CoRA: Collaborative Risk-Aware Authentication

Mastooreh Salajegheh, Shashank Agrawal, Maliheh Shirvanian,
Mihai Christodorescu, Payman Mohassel

Visa Research

Abstract

Today, authentication faces the trade-off of security versus usability. Two factor authentication, for example, is one way to improve security at the cost of requiring user interaction for every round of authentication. Most 2FA methods are bound to user's phone and fail if the phone is not available. We propose CoRA, a Collaborative Risk-aware Authentication method that takes advantage of any and many devices that the user owns. CoRA increases security, and preserves usability and privacy by using threshold MACs and by tapping into the knowledge of the devices instead of requiring user knowledge or interaction. Using CoRA, authentication tokens are generated collaboratively by multiple devices owned by the user, and the token is accompanied by a risk factor that indicates the reliability of the token to the authentication server. CoRA relies on a device-centric trust assessment to determine the relative risk factor and on threshold cryptography to ensure no single point of failure. CoRA does not assume any secure element or physical security for the devices. In this paper, we present the architecture and security analysis of CoRA. In an associated user study we discover that 78% of users have at least three devices with them at most times, and 93% have at least two, suggesting that deploying CoRA multi-factor authentication is practical today.

1 Introduction

Multi-factor authentication (MFA) is a common way to increase the security of authentication, by using multiple pieces of evidence to establish a user's identity, typically categorized as "something you know, something you have, and something you are". There are many MFA schemes, with 2-factor authentication (2FA) schemes being the most popular, in which in addition to a password, the user has to confirm to the authentication server the possession of a challenge code sent from the server via some pre-established channel (e.g., SMS, smartphone push notification, email) or generated via a pre-established mechanism (e.g., hardware token, app). Unfortunately scaling MFA from two to three or more factors impacts usability, even when it improves security, as handling more factors generally requires more user interaction. This trade-off between security

and usability is an unfortunate property of today's authentication approaches, because the growing adoption of Internet of Things (IoT) is a missed opportunity in using *all* of a user's devices as factors during their normal operation.

The availability and adoption of IoT means users own more than one device and often have multiple devices with them or close by during daily activities. From a study conducted in 2018 [29], we know that users (in US) own an average of 4.4 devices, with a whole 35.9% having six or more devices (and this percentage is an increase from 23.4% in 2017). To further understand how many of these devices are typically available to users we ran a user survey, which shows that at home (where most of the logins take place) over 78% of users have three or move devices present and more than 28% have six or more devices present. (See additional survey results in Section 6.) These devices range from smartphones, to laptops, tablets, smart TVs, game consoles, voice-controlled assistants, connected cars, e-readers, to activity/fitness trackers, smartwatches, smart-home devices (refrigerators, thermostat, camera). This abundance of devices in proximity not only indicates that users have many choices of interfaces from which to log in to a service, but also that these devices could act as digital witnesses to each authentication action. Devices used normally every day have sufficient information about their user to collaboratively provide evidence to the user's (digital) identity. Moreover, multiple devices bring redundancy by removing a single point of failure.

Most proximity-based seamless authentication methods address the usability challenge of MFA by having the various devices communicate with each other to exchange the needed information (the evidence of identity), after the successful establishment of proximity among the devices. Thus, instead of the user transferring challenge codes from one device to another, devices in close proximity with each other automate the process. The existing seamless authentication schemes focus on two factors and provide 2FA security by pairing a main device (e.g., a laptop) with a mobile device (e.g., smartphone [7], smart watch [25]). The primary obstacles of adopting these seamless 2FA schemes are security and availability. If the pre-established channel to the mobile device or the mobile device itself are compromised, the security of the scheme degrades to that of a password-only authenticator. If the mobile device is not present, out of power, or otherwise unavailable, the authentication cannot proceed.

We take advantage of mobile and IoT adoption, and in particular avail ourselves of the fact that users often have multiple devices with them, as confirmed by our survey. Furthermore mobile and IoT devices are networked and capable of connecting either directly or via a hub to the Internet. As a result, a user's collection of IoT devices can more easily exchange information, measure context, and authenticate the user *collaboratively*, creating an ideal platform on which to deploy seamless MFA. To this end, we present CoRA, a Collaborative Risk-aware Authentication scheme allowing for as many factors as available to participate in the authentication process and produce a highly assured identity. A visual comparison of CoRA against other MFA and 2FA schemes is shown in Figure 1.

Having many devices available to serve as seamless MFA authenticators

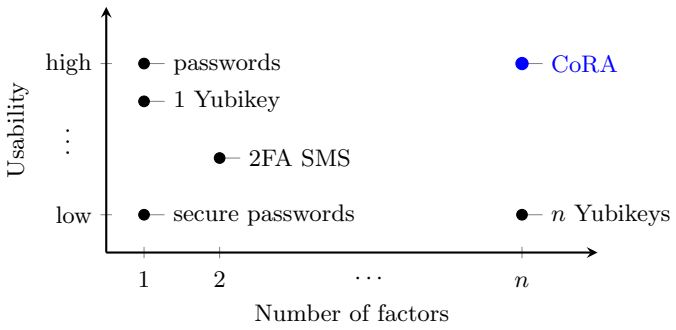


Figure 1: Increasing the factors potentially improves security, however it may affect usability with today's techniques. A password-only authentication system is easy to use but not secure. A secure password is difficult to remember and thus ranks low in terms of usability. Yubikey provides more usability however it has the disadvantages of a single point of failure. CoRA aims to bridge the gap between highly usable seamless authentication and highly secure MFA.

brings in three challenges. First, not all devices may be available at all times, so CoRA must allow for some factors to be absent and must reflect this absence in the produced identity. Second, we must assume that some devices might be compromised (physically or virtually) and thus CoRA must not rely on any single device to place all its secrets in. Third, if some compromise is to be expected, CoRA needs to exclude such compromised devices from participating in the authentication process.

An additional challenge is that of user privacy. We define privacy leak to be the amount of information that the authentication server learns about the user during each instance of the authentication process. Our goal is for the server not to learn any information beyond what is needed to establish a high-assurance identity. In particular, in the IoT context, the user may wish to keep private the set of devices used during an authentication instance, their location, and their state. The privacy requirement adds complexity to any authentication scheme; indeed, if one does not care about privacy, then every device that participates in an MFA scheme can directly communicate its cryptographic evidence (e.g., one MAC or digital signature per device, based on a registered secret key with the server) for the authentication server to verify and tally. This naïve scheme continuously provides information to the server about which devices are in use at what time by the user. CoRA aims to eliminate this privacy concern.

CoRA tackles all of the above challenges while protecting user privacy. Except for the assurance score that accompanies the token, no other information about the devices or the user is sent to the authenticator. In our design, CoRA devices use proximity checking to identify peer devices that are present nearby, apply anomaly detection to determine which of these proximal peers are trustworthy, participate in a multi-level threshold MAC schemes with the trusted ones, and finally compute an authentication token that captures the number of

devices participating (expressed as a cryptographic threshold), the trust shared among these devices (expressed as an assurance score), and information about the authentication request itself (user name for the account, level of authentication needed, etc.). Thus the CoRA privacy-preserving authentication scheme scales to large number of factors, provides risk information, and maintains the usability of a seamless authentication.

In this paper we make the following contributions:

- We introduce a risk-aware, multi-level authentication method, called CoRA, that allows a user to privately employ any number of their devices for authentication, while automatically adjusting for compromised devices (section 3).
- We show how to realize this authentication method through different instantiations of a threshold cryptographic schemes and by applying anomaly detection on each device (Appendix A). Threshold cryptography provides security against credential theft, privacy by hiding device proximity information, and adaptability to varying device-presence conditions, while anomaly detection provides security against device compromise through relative risk assessment.
- We analyze the security of CoRA in the presence of a variety of physical and virtual attackers (section 4), evaluate the performance of our prototype (section 5), and assess the deployability of CoRA through a user survey that studies smart devices and authentication choices (section 6). Our results indicate the CoRA is secure against theft and compromise, adds imperceptible overhead during the authentication task, and is practically deployable now as users have more than sufficient number of devices.

In section 7, we discuss related work in detail.

2 Overview

The CoRA system allows a user to authenticate to a remote server via any of their devices, with the help of the other devices owned by the same user. The intuition is that having, say, five devices owned by the same user that all contribute to the authentication task increases security by raising the bar for the attacker, who now has to control all five devices to authenticate as the user. At the same time, CoRA improves usability by minimizing the explicit authentication work the user has to perform and by relying on automatic cooperation between devices. Here, we explain the threat model, goals, and approach of the CoRA multi-device authentication scheme.

In CoRA, a user owns some number, n, of devices of varying capabilities with respect to computation, security, sensing, and biometric input. A number, $t \in [1 \dots n]$, of devices are available to the user at any given time for performing authentication tasks. For example, a user may own five devices, a laptop, a

smart watch, a smart phone, a smart car, and a smart refrigerator, but only have two of them on hand while traveling (the watch and the phone). CoRA is designed to adjust the authentication outcome to the number t of devices available at authentication time, where "available" means that a device is in proximity to the user.

2.1 Three Authentication Scenarios

Online commerce A user who has six registered smart devices (phone, smartwatch, laptop, smart TV, smart fridge, voice assistant) wishes to make an online payment for a purchase on his laptop. This falls under the category of card-not-present transactions, which unfortunately account for 60%–70% of fraud in payments [13]. One approach to stronger authentication is a form of 2FA that gathers data from the user's laptop (e.g., location, list of apps, other unique system fingerprint) and verifies on the backend, with the downside of weakened privacy. In contrast, CoRA enables a seamless MFA with privacy and similar or higher security. When the user is ready to provide payment information, the laptop talks to nearby devices and asks for "witnesses." If sufficiently many devices answer (as determined by a threshold set by the user, the merchant, and/or the user's financial institution), the user would be given access to make a payment capped only by his credit line or account balance. A low number of present devices, maybe when the user is away from home, will result in a reduced payment capability of up to $50 (for instance).

