

IBM Research Report

Usable Multi-Factor Authentication and Risk-Based Authorization

**Larry Koved, Pau-Chen Cheng, Diogo Marques*, Nalini Ratha,
Kapil Singh, Cal Swart, Shari Trewin**

IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598 USA

*University of Lisbon
Portugal



Usable Multi-Factor Authentication and Risk-Based Authorization

Project Final Technical Report

Funded by the United States Department of Homeland Security
Award Number: FA8750-12-C-0265

Principal Investigator:

Larry Koved
IBM T. J. Watson Research Center
P.O. Box 218
Yorktown Heights, New York 10598

Technical Contributors:

Pau-Chen Cheng
Larry Koved
Diogo Marques
Nalini Ratha
Kapil Singh
Cal Swart
Shari Trewin

Table of Contents

.....	i
List of Figures and Tables	viii
Acknowledgements	x
1.0 Summary.....	1
2.0 Introduction	4
3.0 Risk Perception and Communication	7
3.1 Methods	7
3.2 Assumptions	7
3.3 Procedures.....	8
3.4 Results and Discussion.....	12
3.5 Conclusions	18
3.6 Recommendations	19
4.0 Human Computer Interaction	21
4.1 Methods	21
4.2 Assumptions	21
4.3 Procedures.....	22
4.4 Results and Discussion.....	23
4.5 Conclusions	25
4.6 Recommendations	26
5.0 Biometric Multi-Factor Authentication	27
5.1 Methods	28
5.2 Assumptions	30
5.3 Procedures.....	31

5.4 Results and Discussion.....	33
5.5 Conclusions	40
5.6 Recommendations	40
6.0 Risk-Based Authorization.....	41
6.1 Methods	42
6.1.1 Risk, Uncertainty and Time	42
6.1.2 Uncertainty.....	43
6.1.2.1 Where Is Uncertainty?.....	43
6.1.2.2 Reasons to Account for Uncertainty.....	43
6.1.2.3 Handling/Modeling Uncertainty	44
6.1.3 On Accuracy of Risk Estimate.....	44
6.2 Assumptions	44
6.3 Procedures.....	45
6.3.1 Handling Uncertainty.....	45
6.3.1.1 Handling Normal Fluctuations, Inaccuracy and Incompleteness	45
6.3.1.2 Handling Possibilities Not Covered by Past Data and Experience	45
6.3.1.2.1 A Performance Optimization	49
6.3.2 Estimating Risk	50
6.3.2.1 Bayesian Network Overview	50
6.3.2.3 Using A Bayesian Network Node To Merge Uncertainty	53
6.3.2.4 Configuring a Bayesian Network Node	54
6.3.2.5 Example: Configuring the CPT of the <i>Location Anomaly</i> BN Node	56
6.3.2.6 Example: Configuring the CPT of the Risk BN Node	58
6.3.3 Trust-Value-Risk Based Access Control Policy.....	60

6.3.5	Continuous Learning of Behavior Profiles	66
6.3.5.1	Recognizing the Signs of Emerging New Behavior Patterns.....	67
6.3.5.2	Bridging the Gap Between Human and Machine	70
6.3.5.3	Experimental Validation of the Learning and Scoring Mechanisms	71
6.4	Results and Discussion.....	76
6.5	Conclusions	76
6.6	Recommendations	77
7.0	Mobile Client and Security Services.....	78
7.1	Methods	78
7.2	Assumptions	79
7.2.1	Hybrid Application Design	79
7.2.2	Context-based Security.....	80
7.2.3	Flexible Integration Options	81
7.3	Procedures.....	87
7.4	Results and Discussion.....	89
7.4.1	Core Client-side Components	89
7.4.1.1	Native Plugins and Wrappers.....	89
7.4.1.2	Hybrid layer.....	91
7.4.2	Representative Business Application: Banking	94
7.4.3	Transition to Practice: Integration with ISAM	95
7.4.3.1	Multi-factor authentication for web apps.....	96
7.5	Conclusions	97
7.6	Recommendations	97
8.0	Network Authentication and Authorization Services.....	99

8.1	Methods	99
8.2	Assumptions	101
8.2.1	Integration with network authentication / authorization services and network services providers	102
8.2.1.1	Reverse Proxy	103
8.2.1.2	Mobile Application Service	103
8.2.2	Secure operation and efficient communication protocols	104
8.2.2.1	Secure operation	104
8.2.2.1.1	Network and protocol security	104
8.2.2.1.2	Mobile device security	106
8.2.2.1.3	NAAS and biometrics security	106
8.2.2.2	Efficient communication protocols	107
8.2.3	Inline / out of band processing	112
8.2.4	Biometrics – enrollment and verification	114
8.2.5	Risk Authorization	115
8.2.6	Authentication Challenges	115
8.2.7	Risk Communication	116
8.2.8	Administration	116
8.3	Procedures	116
8.4	Results and Discussion	117
8.4.1	Shim layer	117
8.4.1.1	EAI	118
8.4.1.2	Access	119
8.4.1.3	Login	119
8.4.1.4	Logout	119

8.4.1.5	RegisterUser / RegisterDevice	120
8.4.1.6	PushNotification	120
8.4.2	Core authentication and authorization services	120
8.4.2.1	ResourceAccess2	121
8.4.2.1.1	Predictive scheduling of challenges	124
8.4.3	Authenticators and Context / Risk Evaluators	124
8.4.3.1	Authentication Verifier	124
8.4.3.2	Authentication Enrollers	126
8.4.3.3	Context Evaluators	126
8.4.4	Biometric fusion	127
8.4.5	Authorization policy	127
8.4.6	Administrative functions	129
8.4.6.1	Runtime configurable parameters	129
8.4.6.2	Registries	130
8.4.6.3	Authentication and context / risk evaluators	130
8.5	Conclusions	131
9.0	References	132
9.1	Risk Perception and Communication	132
9.2	Human-Computer Interaction	133
9.3	Biometrics	133
9.4	Risk-Based Access Control	134
9.5	Mobile Client and Security Services	136
9.6	Network Authentication and Authorization Services	137
10.0	Symbols, Abbreviations and Acronyms	138

