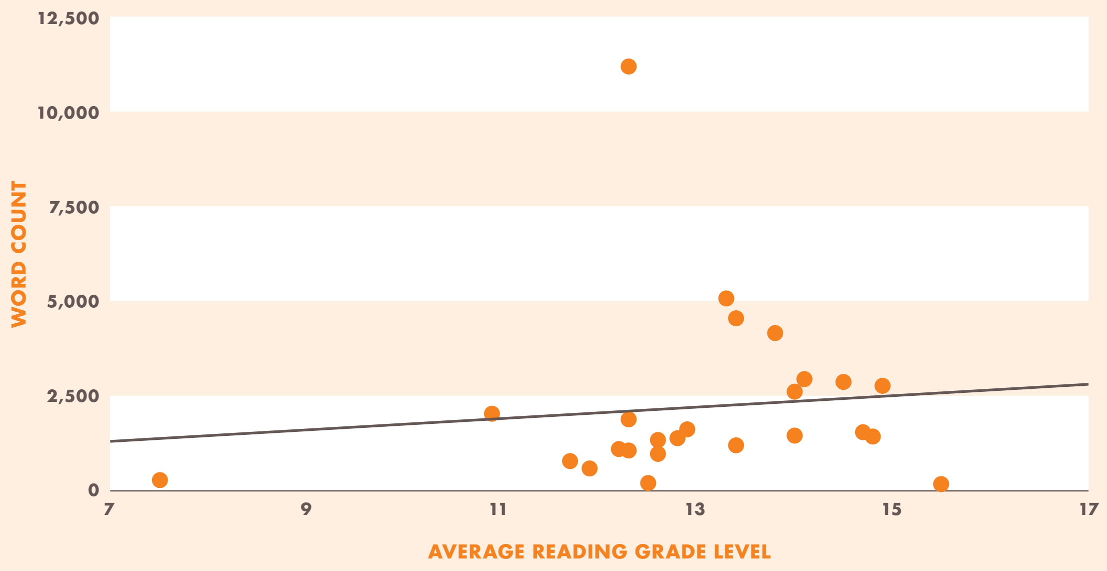
We believe in the potentially transformative power of digital finance. However, we also believe that these powerful financial instruments, their providers, and their users deserve the best possible protections. Our measurement study demonstrated that this industry has significant room for improvement in the availability, accessibility and clarity of privacy policies and in the correct use of security mechanisms in both mobile devices and backend servers.

> It is critical to determine how data is treated and protected in digital finance models, and to measure the policy and infrastructure that is deployed to minimize potential harm.