Access to work email An employee has three registered work devices (laptop, phone, tablet) and wants to log in to check her work email inbox. If the employee is at work and has all of the devices around her, they can all communicate with each other and derive an authentication token of high assurance. Then the enterprise mail server allows full access to the employee's email. If on the other hand the employee is at home and has only her work phone present, a low-assurance token is sent to the mail server, which in turn may allow limited access only to non-confidential messages and attachments. Thus the degree of access would be customized based on the assurance level achieved by the devices present.

Online banking A bank customer has six registered smart devices and wants to log into her online bank account. If the customer is at home and has all of the devices around her, she can use their proximity to construct and present a CoRA token of high assurance. Then the banking web server will allow full access to the account, including capabilities to transfer large amounts of money out of the account and to change the configuration of the account (for example, to add an external money recipient). If on the other hand the customer is at work and has only three devices present (phone, smart watch, laptop), a mid-assurance token will signal to the bank that only balance checks and small money transfers should be allowed. The lowest-assurance token from a single device, when the

customer has only their smart watch with them while on their daily jog, would provide only read-only access to check account balances.

2.2 Threat Model

We assume that an attacker can gain control of a user's devices, either by stealing it and getting physical control or by compromising it by exploiting vulnerabilities in the software or hardware of the device. When an attacker controls a device, we assume they have full access to data and code on the device. In other words, they can make the device(s) in their control refuse to participate in an authentication task, or participate with incorrect data, or trigger fake authentication requests, and so on. For simplicity, in the rest of the paper, we will refer to attacker-controlled devices as *malicious* and to non-malicious devices as *honest*.

We assume that the user's password is compromised (as is usual for any MFA scheme) and a subset of devices are under the attacker's control. (If an attacker knows the password and controls *all* the devices, then he is indistinguishable from the user.) The malicious devices may be in the proximity of honest devices (local attack) or away from them (remote attack).

2.3 Goals

Increased Security If the attacker can get control of the secondary factor (as well as the password) in a 2FA scheme, then he can easily impersonate a legitimate user. If the 2FA scheme is used to log into a bank website, the attacker can log into it too, and has the same control over the funds and assets as the user does. Thus, the compromise of one device completely breaks down the security of a 2FA system (assuming the password has already been leaked).

A multi-factor authentication scheme like CoRA aims to provide stronger security. An attacker should not get full control of a victim's account unless he compromises several devices, which would be much more difficult than compromising one. If a single device is attacked, the damage should be limited (if not zero). As more and more devices are compromised, the amount of damage could increase. In other words, several devices must be compromised to inflict the same amount of harm as in a 2FA scheme.

There are other types of attacks possible such as denial of service, authentication under duress, social engineering, etc. We do not consider them in the scope of CoRA.

Improved Usability Improved usability means that either the user has to memorize fewer secrets than existing schemes or the interactions required for authentication are limited (e.g., only to initiate the authentication task, which is required in most cases to ensure user's intent to authenticate). For example, for this goal, seamless 2FA methods (that do not require user interaction) score higher than interactive 2FA schemes.

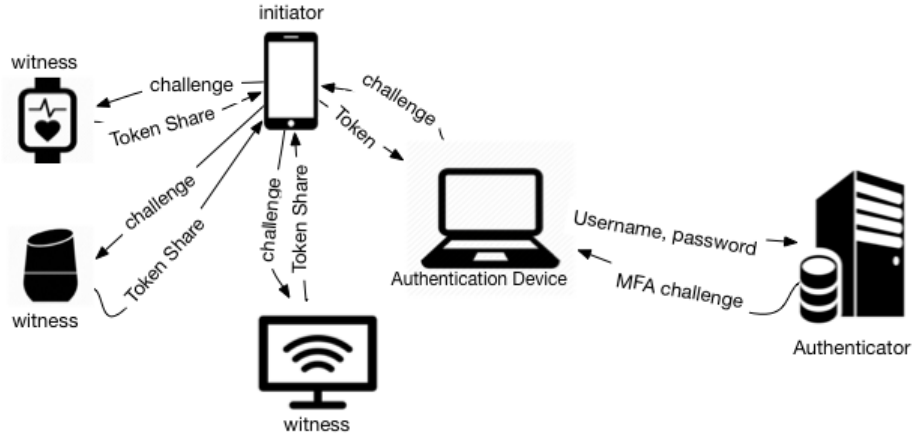


Figure 2: High-level architecture of CoRA.

One way to improve usability is by making the best use of all the devices that *are* available instead of expecting the user to fetch more devices at the time of authentication, which would frustrate them. The challenge is achieving this usability goal without compromising the security goal.

Privacy A typical approach to de-risking the authentication task is to collect as much data as possible about the user, her device(s), and their environment, ship this data to the server, and use anomaly detection to trigger a secondary step-up authentication [10], in which the user has to perform additional tasks to prove her identity. Applying this concept to CoRA's multi-device setting would expose information about all of one's devices, one's personal preferences for each device, and one's choice of devices in different settings (home, work, gym, park). This is a significant intrusion into the one's digital life and would limit adoption by privacy-conscious users. Our goal is to minimize the amount of information sent to the server, to preserve user privacy.

We define our privacy goal with respect to each authentication operation. The server could learn the number of devices that participated in the operation and an overall risk assessment, but not which devices participated nor anything about their context (location, etc.). An important distinction we make is that the privacy goal applies to the authentication operation, but not to the registration in which devices are added or removed. We assume that the user agrees to expose the list of devices they own to the server, such that the server can identify the device that sends the authentication request.

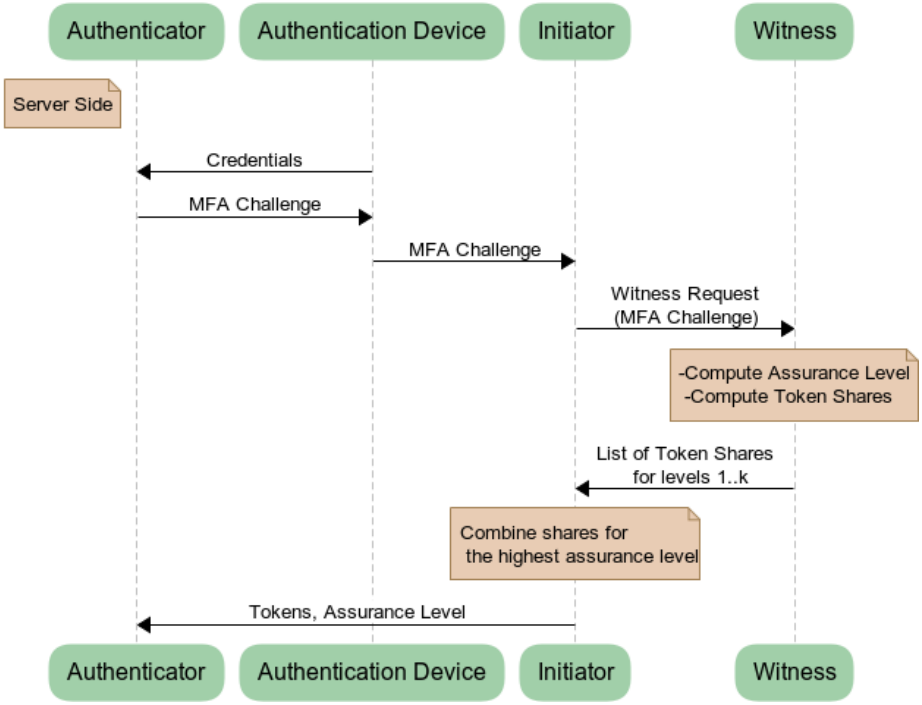


Figure 3: CoRA protocol as MFA

3 System Architecture

This sections explains CoRA high-level architecture (Figure 2), the interactions of components in the distributed design of CoRA, and its building blocks and operations.

3.1 High-level Architecture

For CoRA to work, user's devices needs to be registered with the authentication server, and to receive cryptographic key shares for one or more instances of the assurance level. When a device is unregistered (either due to end of life cycle or due to theft), the server and the remaining devices generate and share new cryptographic keys. In addition to the state and date kept by the server, each device maintains trust scores for every peer device in the registered set; the trust score is computed using anomaly detection over the history of interactions between pairs of devices. The trust scores are stored with and shared only by users' devices and not sent to the authentication server. Devices can thus mutually verify that they belong to the same user, can participate in a threshold cryptographic scheme to compute tokens jointly, and can decide on their own which of their registered peers can be trusted at any given moment.

3.2 Definitions

Before we explain CoRA's protocol, we define the following roles and concepts for the authentication scheme:

- **Initiator**: A device that initiates MFA and token generation by collaborating with other devices.
- **Witness**: Any device that agrees to collaborate with the initiator to compute an authentication token.
- **Whistle Blower**: A device that could directly observe malicious activity and spreads the information to other devices, which may be blind to the attack.
- **Authenticator**: An entity (e.g., server) that offers services and authenticates a user by verifying a token.

Additionally, we define the following concepts that are essential for CoRA's operations:

Trust Score. This score shows the level of trust a device puts into its neighbor. The trust is based on observed behavior of the other device, and its perceived location. Each device maintains a vector of trust scores for all of its neighbors. The trust score is not representative of one specific authentication session, but it is an indication of how "normal" and "reliable" a neighbor behaves and is configured.

Assurance Level. For each authentication request, the MFA token is accompanied with an assurance level that represents the trust and health of the initiator as well as its proximity to other devices. In other words, this assurance level assures the authentication server that the user is authenticated with this degree of certainty.

3.3 CoRA Protocol

When the user attempts to authenticate herself to the server (Figure 3), for example, by entering their credentials in a web browser on a laptop, the web browser connects to one of user's registered devices (e.g., smartphone) via a local communication scheme (e.g., over WebAuthn [31]). This device, which we call the *initiator* for the duration of an authentication operation, discovers all other registered devices in a close proximity. The locally available devices together with the initiator form the set of *present* devices. A registered device that is both present and trusted is the ideal candidate for an initiator.