11.0	Glossary of Terminology	140
------	-------------------------------	-----

LIST OF FIGURES AND TABLES

Figure 3.1. Authentication screen presented in Study 4	11
Figure 3.2. Proportion of participants choosing to authenticate in Study 3.....	14
Figure 3.4. Participants' preferred level of risk information in Study 4	17
Figure 5.1 Biometric elements of the system	27
Figure 5.2 System Architecture	28
Figure 5.3 Flow for biometric acquisition and scoring.....	29
Figure 5.4 FbF mobileOne QuickDock	31
Figure 5.5 Preventing Replay Attacks Through Information Hiding	33
Figure 5.6 Multi-biometrics Web Interface	34
Figure 5.7 Multi-biometrics Enrollment	35
Figure 5.8 Multi-biometrics Verification	36
Figure 5.9 Delete user	37
Figure 5.10 Fingerprint Verification Sample Response	37
Figure 5.11 Face Verification Sample Response	38
Figure 5.12 Voiceprint Verification Sample Response	38
Figure 5.13 Multi-biometrics Fusion Web Interface and Sample Response	39
Figure 5.14 Fusion ROC	40
Figure 6.1 Beta Distribution	46
Figure 6.2 Sample Beta Distribution Curve	47
Figure 6.3 Conversion of <i>pdf</i> to <i>pmf</i> example.....	49
Figure 6.4 <i>pdf</i> to <i>pmf</i> Conversion with Shifted Quanta	50
Figure 6.5 Risk Estimator Based on Bayesian Network	52
Figure 6.6 Making an Access Control Decision Using the TVR Model	62
Figure 6.7 Multi-Round Decision Making with TVR Policy.....	64
Figure 6.8 Our Implementation of the TVR Policy	65
Figure 6.9 Data Point Cache for Learning Example	68
Figure 6.10 Profile and Cache Scores Over Time	68
Figure 6.11 Newness Map	69
Figure 6.12 Merging Profile Score and Cache Score	70
Figure 6.13 Learning Screen Shot - Before Profile is Learned	73
Figure 6.14 Learning Screen Shot - After Profile is Learned	74
Figure 6.15 Learned Geo Locations and Paths.....	75
Figure 7.1 Development lifecycle for hybrid applications.....	80
Figure 7.2. High-level architecture of out-of-band authentication	83
Figure 7.3. Proxy-less out-of-band authentication architecture with inline communication	85
Figure 7.4. Architecture of inline authentication	86
Figure 7.5. High-level Client-side Architecture	88

Figure 7.6. Representative Integration with ISAM	96
Figure 8.1 Representative Network Control Points	102
Figure 8.2 Unprotected Resource Access Message Flows	109
Figure 8.3 Protected Resource Access Message Flows	110
Figure 8.4 Biometric Authentication Iteration.....	111
Figure 8.5 Reference System Architecture.....	117

ACKNOWLEDGEMENTS

We thank the Department of Homeland Security, which sponsored this work under Award Number FA8750-12-C-0265.

We also thank the management at IBM Research for their support of this work.

1.0 SUMMARY

In this project we focus on creating a set of usable authentication and authorization services that leverage capabilities found in mobile devices, such as smartphones and tablets, that were previously unavailable in traditional desktop and laptop systems. Mobile devices are increasingly being used as a primary platform for computing services, whether for mobile banking, social networking, information gathering, or business / workflow applications. The state of mobile application authentication is to use the mobile device as a security token. Because it is awkward and time consuming to enter strong passwords on mobile devices, the security credentials are being cached in the device. Oftentimes applications store security tokens, such as HTTP cookies, on the device to represent that the owner of the device had successfully authenticated via the mobile device and the transactions using device are to be trusted. There is a presumption that if the user can unlock the mobile device, then it must be the legitimate owner of the device who is using it. As a result, the mobile device is now being authenticated rather than the user. If a device is lost or stolen, an attacker can either use the device to obtain goods or services, or extract the credentials and mount a broader scale attack. Even with devices that have built-in authentication, such as fingerprint scanners, it is not clear to the organization receiving the request from the mobile device as to who is actually making the request. It could be friends, family or strangers. It then becomes a question about liability in the case of loss / theft as to who is to be held responsible. Loss of reputation may be harder to recover than loss of specific assets.

We often perceive that there is a tradeoff between usability and security. Security concepts are often difficult to understand or implementations are awkward to use. The goal for the project has been to achieve a balance between security and usability. Specifically, authenticate just enough to maintain sufficient security, leveraging situation, context and history, while accommodating situational impairments and personal preferences. The project demonstrated that we can use advanced authentication and authorization technologies to achieve this goal.

This project uses situation, context and history to determine when, and how much, authentication confidence is required to perform a requested operation from a mobile device. The situation is the request for a service, including mobile banking, workflow, collaboration, or other network requests. The context is the environmental factors, including information about the device, where the user is located, and other devices or people near the device. History tells us about the user's typical behaviors in order for

us to assess whether there is anomalous behavior that may be indicative of higher risk, such as a lost or stolen device. Based on these factors, policy determines the level of authentication confidence required to be allowed to proceed with a transaction. The policy considers that the inputs into the authorization decision process may be noisy, including the results of biometric authentication, contextual factors and history.