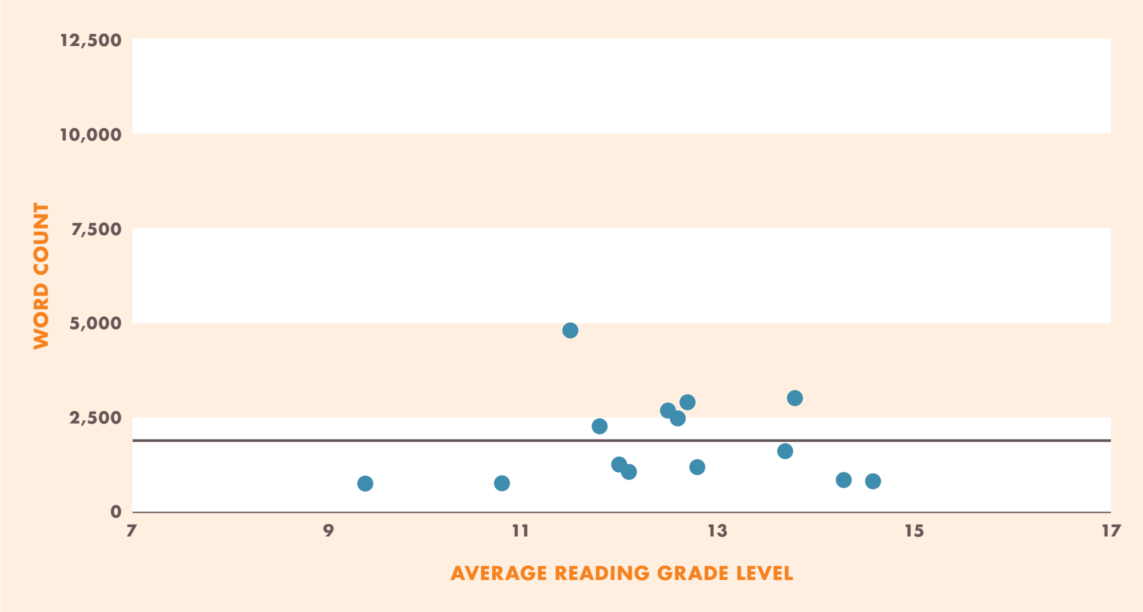
ANNEX A

Word Count vs. Average Reading Grade Level of Privacy Policies

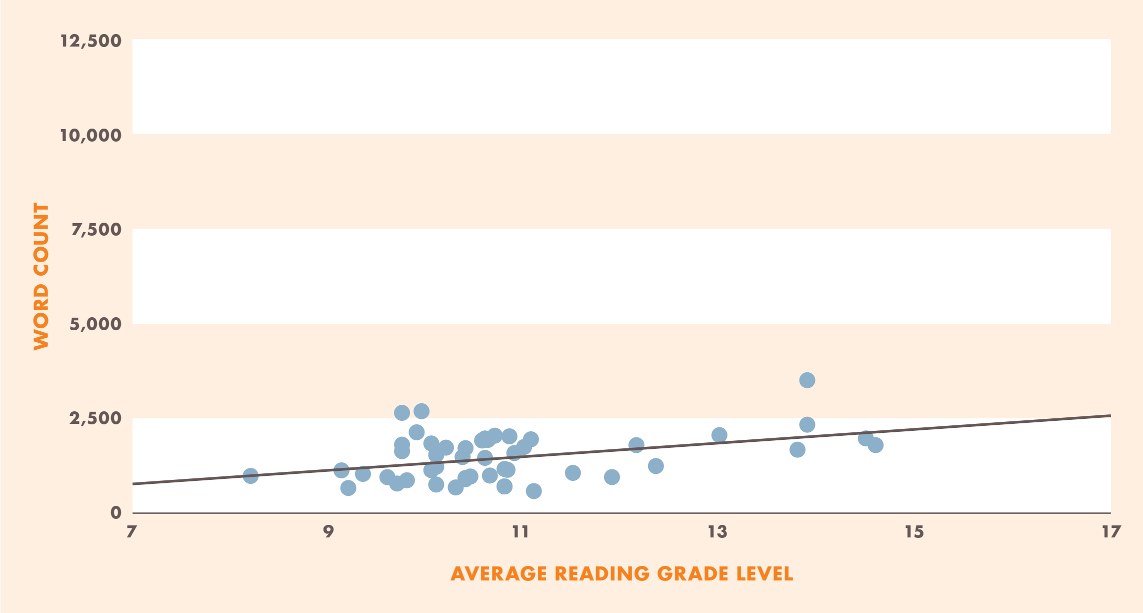
International Digital Lenders' Privacy Policies



US Digital Lenders’ Privacy Policies



US Traditional Banks



ANNEX B

Digital Lenders Evaluated and Analyses Performed

	PRIVACY POLICY ANALYSIS	APP SECURITY ANALYSIS	QUALYS ANALYSIS
Airtel	✓	✓	✓
Atom	✓	–	✓
Azimo	✓	✓	✓
BlueVine	✓	–	✓
Branch	✓	✓	✓
C2FO	✓	–	✓
Coins	✓	✓	✓
CommonBond	✓	–	✓
Creditas	✓	–	✓
CrowdEstates	✓	–	✓
EcoCash Loans	–	✓	✓
Equitel	–	✓	✓
Equity Direct Mobile	–	✓	✓
Farm Drive	–	✓	✓
FastPay	✓	–	✓
GetBucks	✓	✓	✓
IndiaMART	✓	✓	✓
Insikt	✓	–	✓
InstaPaisa	✓	–	✓
InvoiNet	✓	✓	✓
Jimubox	✓	–	✓
Kabbage	✓	✓	✓
KCB	✓	✓	✓
Kiva	✓	✓	✓
Koopkrag	–	✓	✓
Kopo Kopo	✓	✓	✓