At this point the initiator launches a request to all the trusted and present devices for the joint computation of an authentication token. Each device receiving the request determines whether to participate in this joint token computation depending on their trust in the initiator. For example, if there are a total of five devices present (including the initiator), but the initiator is trusted by only two others, only three devices (including the initiator) will jointly compute

the authentication token, and this token will reflect the number of devices that participated.

Once the initiator and the participating devices compute the authentication token at the highest assurance level possible, the initiator will provide the token along with the assurance level back to the web browser, which can proceed to send it to the authentication server. Now the authentication server has several pieces of evidence for the user's identity: the password entered in the browser, the device identity of the initiator, and the CoRA authentication token defining the number of participating devices. These items are then used to make the authentication decision.

3.4 Building Block: Threshold Cryptography

CoRA uses threshold cryptography to ensure no single point of failure and to provide privacy. The main idea behind threshold cryptography is to distribute the secret key material among several devices (instead of storing it entirely on one device) such that at least a threshold number of them are needed to perform cryptographic operations. For example, the signing key of a digital signature could be protected by distributing it among, say, 5 devices. If the threshold is set to 2, at least 2 devices are needed to decrypt any ciphertext.

Importantly, the key material is *never* reconstructed in its entirety on *any* device—even when cryptographic operations are being performed. Thus, no device becomes a single point of failure at any point. Furthermore, threshold cryptographic schemes are backwards compatible: the signatures of a threshold signature scheme (for instance) are no different from the signatures of the regular version of the scheme, no matter which devices participate [22]. This has two main benefits: the verifier code does not need to be modified and it does not get to learn who participated, thus providing some privacy.

We investigate the suitability of two types of authentication schemes for CoRA: symmetric-key based message authentication codes (MACs) and public-key based digital signatures. They are used to confirm that a message came from the stated sender. MACs can be viewed as the symmetric-key (same key or one key) analogue of digital signatures. In a MAC scheme, a secret key is used to authenticate messages. The same secret key is also needed to verify authenticated messages. In a digital signature scheme, on the other hand, there are two keys: a signing key and a verification key. As the names imply, the signing key is used to sign messages while the verification key is used to check if a signature is valid.

A secure MAC scheme guarantees that only with knowledge of the secret key, a message can be authenticated. Digital signatures provide an analogous guarantee: only with knowledge of the signing key, a message can be signed. Both MACs and digital signatures have been studied in a threshold setting. These constructions have very different properties and we will discuss them in detail in Appendix A.1.

To study the algorithms of threshold MACs and signatures in the same framework, we define the notion of a *non-interactive threshold-authentication*

scheme which consists of four algorithms, `Setup`, `GenPartTok`, `Combine` and `Verify` defined below*. We use *token* to refer to an authenticated message for MAC and to a signature for threshold signatures.

- `Setup` takes number of devices n and threshold t as inputs, and outputs a verification key `vk` and a list of n shares of a master secret key `msk`. (For MAC, both `vk` and `msk` are the same.)
- `GenPartTok` takes a message and one of the secret key shares as inputs, and outputs a partial token.
- `Combine` takes at least t partial tokens as input and combines them to produce a full token for "level t". (If fewer than t tokens are provided, `Combine` would fail.)
- Finally, `Verify` takes the verification key and a full token as inputs, and outputs 1 if the token is valid.

Choice of Threshold Authentication. In Appendix A.1, we discuss four different ways in which non-interactive threshold authentication schemes can be instantiated (two instantiations for threshold MAC and two for threshold signatures). We compare these schemes based on computational overhead, storage and communication overhead, and security properties. Our analysis shows that a threshold MAC based on AES is the most suitable for our purposes.

3.5 Multi-level threshold authentication

If a threshold authentication scheme has a threshold of 3, then at least 3 devices are needed to authenticate a message properly. This could be too restrictive in certain situations like the setting we consider in this paper, where a user could only have 1 or 2 devices at certain times of the day. Therefore, we have *multiple instantiations* of a threshold scheme in CoRA with different threshold values. Each instantiation has a separate set of key shares, verification key, etc. When the number of devices available is low, a token of a lower threshold instantiation can be constructed. This makes CoRA more usable and flexible.

The Assurance Levels in CoRA map to the instantiations of a threshold scheme (in other words, to different thresholds). A simple mapping between levels and thresholds could be the identity map, i.e. level ℓ would correspond to a threshold of ℓ. With this map, at least three devices would be needed to generate a token of level three for instance. However, we only require that if an assurance level i_1 maps to a threshold level j_1 and an assurance level i_2 maps to a threshold level j_2, then if $i_1 < i_2$, it must imply that $j_1 < j_2$. Simply put, a lower level corresponds to a lower threshold and provides lower assurance

*The threshold-authentication notion defined is *non*-interactive because a combiner does not need to interact with partial token generators over several rounds. The latter just uses `GenPartTok` to create partial tokens and the former uses `Combine` to combine them.

(because a smaller number of devices need to be present), while a higher level involves more devices and provides a higher assurance.

CoRA provides the designer of an application with the flexibility to chose the number of levels, their interpretation, the thresholds values associated with them, etc. For e.g., a payment application can have three assurance levels where an assurance level i maps to a threshold value of i. A token of the lowest assurance level, which can be generated from any of the devices, may only allow purchases of (say) up to $10; a token of assurance level two may allow purchases of up to $500; and, the highest assurance level token can allow any purchase up to the credit limit.

3.6 Building Block: Anomaly Detection

CoRA precedes any threshold-cryptography step with a decision-making phase, where each device checks the trust score of the initiator and decides if and at what level to participate in the threshold MAC. The trust score of the initiator is based on (1) features observed locally by the participating device (witness) and (2) scores received from the network. A device continuously runs a local anomaly-detector (which we call *TrustMeter*) and updates the trust score of its neighbors as it observes new events (e.g., network traffic).

The local trust score is the result of the intrusion detection scheme running on a device, and is computed for each of the device neighbors, denoted by LS_{ij}, and is normalized as follows:

$$c_{ij} = \frac{LS_{ij}}{\Sigma_j LS_{ij}} \tag{1}$$

Trust Propagation. To take into account the view of other devices in the network, and to value their opinion, the local trust scores are aggregated with the score received from device peers. The trust propagation works based on the idea of transitive trust established by EigenTrust [14], where a device trusts the opinion (i.e., the trust scores) of the devices it trusts. One way to calculate the trust score that device i has for device j, written $T_{i,j}$, is to use a weighted average where the weight is the (normalized) local trust score of device i for its peer k:

$$T_{i,j} = \Sigma_k (c_{i,k} \times c_{k,j}) \tag{2}$$

As the EigenTrust [14] shows, with enough rounds of propagation, the trust score of each device converge to the same value, from all devices point of view. In section 5 we discuss our evaluation results for how the number of propagation rounds affect the authentication result.

For every authentication request, the TrustMeter component of CoRA (running on a witness device) calculates the assurance level, that takes as input the trust score for the initiator device id $TS_{i,j}$, and the proximity of the initiator. The witness device participates in the MFA request, by providing shares for up to assurance level.

Choice of Anomaly Detection. CoRA takes advantage of lightweight anomaly detection techniques to build the TrustMeter. We have quite a few choices for anomaly detection on IoT devices. In Appendix A.2, our analysis of four different detection schemes shows that SVELTE [21] is the most suitable option for us.

3.7 CoRA Operations

The core operations in CoRA are a) authentication, b) device registration, and c) device removal. We assume secure channels for all the communication among the devices and the server, created through mutual authentication using keys generated at device-registration time. Additionally we discuss the operations that the server undertakes at device registration and removal.

Authentication The user enters his username and password on a device (e.g., a laptop, not necessarily registered with the server) to be sent to the authenticator and in turn, if the user credentials are valid, the authenticator sends a challenge to the user's device. The laptop contacts one of user's registered and present devices and that device becomes the initiator. The initiator contacts all of user's other present devices to become witnesses for this authentication request. Each witness checks their trust in the initiator and its proximity, and decides if and at what assurance level to participate. A witness computes the token shares needed for the challenge at the determined assurance level as well at all the levels below, and sends the computed token shares (if any) to the initiator. The initiator waits[†] to receive as many responses as possible, finds the highest assurance level for which it has sufficient shares, and combines the corresponding shares into a token. The token along with the assurance level is sent to the authenticator (through the laptop) for verification. 1 explains the details of the initiator requesting witnesses and then combining the shares, and 2 explains the task of a single witness in this interaction.

Device Registration The registration of a new device for CoRA services is performed by using CoRA itself as an authentication scheme. Because registration is a sensitive operation for the security of the whole system, we also require that the user participates in the process. Device registration is done by collecting the approval of already registered devices in an opportunistic manner. The more devices attest to the addition of the new device, the higher key shares the new device would receive. This allows the gradual and natural introduction of a new device in the user's network; as the user carries the new device to different environments (home, work, etc.), the existing devices have the opportunity to connect to the new device when in proximity and produce key shares for it. We use CoRA as MFA to secure the registration phase beyond just using a password for authentication. Adding a new device is done in two steps:

[†]We chose the wait time to be 2 seconds in our implementation, however, the number might be chosen dynamically based on the network conditions.