Strong authentication seems like a natural solution in an environment where we don't know who may be requesting a service. We know from the literature and anecdotal evidence that people don't like to enter passwords into their mobile devices. Strong passwords are awkward to enter, typically requiring the transition between multiple keyboard layouts to enter alphanumeric characters and symbols, thus making them both difficult and time consuming to enter. Research has shown that authentication using biometric techniques can be more efficient, time wise, than password entry. The literature also tells us that it is possible to achieve greater authentication confidence when combining the results from multiple biometric authentication modalities, an approach known as biometric fusion. Thus, if we have a system with such support, we can take advantage of the multiple biometric authentication modalities to satisfy the step up authentication requirement. Additionally, multiple factors provide useful options in situations where one or more factors are impaired (e.g., face verification is not feasible in a dark room). In this project we incorporate multi-factor biometric authentication, using biometric fusion, to achieve strong authentication in a usable form. Additionally, by embedding one-time use security tokens in the biometric samples, replay attacks can be thwarted.

In comparison to laptop and desktop computers, mobile devices travel to many more places, with the result that they are more frequently lost or stolen. Mobile devices are often left unattended, enabling both curious and malicious people to use the unattended device, which is unprotected by any strong authentication. Mobile devices have many sensors, including accelerometers and gyroscopes. We use one or more sensors to estimate when the owner of the device is no longer in possession of the device and initiate a lock out to prevent unauthorized use. Feedback from commercial customers was very positive for this feature.

One of the key challenges with mobile devices and security is the lack of visual security indicators that indicate the security state of the application or service with which the user is interacting. Desktop browsers have title bars where security indicators, such as URL and HTTPS are embedded. These indicators are typically lacking in web apps. Given that the risk profile for mobile devices is different than for desktop / laptop systems, we recognize that there are additional security risks that may be relevant to the end user. To align the security interests of the end user with those of the service provider, we need to understand the risk perception gap that exists between the end

user and the service provider and provide means for communicating the risks as perceived by the service provider. We conducted studies to see where security perception did not align with expert opinion and enabled the communication of these risks to the end user.

To create a demonstrable system, we developed mobile device software and hosted network authentication and authorization services to integrate all of the elements of the design. Software development on the mobile platforms is fragmented, with different programming languages and services that are platform dependent. To minimize development cost and maximize platform coverage, we used a hybrid runtime environment that maximized software reuse across platforms and minimized platform-specific software components. In addition, the range of sensors and biometric capabilities is rapidly changing. For example, Apple Touch ID fingerprint verification, and the Samsung equivalent, became available as we were developing the system. To accommodate the rapid integration of new biometric and contextual factors into the system, we designed a modular structure that allows for extensibility.

Network authentication and authorization services tie together all of the system elements – risk, authentication, risk communication, user presence detection, applications and services. There are a variety of deployment scenarios – applications, reverse proxies, network access control, etc. We created a set of extensible services that can be integrated into any of these deployment scenarios and demonstrated the flexibility of our approach by integrating with a representative application and a commercial reverse proxy.

As we were interacting with commercial customers, we realized that there are three key sets of users that must be considered for the technologies we developed to succeed – end-user, developer, and security administrators. The latter two were not originally considered when we proposed the project. To address the developer requirements, we devised multiple approaches for integrating mobile apps, web applications and Internet of Things (IoT) with our mobile and authentication services. We created techniques that leverage mobile device capabilities to ease integration with the services we offer. This includes inline and out-of-band authentication techniques. We also include the ability to separate the authentication on the mobile device from the mobile apps to better protect the authentication credentials.

This report describes the project results. We successfully demonstrate the ability to perform usable strong authentication on mobile devices, considering risk assessment, providing risk communication, using a flexible framework that affords easy integration of authentication into mobile, web and IoT applications.

2.0 INTRODUCTION

This project addressed the problem of usable authentication and authorization technologies to enable a strong tie between physical and transactional identity. In particular, we focused our investigation on the context of mobile platforms such as smartphones and tablets, and applications with high security requirements, such as mobile banking and other transactional services. These platforms pose a particularly challenging use case for usable secure authentication. There is a heightened user expectation due to the new interaction techniques with these devices. However, these devices are prone to loss and their physical characteristics make traditional secure authentication regimes, including userid and password, difficult to use.

We investigated usable multi-factor authentication that leverages built-in cameras, microphones, fingerprint scanners, touch screens, accelerometers, GPS, Bluetooth and other sensors which are generally available in most mobile devices such as smartphones and tablets. We developed a risk based authorization framework to explore the tradeoffs and balance between increased user acceptance, the value-at-risk in mobile banking and other such transactions, and the operating characteristics of the sensors on such devices. We validated our approach by conducting studies to understand end-user perception of risk and how this can be used to influence acceptance of multi-factor authentication. We built a proof of concept system, including client side and server / network side frameworks to demonstrate the key concepts of our approach. The balance of this report is organized into the following distinct, yet related, aspects:

Usable Risk-Based Authentication – This part of the project was divided into two related components: In section 3.0 we discuss *Risk Perception and Communication*, and in section 4.0 we discuss *Human Computer Interaction*. We ran several studies to understand what risks users of mobile devices perceive and compare these perceptions with security experts to identify gaps. By understanding perceived risks, we identified opportunities for communicating risks and benefits to users, with the goal of making design recommendations that would improve the overall usability and security of the system. There were several user interface design iterations, informed by the psychometric studies, usability studies were performed to refine and improve these designs.

Biometric Multi-factor Authentication – In section 5.0 we discuss the use of multiple biometrics to improve confidence in the user identity through techniques that are more usable than traditional authentication techniques such as PINs and passwords. Biometric authentication has a long history in law enforcement. However, use of biometrics in unsupervised environments presents new challenges.