	PRIVACY POLICY ANALYSIS	APP SECURITY ANALYSIS	QUALYS ANALYSIS
Kubo Financiero	✓	–	✓
Lending Club	✓	–	✓
Lulalend	✓	–	✓
M-Pawa	–	✓	✓
MCo-op Cash	–	✓	✓
Micromobile	–	–	✓
MoneyTap	✓	–	✓
OnDeck	✓	–	✓
Pay Your Tuition Funds	✓	–	✓
PaySense	✓	✓	✓
PesaPata	–	✓	✓
Prosper	✓	✓	✓
Puddle	✓	–	✓
RoadLoans	✓	✓	✓
Saida	–	–	✓
Salud Fácil	–	–	–
SMECorner	✓	–	✓
Social Lender	✓	–	✓
Suregifts Redemption	–	✓	✓
Tala	–	–	✓
Taplend	–	✓	✓
Tencent	✓	✓	✓
Upstart	✓	–	✓
WeFinance	✓	–	✓
Yoco	✓	✓	✓
Zidisha	✓	✓	✓

Notes

[1] We excluded Salud Fácil of Mexico from this portion of the study because its site is solely in Spanish. To be relevant for users, local language versions would be needed but, overall, we found few policies available in such languages.

[2] Federal Deposit Insurance Corporation. Privacy Rule Handbook. Accessed through the FDIC web site at https://www.fdic.gov/regulations/examinations/financialprivacy/handbook/index.html.

[3] GSMA (The GSM Association). Mobile Privacy Principles: Promoting Consumer Privacy in the Mobile Ecosystem. Accessed through the GSMA web site at: https://www.gsma.com/publicpolicy/wp-content/uploads/2016/02/GSMA2016_Guidelines_Mobile_Privacy_Principles.pdf.

[4] Bowers, Jasmine, Bradley Reaves, Imani N. Sherman, Patrick Traynor, and Kevin Butler. 2017. *Regulators, Mount Up! Analysis of Privacy Policies for Mobile Money Applications*, In "Proceedings of the USENIX Symposium on Usable Privacy and Security (SOUPS)." Accessible online at: https://www.usenix.org/system/files/conference/soups2017/soups2017-bowers.pdf.

[5] Central Intelligence Agency. "Field listing: School life expectancy (primary to tertiary education)." Accessed at https://www.cia.gov/library/publications/the-world-factbook/fields/2205.html#uk.

[6] The companies not analyzed in this section operate through webpages, rather than mobile apps. Therefore, we were unable to examine their operations without explicit access to codes from the company.

[7] National Institute of Standards and Technology (NIST). US Department of Commerce. June 2, 2005. "NIST withdraws outdated data encryption standard." Accessed online at: https://www.nist.gov/news-events/news/2005/06/nist-withdraws-outdated-data-encryption-standard.

[8] National Institute of Standards and Technology (NIST). US Department of Commerce. July 11, 2017. "Update to current use and deprecation of TDEA." Accessed online at: https://csrc.nist.gov/news/2017/update-to-current-use-and-deprecation-of-tdea.

[9] An alternative standard, the Secure Sockets Layer (SSL), was used prior to TLS. SSL/TLS are often used interchangeably when discussing secure communications. We will use the correct term where appropriate.

[10] Qualys, Inc. "SSL Server Test" accessed online at https://www.ssllabs.com/ssltest/.

[11] Qualys, Inc. May 8, 2017. "SSL Server Rating Guide." Accessed online at: https://github.com/ssllabs/research/wiki/SSL-Server-Rating-Guide.

[12] This is a nuanced issue. Comparing two unrelated ciphers is difficult and the protection provided by a specific key length in one may not be equal to the protection provided by the same key length in the other. However, general practice for publicly-vetted symmetric encryption ciphers is to use keys of at least 128 bits, with 256 bits being the currently recommended standard.

[13] Misspelled lyrics to the song "Chop Suey!" by the rock band System of a Down.

[14] GitHub, Inc. 2018. "juspay ec-android-demo." Accessed online at: https://github.com/juspay/ec-android-demo/blob/master/expresscheckout/build.gradle

[15] MD5 and SHA1 are standardized hashing algorithms, not standardized encryption algorithms. Hashing algorithms are used to make concise representations of content and are crucial to ensuring the integrity (as opposed to confidentiality, which is provided by encryption) of communications.

[16] This permission can allow an application to gain access to sensitive logs, learn the identity of the other applications running on the phone and read web bookmarks.

The Center for Financial Inclusion at Accion (CFI) is an action-oriented think tank that engages and challenges the industry to better serve, protect, and empower clients. We develop insights, advocate on behalf of clients, and collaborate with stakeholders to achieve a comprehensive vision for financial inclusion. We are dedicated to enabling 3 billion people who are left out of—or poorly served by—the financial sector to improve their lives.

www.centerforfinancialinclusion.org

www.cfi-blog.org

@CFI_Accion

CENTER *for*
FINANCIAL
INCLUSION | ACCION