Algorithm 1 Witness Request: To create an authentication token, a device broadcasts a request to its neighbors for collaboration. The procedure `sendMsg` takes as input the receiver address d and the data to be sent.

```
1:  procedure WitnessRequest(c)
2:              // c is the challenge from the authenticator
3:              // Witness responses are collected in all_shares
4:      ∀l ∈ [1..MAX_LEVEL] . all_shares[l] ← ∅
5:      for d ∈ Nearby&RegisteredDevices do
6:          sendMsg(d, WITNESS_REQUEST‖c)
7:
8:              // Wait for witnesses to respond
9:      wait(2 seconds)
10:
11:     T ← ∅
12:     for l ∈ keys(all_shares) do
13:         if |all_shares[l]| ≥ THRESHOLD_l then
14:             T ← T ∪ ⟨l, Combine(all_shares[l])⟩
15:
16:     assuranceLevel ← max{l : ⟨l, x⟩ ∈ T}
17:     let token s.t. ⟨assuranceLevel, token⟩ ∈ T
18:     sendMsg(authenticator, ⟨token, assuranceLevel⟩)
19:
20: procedure MessageReceived(msg)
21:             // msg is the message received from a witness
22:     for ⟨l, ts⟩ ∈ msg do
23:         all_shares[l] ← all_shares[l] ∪ {ts}
```

Algorithm 2 Witness Response: As a response to witness request messages, a witness device responds to the initiator with a token share.

```
1:  procedure WitnessResponse(msg)
2:          // Current witness is device i
3:          // msg is the message received from initiator
4:          // K_j is device i's secret key share for level j
5:
6:          // AL depends on proximity of device i
7:          //    to the initiator and on the trust
8:          //    it places in the initiator
9:      AL ← AssuranceLevel (i, initiator)
10:
11:     shares ← ∅
12:     for level ∈ [1 ... AL] do
13:         token_share ← GenPartTok(K_level, msg)
14:         shares ← shares ∪ {⟨level, token_share⟩}
15:     sendMsg(initiator, shares)
```

Step 1. The user logs into the device-management portal using her username, password and registers the new device using an interactive 2FA or MFA protocol (any will do, the goal is to verify user consent to device registration). The new device then sends its public key to the portal and in turn receives a TLS certificate along with the key share for assurance level one.

Step 2. At any point in time after *Step 1* above, the new device runs a discovery algorithm to find any of the user's other registered devices and sends them the message REGISTER_REQUEST$\|$*certificate*. Each of the registered devices, on receiving a valid certificate in the registration request, prompts the user for permission to accept the new device which the user has to provide via an interaction with the device. Each device then generates token shares for all levels and sends them to the new device. The new device collects token shares from existing devices and when sufficient shares are obtained for any one level, combines them into a full token. On receipt of a token at a given level, the Authentication Server provisions the new device with a key share for that level. These steps are similar to a run of the authentication algorithm, with the distinction that the AssuranceLevel invocation is replaced by a user prompt. Finally, adding a new device might increase the maximum assurance level, which means that the server will send key shares for the new assurance level to all registered devices.

Device Removal In removing a new device, all devices (in proximity or remote) have to be notified to remove that device from their list and set its trust score to zero. If the user is removing a device, the key shares or any CoRA related data would be wiped deleted. If the device is stolen, the keys for all levels have to be updated on the authenticator side and on all devices. Updating the keys for all devices may not be efficient but one can reasonably assume this

operation is rather infrequent.

Server Actions on Device Registration and Removal When a device is added through registration or removed, a server has several options for changing its authentication requirements. In CoRA authentication requirements are expressed through assurance levels, each one defined by the number of devices needed to participate to achieve that assurance level. Thus, when the number of devices increases from n to $n + 1$, the server can:

1. *Favor security* by increasing the device-participation thresholds for each assurance level. This way more devices are needed to reach the same level of assurance as before the new registration, with the downside that usability may decrease, as the user has to carry more devices.

2. *Favor usability* by keeping the thresholds the same as before. Now the user has more freedom in choosing devices to carry with them in various settings, to reach the same level of assurance as before. There is a slight decrease in security, as the attack surface is larger.

3. *Take a hybrid approach* by increasing thresholds for some assurance levels (e.g., the higher ones) and keeping the thresholds unchanged for the other assurance levels.

4. *Require the user to remove a device*, maybe by having the user select a device that is infrequently used and could be removed without impacting usability. There is no change in security or usability.

A symmetric set of choices arise when a device is removed. As applications define their own assurance levels and device-participation thresholds, we expect that different application servers will take distinct, risk-adjusted approaches to managing the assurance levels.

4 Security & Privacy Analysis

4.1 Security Analysis

Recall the threat model and security goals discussed in Sections 2.2 and 2.3. We consider an adversary who knows a user's credentials and his goal is to authenticate as the targeted user to a remote server at the highest assurance level possible. If the attacker does not have control over any of the victim's registered devices, then he cannot initiate the MFA process. So we assume that he has compromised at least one device.

Strongest attacks If all devices are malicious (i.e. compromised), then we cannot hope to get any security. So let us assume that a strict subset of the devices are malicious. The honest devices may be in the proximity of malicious ones (local attack) or away from them (remote attack). In the first case, if

malicious devices are able to fool the TrustMeter (e.g., by acting quickly, evading intrusion detection), they can appear *trusted* to be honest devices. In the second case, when malicious devices are not in proximity, attacker's task is much harder. He also needs to fool the proximity check system.

The attacks described above allow the adversary to obtain the approval of all the devices (if only for a small amount of time) and create a token at the highest assurance level. However, compromising devices without being noticed, as well as evading the proximity checks is highly difficult to carry out, especially against a diverse set of devices.

With a well-designed proximity check mechanism and an accurate TrustMeter, which can quickly detect intrusions and effectively propagate such information, CoRA can prevent such attacks and limit them to a low assurance level.

Common attacks More generally, an adversary would have control over some of the devices, say ℓ, but cannot trick honest devices into collaborating. Take the common problem of device theft for instance: honest devices will not collaborate with stolen ones because they are not in close proximity. In such cases, the attacker has access to all the cryptographic material on the ℓ devices but nothing else.

Now, a secure threshold token generation scheme like `AES-MAC` guarantees that no token of threshold higher than ℓ can be constructed from ℓ devices. Thus, the attacker can only construct tokens for those assurance levels that require ℓ devices or less. For example, if two devices are stolen from a user and an assurance level has a threshold of three, then it is infeasible for the attacker to authenticate at this level. The adversary could still generate tokens at lower assurance levels, but those levels would provide limited capability. This increased security should be contrasted with 2FA where compromise of a single device grants full access.

4.2 Privacy Analysis

Privacy is an important goal for CoRA and the use of threshold cryptography helps achieve it. In a threshold token generation scheme, the final token does *not* depend on which t devices participate. In other words, the validity of a token ensures that at least t devices participated, but not which t out of n. Consider the `AES-MAC` token scheme for instance, described in detail in Section A.1. A token on a message m here is defined as the XOR of *all* $f_k(m)$ for $k \in U$, where f is a PRF like AES and U is the set of *all* keys. Any set of t or more parties has all the keys in U, so they can collaborate to generate tokens. However, the final token looks the same whether it comes from one set of parties or another.

Thus, CoRA is able to provide privacy guarantees to users. The authenticator does not learn which devices participated, nor any information about their location, or behavior. All the authenticator learns is the highest assurance level at which the devices decided to collaborate with each other. Any trust or mistrust that the devices have for each other, captured as trust scores in

the TrustMeter, is only to determine the level of participation. They are never shared with the authenticator.

5 Implementation and Evaluation

We have implemented CoRA in Java, for desktop and Android devices, including a phone, and a smart watch. CoRA implementation consists of three components: 1) a CoRA app that allows Android devices to register for CoRA to act as an initiator or a witness device, 2) a browser plug-in so that the laptop could also act a CoRA device, and 3) an authentication server component that supports device registration, share computation, and token verification. The code included 1982 lines of Java code, 98 lines of JavaScript. We have tested all the components together, with one authentication server, one laptop running a shopping site in the browser, three smart phones, and a smart watch. The devices communicate via HTTPS and are all assumed to have a WiFi connection. In our implementation, the initiator waits for two seconds to receive witness responses. However, in all of our experiments, all the token shares were received by the initiator in less than 500 ms. In addition to the code for desktop and mobile devices, we have implemented a simulator to evaluate CoRA in scale. The simulator has the same crypto and authentication logic as the actual code, however, the network communication, the device compromise, and anomaly detection have been simulated.

Security and usability of CoRA could be tuned by adjusting the number of registered devices, assurance levels, and the threshold value for token generation. In the following, we discuss three settings for the parameters and explain how CoRA compares against 2FA.

Assurance Level 1: $n = 6$ devices, threshold $k = 1$ In this simple setting, any of the six registered devices could be used for the second factor of authentication. Security is on par or lower than that of any 2FA using one device only, since the attacker could choose any of the six devices to compromise (larger attack surface than having one device only). However, the usability is higher than a 2FA system because the chances of having one of the six devices nearby and available is higher.

Assurance Level 2: $n = 6$ devices, threshold $k = 3$ The user needs to have any three out of their six devices nearby and trustworthy enough to collaboratively compute the MFA token. Security is higher than other 2FA schemes, as the attacker has to compromise or steal at least three devices. Usability is also higher than interactive 2FA (because there is no need for user interaction in CoRA), yet potentially lower than seamless 2FA (since the user must have at least three devices present).