Mobile device users are not biometrics or security experts, thus are not well qualified to decide which biometric is best suited for obtaining the requisite authentication confidence. Since users need to be authenticated in many diverse situations, even cooperative users will find some biometrics to be unusable in many scenarios. For example, in an area with a darker lighting condition, face recognition may not be very useful. Similarly, in a noisy environment, speaker recognition may be challenging. Thus use of multiple biometrics and fusion of their result has been incorporated in our system to address this problem. We also include information hiding in the biometric signal as a means for preventing replay attacks on the system.

Risk-Based Authorization – In section 6.0 we focused on building up predictive risk analytics and conducting iterative tests on the mathematical model we developed to estimate risk and make principled decisions about authorization decisions. This component performed real-time learning of user behaviors and combined the results with a fuzzy logic based approach that enable balancing of end-user acceptance, value-at-risk and the performance of various biometrics via biometric fusion. In this report we lay out the motivation and mathematical foundations of our approach, and an example of real-time learning as one of several inputs into the authorization decision process.

Secure Authentication Frameworks – To build an operational system, there needs to be secure software components on both the mobile client and in the network-based server / service.

In section 7.0 we review the design of the *Mobile Client and Security Services*. To achieve coverage on multiple platforms (Apple iOS and Google Android), a hybrid application development platform was used so that most of the user interaction and security services would be common across all platforms. The services include user interaction (user interface), management of authentication, frameworks to manage authentication and context data collection, and the handling of the various integration options – inline and out of band. Platform-specific functions are written as plug-ins that are called by the platform-independent code.

In section 8.0 we review the design of the Network Authentication and Authorization Services (NAAS). In this section we discuss the representative network access control points, the multiple modes of integrating the mobile authentication into client apps and how that influences the design of the NAAS, the three sets of users of the system – *end user*, *software integrator*, and *security administrator*. Each of these user groups have a set of usability concerns that we addressed in our design. The

NAAS ends up being the focal point for all of the aforementioned work streams, we discuss how they are integrated into the overall system.

Our approach was iterative, which included the development of operational prototypes and evaluating them on mobile devices to understand the strengths and shortcomings of the various authentication and authorization approaches explored in this project. To demonstrate viability of the developed technologies, we integrated the NAAS, and hence the various work streams, into a *reverse proxy* server, IBM's *Security Access Manager for Mobile* (ISAM for Mobile)¹, that acts as an authentication and authorization service for HTTPS-based service requests that are typical of mobile, web and Internet of Things (IoT) applications.

¹ <http://www-03.ibm.com/software/products/en/access-mgr>

3.0 RISK PERCEPTION AND COMMUNICATION

If organizations wish to promote secure behavior in their employees or customers when using mobile devices, it is important to know whether users are aware of the risks. If they are not, then risk communication methods are an appropriate response. If they are already aware of the risks, but choosing not to act, then a different approach is needed. The studies reported here provide information that can support decision-making, by reporting which risks come readily to mind in a set of mobile access scenarios.

The objective of the studies is to contribute to the understanding of perceived information security risks in sensitive mobile transactions. By understanding perceived risks, we identified opportunities for communicating risks and benefits to users, with the goal of making design recommendations that would improve the overall usability and security of the system.

3.1 *Methods*

We carried out two studies of risk perception, using a survey-based format. One (Study 1) was conducted on Amazon Mechanical Turk, and presented participants with personal banking scenarios (the ‘Personal Banking’ group). The second survey (Study 2) was conducted within IBM Research, and presented participants with job-related data access scenarios (the ‘IT Worker’ group). A group of IT Security experts were consulted to provide a baseline set of security risks. The results of these surveys were used to develop a taxonomy of perceived risks, which informed the risk communication design, and was used in the risk communication study that followed.

We carried out two further studies of risk communication, both using Amazon Mechanical Turk. Study 3 presented participants with an authentication decision, and measured the effect of manipulating the stated beneficiary of the authentication on willingness to authenticate, and acceptance of the authentication burden. Study 4 presented alternative designs of risk communication, and probed participants’ preferences for specific or general communication of risk factors detected in the context of an access request.

3.2 *Assumptions*

Studies 1, 3 and 4 relied on the following assumptions:

- Web-based survey questions on perceived risk in imagined scenarios will bring to mind similar risks to the ones people are aware of in real mobile situations.

- US-based experienced Amazon Mechanical Turk workers with high approval ratings are reasonably representative of mobile device users in the US and the UK, and provided honest responses.

3.3 Procedures

The materials and procedures for Studies 1 and 2 were described in detail in separate technical reports from the Principal Investigator: “*Perceived Risk in Mobile Authentication and Interaction*.” and “*Design of Psychometric Studies on Security Risk Perception for Mobile Authentication and Authorization*”. Also see “*Perceptions of Risk in Mobile Transactions*” [Trewin, Swart, Koved and Singh, 2016].

In brief, participants were presented with mobile scenarios describing six locations, and asked whether it would be safe to access a specific kind of information at that location. After considering eighteen such scenarios, participants were asked the following open questions:

- “What else would you want to know about the situations described in this study to decide whether it is safe to access or enter sensitive information on your smartphone there?” Responses to this question reveal factors that the individual would consider when evaluating risk, such as the type of network connection or presence of other people.
- “What, if any, are the security risks you see in these situations?” Responses here indicate the specific threats that the individuals are aware of, such as device theft or network eavesdropping.
- “What factors affect your decision whether to access sensitive information in a given situation?” This question goes beyond risk perception to reveal other factors that people will take into account when deciding whether to accept the perceived risk, such as the urgency of the need to access the information, and ability to go to a safer place.

The locations included a variety of high and low risk situations: familiar and unfamiliar places, different kinds of likely connection method, and different risks of theft or observation.

The materials and procedures for Study 3 are described in detail in the technical report: “An Experiment in Re-Framing Authentication Decisions With Short-Term Benefits”. This study was an online experiment, with a between-subjects design. By including an authentication dialog mid-task, we sought to elicit realistic decision-making. Since users

behave differently when they know that their security behaviors are under scrutiny [Braunstein, Granka and Staddon, 2011; Fahl, Harbach, Acar and Smith, 2013], deception was necessary. The procedure consisted of two consecutive tasks, each a separate HIT (*Human Intelligence Task*). The first task was to complete a short survey. Upon finishing it, participants were assigned a password, instructed to retain it, and invited to participate in the second task. We did not allow participants to choose their own password to prevent them re-using a valid one, which would constitute a risk. All participants were assigned the same pre-generated password, “mia8h”. The second task started with an authentication request. Participants were randomly assigned to one of four conditions, with the following prompts:

- Please enter your password. This allows us to combine your responses from the two studies. [benefit to a third-party]
- Please enter your password. This allows you to skip the questions you already responded to. [benefit to user]
- Please enter your password. This allows us to combine your responses from the two studies. It also allows you to skip the questions you already responded to. [benefit to both]
- Please enter your password. [control]

Participants could either insert a password and click “OK”, or click a prominently displayed button with the text “Skip entering password”. In both cases, this led to an exit survey. The main response variable was the binary decision to either authenticate or skip. Additional measurements were collected to help explain the possible outcomes.

The next study, Study 4, carried out as a task in Amazon Mechanical Turk, presented 79 participants with a scenario in which their friend had started using an email or banking service with mobile risk-based, multi-factor biometric authentication. Participants were shown the main authentication screen, where users can select their preferred authentication method. The screen contained a risk icon, an authentication prompt, and a set of authentication buttons, as shown in Figure 3.1. After asking questions about perceived risk in the scenario, participants were presented with three alternative versions of the authentication user interface. These differed only in the design of the authentication prompt, which was varied to provide no risk information, general information, and specific information. Specifically:

- [No risk information] Are you really Dina? Choose one of the authentication methods below.

- [General risk information] Something seems different. Are you really Dina? Choose one of the authentication methods below.
- [Specific risk information] You are not usually at this location. Are you really Dina? Choose one of the authentication methods below.

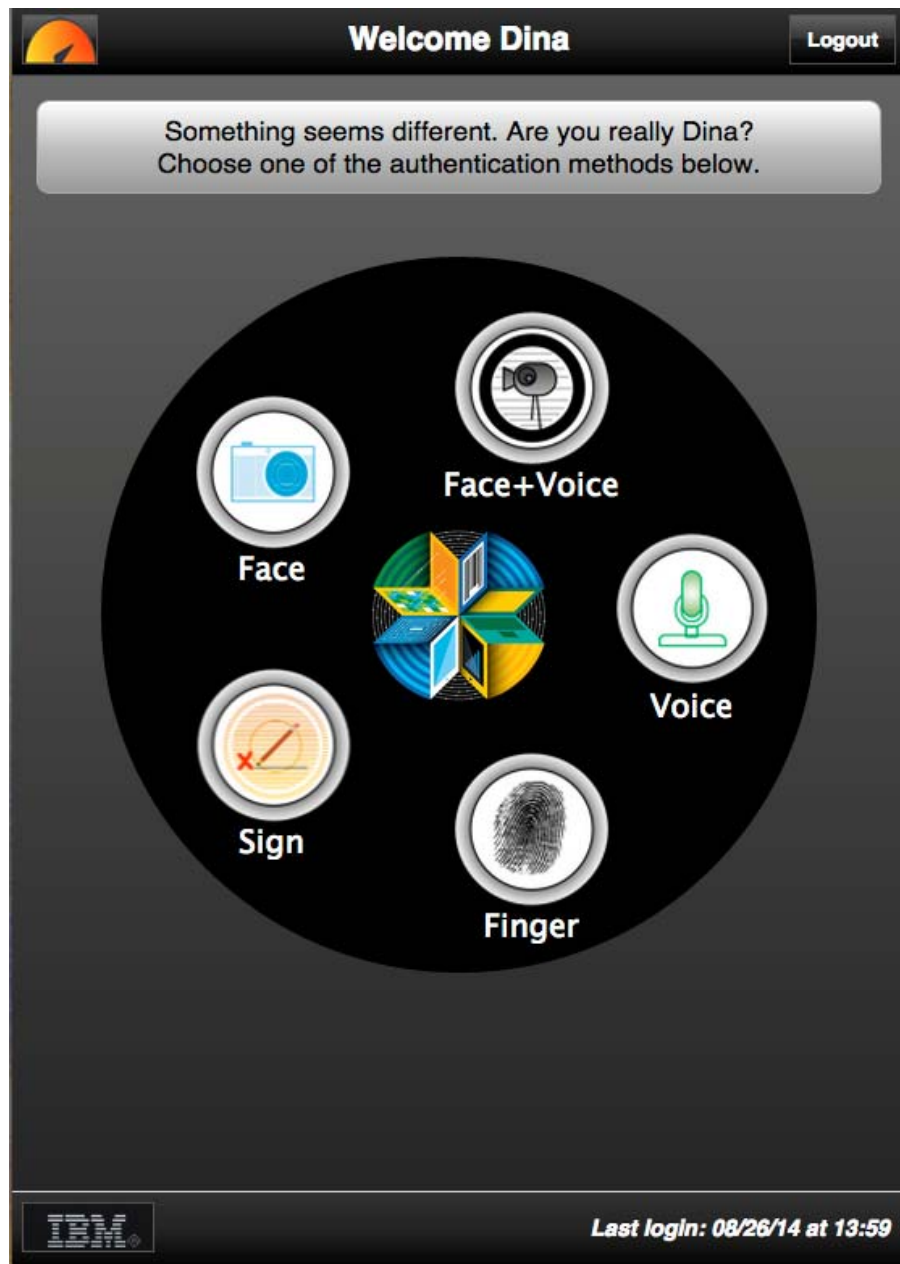


Figure 3.1. Authentication screen presented in Study 4

This study explored whether participants preferred to see more or less information about the risk. We also repeated the open question “What, if any, are the security risks

you see in these situations?” used in Studies 1 and 2, and asked about perceived security of the system, and participants’ willingness to provide their location, necessary for risk assessment.