Assurance Level 3: $n = 6$ devices, threshold $k = 6$ In this case, the user needs to have all six devices present and behaving normally. This produces

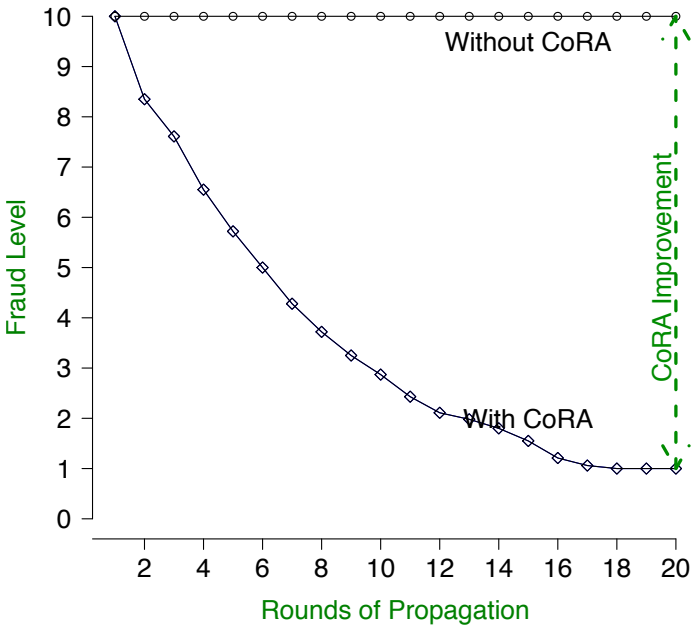
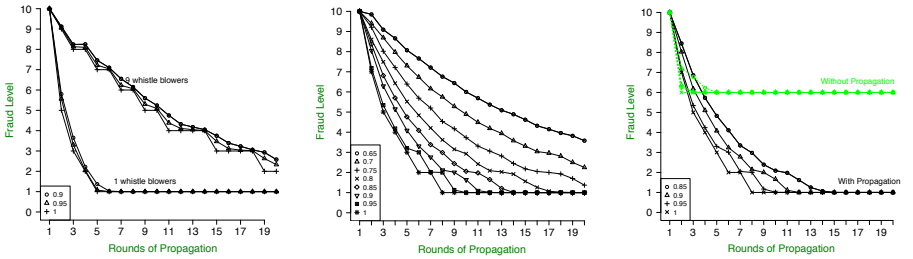


Figure 4: CoRA reduces the fraud level, even in the presence of compromised devices, by propagating trust information across devices. In this experiment the user has ten devices, one of which is compromised and four of which are whistleblowers with a TrustMeter accuracy of 0.9, running for more than 20 rounds. When the compromised device tries to authenticate itself, it can only generate a token at assurance level of one. Without TrustMeter, the compromised device would have been authorized at the highest level. At propagation round one, we are assuming the adversary is undetected, and therefore it convince all witness devices to participate in the protocol.

the highest level of security, more than other 2FA schemes, as all six devices must be compromised for an attack to succeed. Usability is somewhere in between interactive 2FA and seamless 2FA (since having all six devices present for authentication may be burdensome).

5.1 TrustMeter Evaluation

The TrustMeter plays a key role in the security of CoRA. We dedicate this section, to evaluate how the TrustMeter accuracy and propagation method impact the security of authentication. For scaling our experiments to more than a few devices, and to easily tune the parameters, we have implemented a simulator that takes as input the TrustMeter accuracy, number of devices, number of whistleblowers (i.e., devices who observed directly a peer device that is malicious), number of malicious devices, and the option of trust propagation. To evaluate the effect of each parameter on the security of CoRA, we have defined the metric, fraud level, the highest assurance level that the adversary could au-



(a) The higher the number of whistleblowers, the faster the fraud level is reduced.

(b) The lower the accuracy of trustMeter, the longer it takes to reduce fraud level.

(c) The trust propagation informs witness devices about malicious initiators, even when they cannot directly observe the malicious activities of the initiator device.

Figure 5: CoRA's use of trust measurement and trust propagation.

thenticate itself at. In all the experiments, there are ten devices, all connected, but not all of them are necessarily able to directly detect the malicious activity of the attacker. In all the scenarios, the initiator is malicious (in order to initiate the token generation).

We first consider a simple case where there are four whistleblowers (devices that could directly observe the malicious node) with a fixed accuracy of 0.90 and the trust propagation enabled. As the trust information travels through the network, after sixteen rounds, all devices have updated their score about the malicious device and do not trust it enough to help it in authentication. After 16 rounds, the best the malicious device can do is to authenticate itself at assurance level one (Figure 4), since the initiator has one token itself. Without CoRA (specifically the trustMeter, and the crypto module), the compromised initiator would be undetected, and all devices would help it be authenticated at the highest level.

Another factor that affects the fraud level is the number of whistleblowers in the network. Figure 5a presents two extreme cases where there is only one whistleblower in the network (e.g., only the router observes the anomalous behavior) versus the case when every device in the network can directly catch the anomalous behavior of the malicious device (e.g., DDoS and heavier than usual traffic could be noticed by all devices). This means for a network where not all of devices have direct connections to observer the malicious behavior, either the fraud level is higher or one could tune the frequency of running TrustMeter to compensate for the slow detection.

Accuracy of the anomaly detection that feeds the local trust scores affects how fast the compromised initiator could be detected. For the experiment results shown in Figure 5b, there are four whistleblower devices. In the experiments, we set the accuracy value in a range of 0.65 to 1. The results show that

although accuracy is an important factor in fraud level, however if the malicious behavior is consistent, then even a low-precision TrustMeter and enough rounds of trust propagation could lead to the malicious device being detected by all witnesses.

By propagating the trust score to other devices, the protocol ensures that all the devices in the network learn about the malicious devices, even those whom they could not directly observe. If, on the other hand, trust propagation is disabled and each device only acts on its own observations, the adversary will be able to collaboratively generate higher-level tokens and CoRA's capacity to reduce fraud is capped (Figure 5c).

5.2 Performance Evaluation

The two main CoRA operations at the time of authentication are token share generation on witness devices and token share combination on the initiator. Assuming six registered devices, if three of the devices participate at assurance level 2, it takes about 32 ms to generate token shares for all levels, and about 7.5 ms to find the best level & combine the shares (Table 1). The testing is performed on a smartphone with a 2.36 GHz CPU. The measurements here are larger than the numbers shown previously in subsection A.1 due primarily to the performance gap between a laptop and a phone.

Table 1: Performance of threshold MAC operations, measured on a smartphone at assurance level 2 (average over ten experiments).

Operation	**Mean (ms)**	**StdDev**
Generate share	32.10	1.37
Combine shares	7.60	0.52

Contemporary IoT devices are sufficiently powerful to perform on par with a smartphone. For example, devices such as the Nest thermostat, the Huawei Watch 2, and Google Home all have 1+ GHz CPUs and 0.2-4 GB of RAM. For a single core, 1GHz device, on average it takes about 124.8 ms to compute a threshold MAC token. This number includes the communication time (over wifi) and threshold MAC computation. This number is negligible compared to the 25 seconds delay of today's interactive 2FA system [30].

6 Usability of CoRA

Since CoRA does not require any user interaction and only availability of devices is sufficient to authenticate users, the main question regarding usability of the system is whether users have access to their personal devices in various authentication settings. To understand feasibility of CoRA in different settings and plan for its real-life deployment, we ran a survey study discussed in this section.

Table 2: Demographic Information

Gender	Female: 25.32%, Male: 73.42%, Other: 1.27%
Age	18-24: 1.27%, 25-34: 50.63%, 35-44: 26.58%, 45-54: 12.66%, 55-64: 67.59%, 65+: 1.27%
Education	Less than high school degree 1.25% High school degree or equivalent 0.00% Some college but no degree 1.25% Associate degree 2.50% Bachelor's degree 23.75% Master's degree 45.00% Doctorate degree 26.25%
Computer Science Professional	Yes 82.50% No 17.50%
General Computer Skills	Excellent 48.75% Very Good 37.50% Good 12.50% Fair 0.00% Poor 1.25%
General Computer Security Skills	Excellent 23.75% Very Good 36.25% Good 27.50% Fair 10.00% Poor 2.50%
Comfort with New Technology	Extremely Comfortable 48.75% Very Comfortable 41.25% Moderately Comfortable 10.00% Slightly Comfortable 0.00% Not at all Comfortable 0.00%
Salary	Less than $20,000 1.25% $20,000 to $34,999 2.50% $35,000 to $49,999 1.25% $50,000 to $74,999 1.25% $75,000 to $99,999 7.50% Over $100,000 58.75% Prefer not to Answer 27.50%

6.1 Survey Setup

We set up a survey on LimeSurvey service which consisted of 20 questions presented in three parts:

I: Demographics & technical background. The questions in this category poll for the participants' demographic information (i.e., gender, age, education and income), and their computer science and security background. In asking these questions, we aim to understand the diversity of the population and the possibility of generalizing the responses.

II: Smart devices. In this set of questions, we asked the participants about their possession and daily use of smart devices. The intention behind these questions is to understand the population's usage of smart devices and assess practicality of CoRA in different scenarios.

III: Passwords & 2FA. We asked the participants about their knowledge of passwords and two-factor authentication, the motivation to adopt them, and the reason if not so. We also asked about the services they think need strong authentication. This set of questions help us understand whether the population willingly uses 2FA for security or, if not, how we can offer an authentication method that addresses such concerns.

6.2 Survey Results

We followed our organization's privacy procedures to obtain permission to collect data for this survey. We advertised the survey on a few organizational mailing lists, and spread the word among colleagues and friends. We collected responses from 80 participants. We informed the participants about the intention of the study. We assured them that the data collected will solely be used for the purpose of the study and will be analyzed anonymously in an aggregated format. We summarize the responses here.

I: Demographics & technical background. Table 2 shows the responses to the demographic questions. Majority of the population are young, educated, relatively high income, young professionals with background in computer science and good understanding of computer security and motivated to use new technology. While this population seems to be above average in terms of owning and using smart devices, these early adopters of the technology in fact represent where the market would take us in the future. Examples are mobile phones that are now replaced with smartphones, TVs offering internet connectivity, and hand watches being replaced with smart watches. Therefore, this demographic represents the general population in near future.

II: Smart devices. We asked the participants about smart devices they own. We listed 14 smart devices as shown in Figure A.1, and asked the participants to select devices they own and name those not listed. Almost all participants

own a laptop and a smartphone. Over 66% have tablets and smart TVs. Game consoles, smart home devices, digital media player, smart watch, and activity trackers, were owned by close to 40% of participants. Other less popular devices were connected cars, VR, smart thermostats, and home security cameras.

We then asked participants to estimate the fraction of their daily logins that happen from different locations including home, work/school, while commuting, while shopping, and at restaurant/coffee shops. We consider these five locations to represent settings where most of the users' authentications happen. We then asked the participants to select the number of smart devices they have available while logging in from each of the five locations. We explained that by "availability" we mean the devices are charged and in proximity of the login client. Answer to these questions clarifies the availability of devices during authentication and hence practicality of CoRA.