3.4 Results and Discussion

In Studies 1 and 2, we found that the dangers of mobile transactions (as expressed by a group of IT Security Experts) did not easily come to mind in the IT Worker and Personal Banking scenarios that were presented. IT Workers had higher overall awareness of risk and were more likely to lock their phones. 68% of IT Workers, and 41% of Personal Banking participants used a lock on their smartphones.

The most commonly cited general classes of risks were shoulder surfing and network snooping. Respondents indicated that they consider these threats, their current situation, and the people around when deciding to perform a mobile transaction. Device theft and loss, hacking and malware were not prominent concerns. Unlike security experts, participants did not express concern over data stored on devices, and were less specific about the important factors around network security.

Table 3.1 summarizes the risks identified by each study group, and the percentage of respondents in each group who identified each type of risk.

Locations with close proximity of other people (busy café, crowded street) were perceived as less safe, with 2 out of 3 participants reporting being aware of the risk of being observed, but only 1 in 3 saying this factor would influence their decision whether to perform a transaction. Some users reported taking steps to reduce shoulder surfing risk, and it was seen by many as a controllable risk.

In Study 3, reported here, we found that perception of short-term benefits doesn't improve acceptance of requests. In general, participants did not find the authentication request overly annoying ($M = 6.4$, $SD = 3.4$, 10: “not annoying at all”) or unreasonable ($M = 7.1$, $SD = 2.9$, 10: “very reasonable”), although there was considerable dispersion. For both the annoyance ($F = 1.23$, $p = 0.30$) and reasonability ($F = 0.782$, $p = 0.51$) scales, there were also no statistically significant differences across conditions. Figure 3.2 shows the proportion of participants who chose to authenticate in the four different prompt conditions.

Table 3.1: Categories of security risk perceived in the scenarios by IT Workers (IT), and Personal Banking consumers (PB), showing percentage of each group identifying at least one risk in that category

Type of Risk	Description	IT	PB
Network Risks	Risks encompassing ways that information could be captured en route to a destination	62	51
Observation Risks	Information or passwords being observed while a device is being used.	60	28
Device Risks	Loss, theft, or otherwise obtaining data or login credentials directly from the device itself	52	13
Remote Service Risks	Risks related to the service being accessed (specifically referencing the unknown retailer)	26	23
Loss of Information	Risk of information being lost or account access credentials being obtained by a third party	19	21
Situational Risks	Risks associated with the personal safety of the situation	10	6