The response as displayed in Figures A.2 and A.3 shows that most of the times users log into websites and services such as social networks, banks, and email, while at home or at work. The average frequency of login was reported as: home 47.75% (20.54), work 35.16% (20.15), while commuting 10.58% (8.43), while shopping 3.75% (3.85), and at restaurants and coffee shops 4.57% (4.81). Importantly, the two locations with the highest login frequency also had more devices available. At home, 40% of the participants had three or four devices available and close to 30% had six or more devices available. Overall, around 78% of the participants had three or more devices available and more than 93% had two or more devices available at home. At work, 60% of the participants had two devices available and over 85% had at least two devices available. Number of available devices in the locations with lower login frequency, such as during a commute, while shopping, and at restaurants, was on average lower, with the majority of participants having one to two devices available and a few having more devices available.

III: Passwords & 2FA. We asked the participants several 5-scale Likert questions about their familiarity, frequency of logins with passwords and second factors, and motivation for adopting or declining second factors. The results is presented in Figures A.4 and A.5. While close to 70% of the participants very frequently or frequently visit websites that require password authentication, only close to 45% of participants very frequently or frequently authenticate to websites using second factors, and 29% of them only sometimes use second factors. Over 70% of the participants are extremely or very familiar with passwords and second factors, and all the 80 participants have used second factors in past. All the participants reported they are familiar with PIN/Code based 2FA, and majority of them are also familiar with security tokens and push based 2FA.

On the motivation questionnaire, as shown in Figure A.6, only 32% of our highly educated and tech savvy participants reported using second factors voluntarily. 71% of participants use second factor because it is mandated by the service providers and 62% use it because it is mandated by their employer. 56% of the participants are aware of security improvements second factors provide.

Questions about reasons for not using second factors, as presented in Figure A.7, give a valuable insight into the importance of offering multi-factor techniques that are more efficient and easier to use. 65% of the participants reasoned that second factors make the login process slow, 15% thought second factors are difficult to use, and 12% were concerned about the secondary device not being available during login (e.g., out of battery). Note that 2FA adoption may in fact be lower in general public given the demographics of our participants.

Finally, we asked about services for which the participants find security to be critical. Majority of our participants understand the importance of second factor authentication as noted in their responses. Almost all this population (96.25%) uses or prefers to use second factors for banks and financial services. Other services such as healthcare (67.50%), emails and messaging (58.75%) and social networks (41.25%) were also among the services that can benefit from second factors. 26.25% of participants also prefer to use second factor for online shopping websites, perhaps because payment and address information are stored on these websites. These choices are inline with the responses to the question about the required level of security for services. Almost all the participants considered security of banks and financial institution to be extremely or very critical, followed by 88% for healthcare, 80% for email and messaging services, close to 60% for shopping and social networks, and only 35% for entertainment services. The responses to these questions are presented in Figures A.8 and A.9. Answer to this set of questions along with the number of devices people own help us plan for real-world deployment of CoRA.

7 Related Work

Zero-interaction authentication (ZIA), first introduced by Corner et al. [6] aimed to improve security of authentication in a seamless way by asking the user to wear an additional device that communicates with the laptop over a short-range to prove its proximity. With the emergence of smart phones, a rich body of work focused on distant-bounding techniques for ZIA by using the available resources and sensors on smart phones. We discuss a summary of distance bounding works in this section, as each of these work could be scaled to multiple devices using CoRA.

Halevi et al. [12] propose a secure proximity detection scheme based on ambient sensors such as audio and light data. PhoneAuth [7] uses Bluetooth communication between the phone and the laptop to prove proximity for 2FA. Drones to the rescue [23] offers a purely ambient physical sensing approach to ensure proximity of prover and verifier to prevent replay attacks. Marforio et al. [17] propose a location-based verification approach that relies on the trusted environment available in some phones. Sound-Proof [15] proves the proximity of user's device by recording audio on the phone and the laptop at the same time, and comparing the two on the phone. While Sounds-Proof is seamless, browser-compatible, and effective, it remains secure only if the phone is not compromised [24]. Listening watch [25] proposes separating the recording role and

the comparison role in two devices, and also makes the audio more specific to a code sent by the server. While this work defends against near-far attacks, it is still vulnerable to compromising the phone or the communication link between the phone and the web-server. Since the phone's accept or reject message is not bound to the response of the watch, a compromised phone could simply verify being co-located with the laptop (browser). Truong et al. [28, 27] investigated different channels for secure verification of proximity and concludes that fusing multiple sources would improve security if the attacker cannot compromise multiple channels at the same time. CoRA defends against such an adversary by using multiple devices that are diverse in their hardware and software and more difficult to compromise simultaneously.

While the above seamless 2FA contributions improve usability, they are all dependent on one source of information and on one device, which could be replaced by a distributed system (e.g., CoRA) to improve security and reliability. In particular, CoRA makes it cryptographically impossible for the initiator device to compute a token better than level one if other devices refuse to participate. Moreover, prior work either explicitly or implicitly assumes that all phones have a secure environment which is not always the case. By secret-sharing the secret, CoRA allows for flexibility of choosing any of the user's nearby devices with enough computational power to initiate 2FA even when the device does not have a secure element, which is the case for most IoT devices.

The frictionless authentication system of [18] also proposes the use of multiple devices and threshold RSA signatures to authenticate users. CoRA improves and enhances their approach in several ways: 1) it enables the use of threshold MACs (using only AES) which results in more efficient and widely available implementations 2) it introduces multiple assurance levels, each of which is tied to a different secret-shared key, 3) proposes a mechanism for adding new devices and levels which requires careful adjustment of key generation and sharing for each cryptographic instantiation. Finally, to the best of our knowledge, CoRA provides the first implementation of such a multi-device collaborative scheme with detailed security and performance analysis. Moreover, CoRA could also be used as a local authenticator in other 2FA frameworks such as prompt based 2FA (e.g., [9, 2]) and FIDO [11] replacing user's approval on local trusted device.

8 Conclusion & Future Work

CoRA offers a collaborative authentication mechanism through the use of threshold cryptography and anomaly detection. CoRA allows for a distributed MFA scheme where the application layer can choose the balance for security and usability. The token collaboratively generated by the devices is always accompanied by an assurance level that assists the authenticator with access control and granting the right level of privilege to the user. While by increasing the number of participating devices in MFA, CoRA has improved security, it maintains user privacy in the sense that it does not send any extra information to the backend except for the abstract assurance level. CoRA's fraud detection could

be improved by taking advantage of the authentication history of the initiator, such as the timing, frequency, and outcomes of past authentications.

References

[1] Mohammed Ali Al-Garadi, Amr Mohamed, Abdulla Al-Ali, Xiaojiang Du, and Mohsen Guizani. A survey of machine and deep learning methods for internet of things (iot) security. *arXiv preprint arXiv:1807.11023*, 2018.

[2] Authy. Authy onetouch – modern authentication for any application. Available online at `https://authy.com/blog/authy-onetouch-modern-authentication-for-any-application/`, 2020.

[3] Elaine Barker. Nist special publication 800–57 part 1, revision 4, 2016.

[4] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography: Public Key Cryptography*, PKC '03, pages 31–46, London, UK, UK, 2003. Springer-Verlag.

[5] Certicom Research. Sec 2: Recommended Elliptic Curve Domain Parameters. Available online at `http://www.secg.org/sec2-v2.pdf`, 2010.

[6] Mark D Corner and Brian D Noble. Zero-interaction authentication. In *Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 1–11. ACM, 2002.

[7] Alexei Czeskis, Michael Dietz, Tadayoshi Kohno, Dan Wallach, and Dirk Balfanz. Strengthening user authentication through opportunistic cryptographic identity assertions. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 404–414. ACM, 2012.

[8] Rohan Doshi, Noah Apthorpe, and Nick Feamster. Machine learning DDoS detection for consumer internet of things devices. *arXiv preprint arXiv:1804.04159*, 2018.

[9] Duo. Duo push to start. Available online at `https://duo.com/product/multi-factor-authentication-mfa/authentication-methods/duo-push`, 2020.

[10] EMVCo. 3d secure v2.0. `https://www.emvco.com/emv-technologies/3d-secure/`, 2017.

[11] FIDO Alliance. How fido works. Available online at `https://fidoalliance.org/how-fido-works/`, 2020.

[12] Tzipora Halevi, Di Ma, Nitesh Saxena, and Tuo Xiang. Secure proximity detection for nfc devices based on ambient sensor data. In *European Symposium on Research in Computer Security*, pages 379–396. Springer, 2012.

[13] Juniper Research. Online payment fraud whitepaper. Available online at `https://www.experian.com/assets/decision-analytics/white-papers/juniper-research-online-payment-fraud-wp-2016.pdf`, 2016.

[14] Sepandar D Kamvar, Mario T Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651. ACM, 2003.

[15] Nikolaos Karapanos, Claudio Marforio, Claudio Soriente, and Srdjan Capkun. Sound-Proof: Usable two-factor authentication based on ambient sound. In *USENIX Security Symposium*, pages 483–498, 2015.

[16] Soo-Yeon Lee, Sa-rang Wi, Eunil Seo, Jun-Kwon Jung, and Tai-Myoung Chung. Profiot: Abnormal behavior profiling (abp) of iot devices based on a machine learning approach. In *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, pages 1–6. IEEE, 2017.

[17] Claudio Marforio, Nikolaos Karapanos, Claudio Soriente, Kari Kostiainen, and Srdjan Capkun. Smartphones as practical and secure location verification tokens for payments. In *NDSS*, 2014.

[18] Mustafa A Mustafa, Aysajan Abidin, and Enrique Argones Rúa. Frictionless authentication system: Security & privacy analysis and potential solutions. *arXiv preprint arXiv:1802.07231*, 2018.

[19] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and kdcs. In *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'99, pages 327–346, Berlin, Heidelberg, 1999. Springer-Verlag.

[20] Mehdi Nobakht, Vijay Sivaraman, and Roksana Boreli. A host-based intrusion detection and mitigation framework for smart home iot using openflow. In *Availability, Reliability and Security (ARES), 2016 11th International Conference on*, pages 147–156. IEEE, 2016.

[21] Shahid Raza, Linus Wallgren, and Thiemo Voigt. SVELTE: Real-time intrusion detection in the internet of things. *Ad hoc networks*, 11(8):2661–2674, 2013.