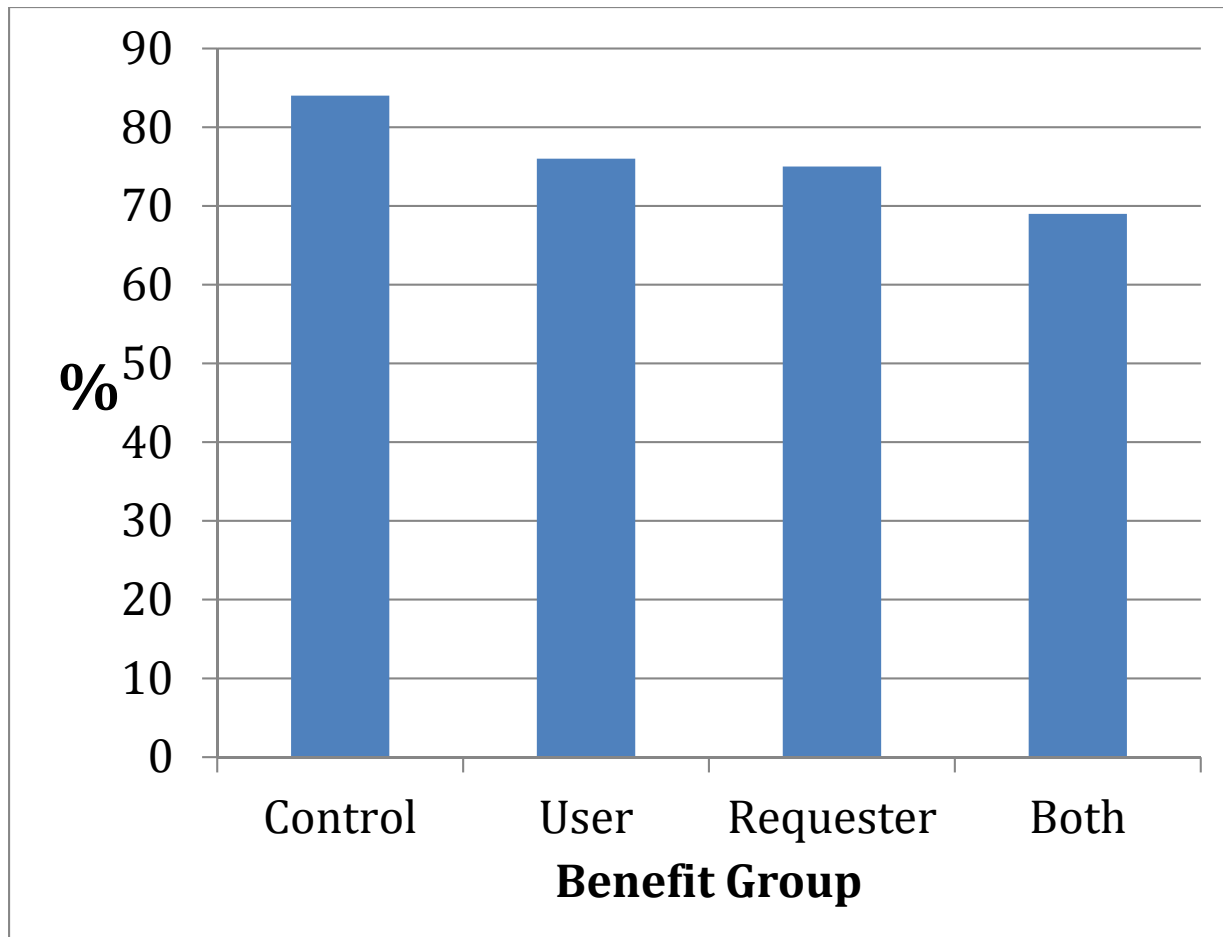


Figure 3.2. Proportion of participants choosing to authenticate in Study 3

Overall, there is no evidence that the proportion of participants who decided to authenticate is significantly different among groups ($\chi^2 = 5.86$, d.f. = 3, $p = 0.12$). Since the experiment was powered to detect an effect size of 0.25 or higher, we can say that if an effect exists, it is most likely not large (indeed, Cramer's $V = 0.18$). Communicating a benefit to the user versus the requester resulted in approximately the same acceptance rate. Comparing to the control, it resulted in a reduction of the acceptance rate, but that difference cannot be generalized ($\chi^2 = 1.61$, d.f. = 1, $p = 0.20$).

Frustration and non-compliance arose most often because the user did not have their password. At the point of authentication, drawing attention to the benefits did not impact this frustration. It is possible that, as others have found, participants don't pay much attention to the prompts [Bravo-Lillo, Crabor, Downs, and Komanduri, 2011; Egelman,

Cranor, Hong, Egelman and Hong, 2008; Felt, Reeder, Almuhiemedi and Consolvo, 2014]. Interestingly, those who were more concerned about privacy and more confident in their security knowledge may have made a greater effort to retain and use their passwords, leading to greater acceptance.

In Study 4, reported here, we found a high level of perceived security of the multi-factor biometric authentication for a bank account (mean response 7.4 out of 10, Std. Dev. 2.6). 58% of participants said they would allow the bank to track their location, 29% would not, and 13% were unsure. Participants reported similar sources of risk to those in Studies 1 and 2, summarized in Figure 3.3. The café scenario was considered to have a risk level of 6/10, while the home scenario was rated 3/10.

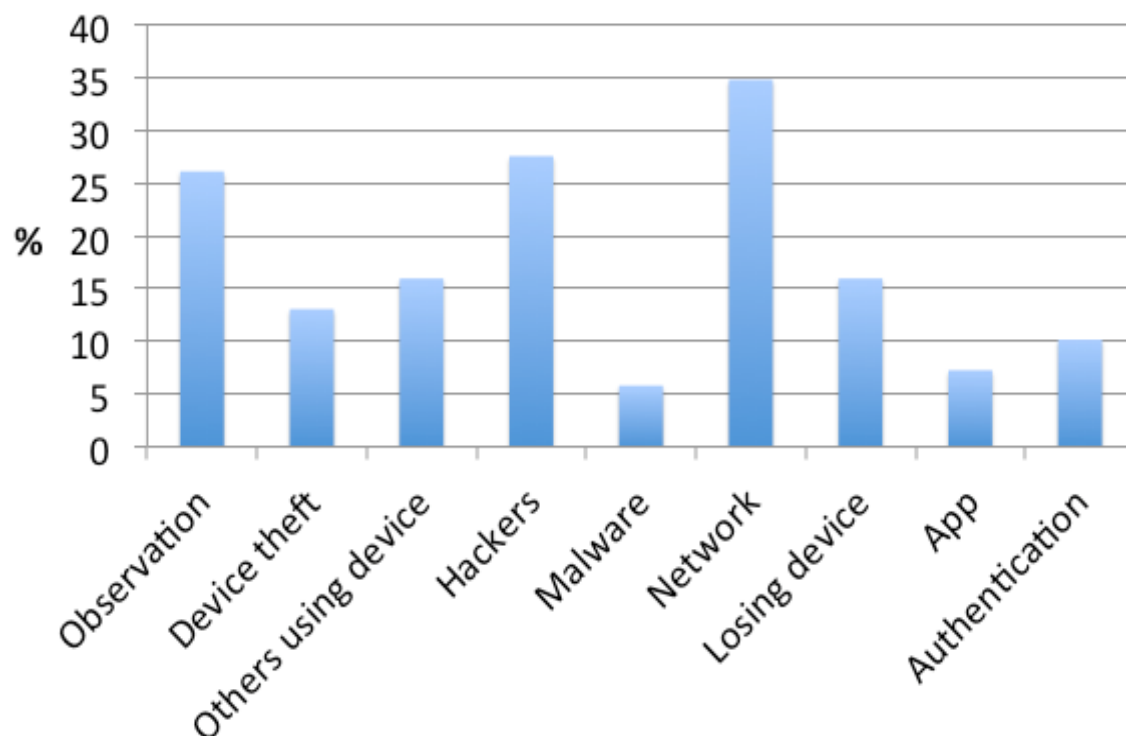


Figure 3.3. Risks identified in Study 4

One difference from the risks identified in Studies 1 and 2 is the new category 'Others using device'. This category encompasses statements about other people physically using the device, but without specifying loss or theft. It includes insider attacks and

family members. Example statements in this category are *“I think that if someone gets ahold of your smartphone they may be easily able to access your information.”*, *“Also, it may increase the rate of kidnapping, or phone jacking, making people log into their phones so the criminals can easily drain the account.”*, and *“If someone else uses your phone while you are still accessed to your bank account.”*