[22] Victor Shoup. Practical threshold signatures. In *Proceedings of the 19th International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'00, pages 207–220, Berlin, Heidelberg, 2000. Springer-Verlag.

[23] Babins Shrestha, Nitesh Saxena, Hien Thi Thu Truong, and N Asokan. Drone to the rescue: Relay-resilient authentication using ambient multi-sensing. In *International Conference on Financial Cryptography and Data Security*, pages 349–364. Springer, 2014.

[24] Babins Shrestha, Maliheh Shirvanian, Prakash Shrestha, and Nitesh Saxena. The sounds of the phones: dangers of zero-effort second factor login based on ambient audio. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 908–919. ACM, 2016.

[25] Prakash Shrestha and Nitesh Saxena. Listening watch: Wearable two-factor authentication using speech signals resilient to near-far attacks. In *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 99–110. ACM, 2018.

[26] Nigel P. Smart. Algorithms, Key Size and Protocols Report, H2020-ICT-2014 – Project 645421, ECRYPT – CSA. Available online at `http://www.ecrypt.eu.org/csa/documents/D5.4-FinalAlgKeySizeProt.pdf`, 2018.

[27] Hien Thi Thu Truong, Xiang Gao, Babins Shrestha, Nitesh Saxena, N Asokan, and Petteri Nurmi. Comparing and fusing different sensor modalities for relay attack resistance in zero-interaction authentication. In *Pervasive Computing and Communications (PerCom), 2014 IEEE International Conference on*, pages 163–171. IEEE, 2014.

[28] Hien Thi Thu Truong, Xiang Gao, Babins Shrestha, Nitesh Saxena, N Asokan, and Petteri Nurmi. Using contextual co-presence to strengthen zero-interaction authentication: Design, integration and usability. *Pervasive and Mobile Computing*, 16:187–204, 2015.

[29] VISA and PYMNTS.com. How will we pay – A week in the life of a connected consumer. Available online at `https://www.pymnts.com/news/payments-innovation/2018/pymnts-visa-study-on-how-connected-devices-and-voice-change-how-and-who-consumers-pay/`, 2018.

[30] Catherine S Weir, Gary Douglas, Martin Carruthers, and Mervyn Jack. User perceptions of security, convenience and usability for ebanking authentication tokens. *Computers & Security*, 28(1-2):47–62, 2009.

[31] World Wide Web Consortium (W3C). Webauthn. `https://www.w3.org/TR/webauthn`, 2021.

[32] Bruno Bogaz Zarpelão, Rodrigo Sanches Miani, Cláudio Toshio Kawakani, and Sean Carlisto de Alvarenga. A survey of intrusion detection in internet of things. *Journal of Network and Computer Applications*, 84:25–37, 2017.

Appendix

A Choosing the CoRA Building Blocks

For both threshold authentication and TrustMeter functionality, the research literature provides a variety of options. We compare several of them here and provide a rationale for our implementation choices.

A.1 Choice of Threshold Authentication

In Section 3.4, we defined the notion of a (non-interactive) threshold authentication scheme that covers both threshold MACs and signatures. There are several ways to instantiate them in literature.

First we look at the instantiations of threshold MACs. A pseudo-random function (PRF) is a special kind of MAC where authenticated messages look like random strings. Naor et al. [19] introduced the notion of threshold pseudo-random functions (PRFs) in 1999. They constructed threshold PRFs (or MACs) in two different ways, one based on cyclic groups where the the Decisional Diffie-Hellman (DDH) assumption is believed to hold and another based on any standard PRF. Similarly, threshold versions of digital signature schemes like DSS, RSA, BLS, and ECDSA have been proposed. We will look at threshold versions of RSA [22] and BLS [4] in more detail. We choose these two schemes because their signature generation process is non-interactive (see *).

We discuss four different threshold authentication schemes as potential candidates for CoRA:

- A threshold MAC based on any PRF [19]. We will use AES as the PRF.
- A threshold MAC based on the hardness of DDH assumption in cyclic groups [19].
- A threshold version of RSA due to Shoup [22].
- A threshold version of BLS signatures, based on the hardness of the Gap-Diffie-Hellman [4].

We will use `AES-MAC`, `DDH-MAC`, `RSA-Sig` and `BLS-Sig`, respectively, as a shorthand for them. Each of these schemes have their pros and cons. However, our investigation concludes that `AES-MAC` is the best fit for CoRA.

`AES-MAC` Naor et al. [19] proposed a simple threshold MAC construction based on any pseudo-random function (PRF). Let f be a PRF, n the total number of devices, and t the threshold.

- `Setup`: The set of n devices is divided into subsets of size $n - t + 1$ and a PRF key is associated with each one of them. Let U denote the entire collection of $\binom{n}{n-t+1}$ keys. U is the master secret and the verification key. A device is given PRF keys for all the subsets to which it belongs, so every

device gets $\binom{n-1}{n-t}$ keys. In other words, a key share of a device is defined by a set of $\binom{n-1}{n-t}$ PRF keys.

- `GenPartTok`: Given a message m, a device will evaluate f on m with *every* PRF key it has. The partial token is the set of all evaluations.
- `Combine`: A MAC on a message m is defined to be the XOR of all $f_k(m)$ for $k \in U$. Suppose we have partial tokens from a set S of at least t devices. Observe that every subset of size $n-t+1$ must intersect with S, so for any $k \in U$, there is at least one device in S that had k. Therefore, from the evaluations sent back, one can eliminate the duplicates and XOR the rest to compute the MAC (the full token).
- `Verify`: Given the set of all keys U, one can easily verify a (full) token by recomputing it.

Consider any subset S' of $t-1$ devices. There must be at least one subset of size $n-t+1$ that does not intersect with S'. So there is at least one key in U, say $k^\star$, that none of the devices in S' know about. As a result, S' cannot predict the output of $f_{k^\star}(m)$. Thus, it'd fail to compute the MAC on m.

We can instantiate the PRF f in the above construction with AES. It has a fixed block size of 128 bits, i.e. the inputs and outputs are 128 bits. Key size could be one of 128, 192 or 256 bits depending on the level of security desired.

Other schemes Due to space constraints, we discuss the other three candidates (`DDH-MAC`, `RSA-Sig` and `BLS-Sig`) in the full version.

Comparison We compare these schemes based on computational overhead (for generating a token share and for combining shares), storage and communication overhead (for key shares, for token shares, and for tokens), and security properties (ease of adding new devices, security against Authentication Server compromise). For 128 bits of security, NIST [3] and ECRYPT-CSA [26] recommend key-size of 128 bits for AES, 3072 bits for RSA and 256 bits for ECC (elliptic-curve cryptography), which can be used for `DDH-MAC` and `BLS-Sig`. We will assume the maximum number of devices to be six because a large percentage of the population own six or fewer devices according to [29], and about 70% of the participants of our survey have less than six devices available when authenticating from the main login location (home). We will instantiate both `DDH-MAC` and `BLS-Sig` in elliptic-curve groups of prime-order. Specifically, we use the curve `secp256k1` [5].

Table A.1 provides a summary of our evaluation of the four schemes. In the table, a *yes* value for *Verifier compromise* means that the scheme is *insecure* when the Authentication Server is compromised. Except for `AES-MAC`, the threshold value affects only the combination time of schemes. We choose $t = 3$ because three is an intermediate threshold value, it's the worst case scenario for `AES-MAC`, and other schemes will only perform worse at higher values of threshold. Please refer to the full version for a detailed comparison.

Table A.1: A comparison of threshold MAC and signature schemes. Token is either a MAC or a signature depending on the scheme. Computation times were measured on a MacBook Pro laptop with 2.9 GHz Intel Core i7 processor and 16 GB memory.

Metrics	`AES-MAC`	`DDH-MAC`	`RSA-Sig`	`BLS-Sig`
	Time (ms)			
Generate share	0.1	1.6	5.7	1.6
Combine shares	0‡	4.8	28.5	4.8
	Object size (bits)			
One key share	1280	256	3072	256
One token share	1280	256	3072	256
One full token	128	256	3072	256
	Properties			
Adding devices	easy	easy	difficult	difficult
Verifier compromise	yes	yes	no	no

‡ *The combine operation for* `AES-MAC` *involves only the elimination of duplicates and XOR'ing, so the time taken is negligible compared to others.*

Table A.2: A comparison of IoT anomaly detection schemes.

Detector	**Model**	**Threat**	**Accuracy**
SVELTE [21]	distributed	routing	90–100%
ML-IoT [8]	centralized	DDoS	99.9%
IoT-IDM [20]	centralized	device	96.2–100%
ProFiOt [16]	centralized	device	93.7%

From the table it is clear that `AES-MAC` is much faster at share generation and combination. The size of key and token shares could be a little bit worse, but full tokens are smallest. Adding devices is also easy for `AES-MAC`. The main downside is that if the verifier is compromised, then the master key can be stolen. However, we believe that the benefits of using `AES-MAC` outweigh the negative.

A.2 Choice of Anomaly Detection Scheme

CoRA takes advantage of lightweight anomaly detection techniques to build the TrustMeter. Each device develops an opinion about its peers based on the information it gathers during peer interactions. There is a rich body of work on distributed intrusion detection and more specifically intrusion detection for IoT devices [32, 1, 21, 8, 20, 16].

Table A.2 lists four anomaly detection schemes designed for IoT systems.

SVELTE [21] is a real-time intrusion detection for the Internet of Things that detects network attacks by passively observing network traffic. This work shows that their intrusion detection is light enough to run on IoT devices. The recent work of Doshi et al. [8] detects DDoS attacks with high accuracy but in a centralized manner. ProFiot [16] detects device compromise by analyzing sensor data. IoT-IDM [20] detects malicious activity of devices by analyzing their network traffic in a centralized manner using one device to monitor and analyze traffic. For CoRA requirements, a network-based distributed intrusion detection scheme works best. Therefore, we find SVELTE [21] the most suitable option.

While this paper does not aim to invent a new anomaly detection scheme, we envision features such as packet size, destination, source, number of bytes received per second, number of authentication requests, and time of requests as inputs to the TrustMeter local anomaly detection. These parameters are readily available each time a device interacts with any other device and have proven useful in existing IoT anomaly detectors.

A.3 Additional Charts & Figures

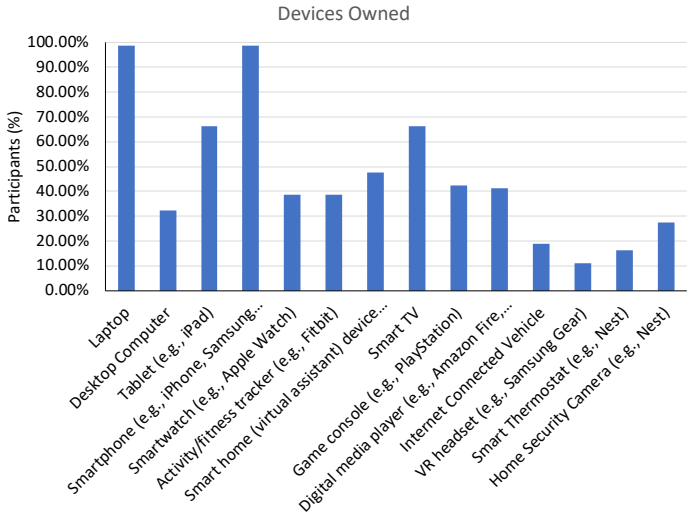


Figure A.1: Smart devices owned by the participants

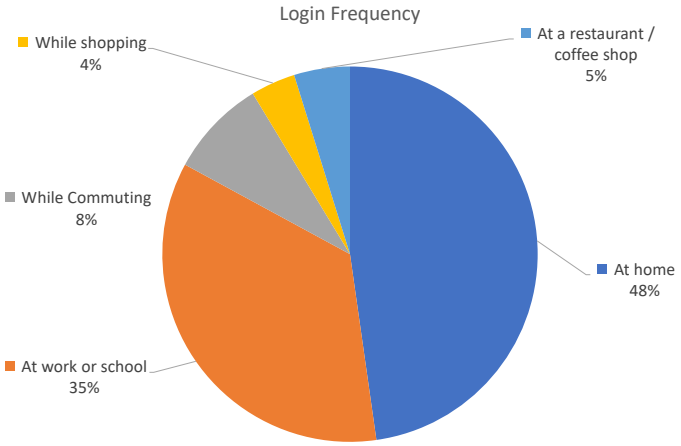


Figure A.2: Fraction of logins from different locations

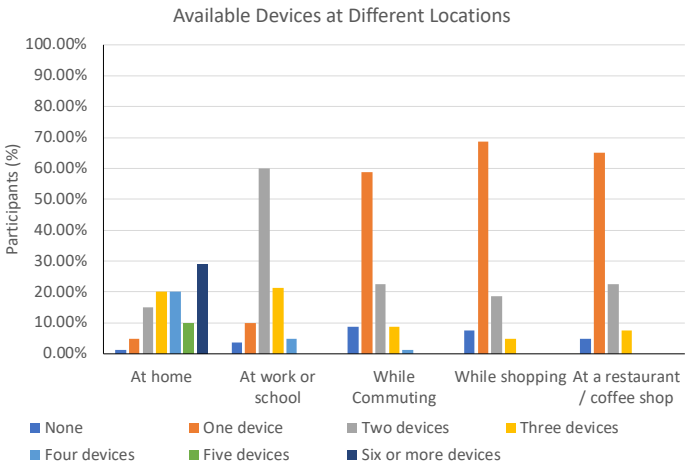


Figure A.3: Devices available when logging in from different locations

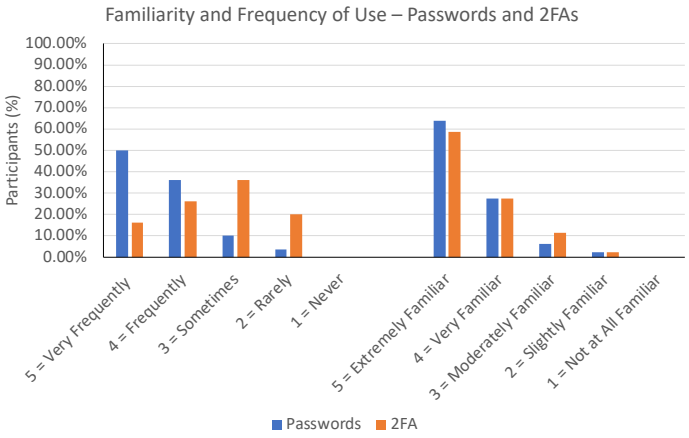


Figure A.4: Participants' familiarity with passwords and second factors

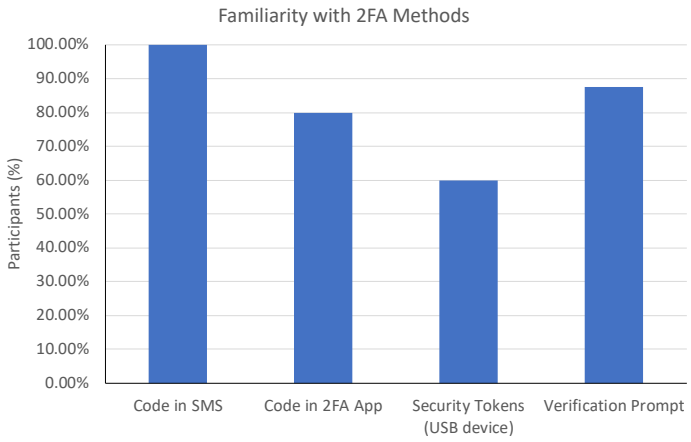


Figure A.5: Participants' familiarity with different 2FA methods

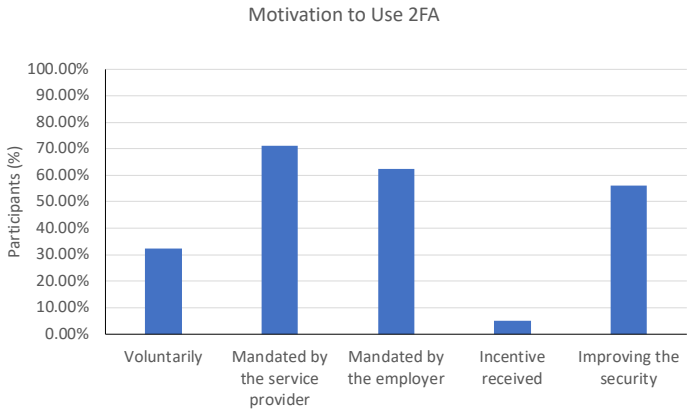


Figure A.6: Participants' reasons and motivation for using 2FA

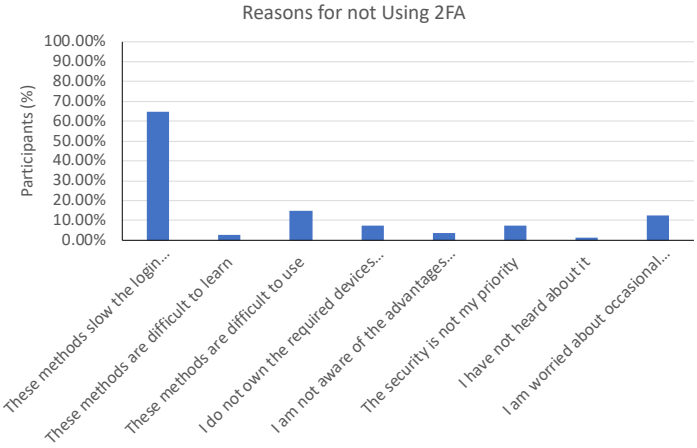


Figure A.7: Participants' reasons for not adopting 2FA

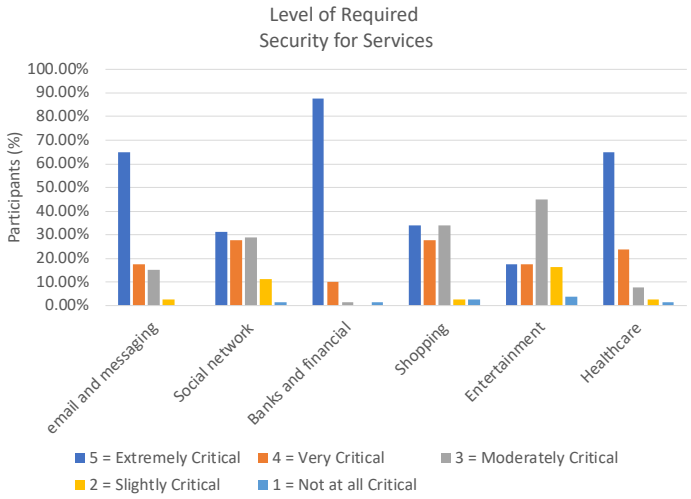


Figure A.8: The level of security required as per the participants' opinion for different services

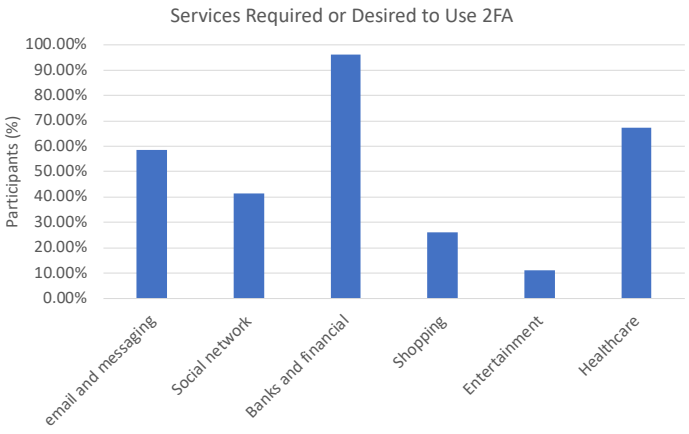


Figure A.9: Services that can benefit from 2FA as per the participants' opinion