A second notable difference in this study was the blending of network and digital device attacks into a single concern about ‘hackers’. Whereas responses in studies 1 and 2 were coded as either network or device concerns, it was not always possible to make this distinction in Study 4’s responses, leading to the creation of a single ‘Hackers’ category. Example statements include *“Hackers are good at what they do.”*, *“Smartphones are much easier to hack.”*, *“I see that hackers can duplicate data, possibly copying the user’s information and try to enter it.”*

Thirdly, seven participants suggested security risks related to the multi-factor authentication method itself, leading to a new ‘Authentication’ category of risk. For example, *“If you are using a system that makes it easy for you, then it may be easy for anyone else.”*, *“the app making an error in authentication”*, and *“If someone got their hands on her phone, and found a picture of her on her phone they could easily use that to get into her bank account.”*

Table 3.2 compares the percentage of people in Study 1 (Personal Banking group) mentioning each source of perceived risk with the findings of Study 4 (Personal banking scenario with multi-factor biometric authentication). Observation risks remained similar, device risks were more frequently mentioned, and network concerns were lower.

Table 3.2. Comparison of risks reported in Study 1 and Study 4

Type of Risk	Description	Study 1	Study 4
Network Risks	Risks encompassing ways that information could be captured en route to a destination.	51	35
Observation Risks	Information or passwords being observed while a device is being used.	28	26

Device Risks	Loss, theft, or otherwise obtaining data or login credentials directly from the device itself.	13	51
Remote Service Risks	Risks related to the service being accessed or implementation of the app being used	23	7
Loss of Information	Risk of information being lost or account access credentials being obtained by a third party	21	25
Situational Risks	Risks associated with the personal safety of the situation.	6	0

Figure 3.4 summarizes Study 4's participants' preferences for level of information about risks provided in the authentication prompt. 43% expressed a preference to be provided with specific information about the identified source of risk as assessed by the authorization system.

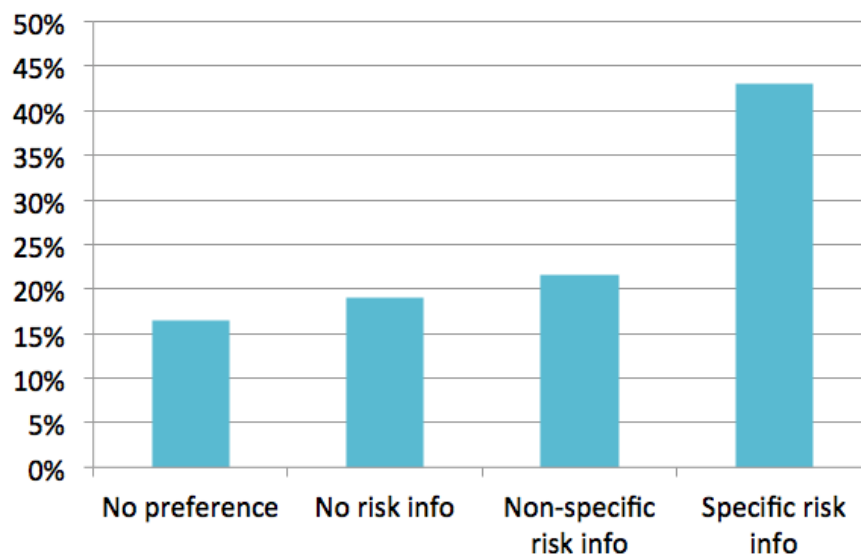


Figure 3.4. Participants' preferred level of risk information in Study 4

3.5 Conclusions

Participants expressed a high level of trust in both apps and web sites on mobile devices, and willingness to use them for sensitive transactions in trusted locations. Reporting rates for risks were low. Based on our surveys, users are most conscious of network and shoulder surfing risks. Consequently, the effect of location on perceived risk was tied to the presence of other people, and nature of the network connection.

Different user groups, applications, and scenarios can lead to significant differences in perceived sources of risk for mobile transactions, and level of awareness of those risks.

These results are based on questionnaires, in which participants were given partial information about a situation and asked to consider the risks. Although risk perception reports can be more reliable predictors of actual behavior than what people say they will do [Keith, Thompson, Hale, Lowry & Greer, 2013], responses may still differ from the risks perceived and factors considered by users in a transaction performed in a real mobile environment. This study does not demonstrate whether these factors would be predictive of actual decisions people make. A complementary study could query participants about perceived risks when they are in locations they normally visit. This would provide data from real environments, but these would be a limited and unique set of environments per person, and it would not be easy to systematically explore factors like the presence of other people, or the familiarity of the location, as we have begun to do here. We did not ask participants about their exposure to the various risks, through personal experience, anecdote or media reports. Thus, we cannot assess the extent to which these risk assessments may have been influenced by, or generalized from prior positive or negative experiences.

These findings suggest ways that organizations could tailor communication with employees and customers to address common concerns, depending on the technological awareness of the user group. Risks that are important to organizations but do not come readily to the minds of users represent a mismatch. Where there is such a mismatch, risk communication may be of value. When user and system perceptions are aligned, there is greater likelihood that users will accept and comply with organizational security requirements such as multi-factor authentication methods.

In Study 3, we explored one possible mechanism for improving compliance – communicating a short-term beneficiary of an authentication, and found no evidence suggesting a benefit from such communication. We observed high levels of acceptance of the authentication in this task. Given this, and the small proportion of participants who cited the benefits given in the prompt, we cannot show that the experiment induced

reasoning about the possible benefits. The overall high acceptance rates point to a possible limitation of the study. In seeking to assure high quality responses we recruited workers with good reputation, who may be more compliant [Peer, Vosgerau and Acquisti, 2013].

However, we note that the less was said about the beneficiary in the authentication prompt, the more participants chose to authenticate. Inducing reasoning could have an adverse effect on outright acceptance, which is predicted by some theory [Acquisti and Grossklags, 2005; Evans, 2003 ,Herley, 2009; Tversky and Kahneman, 1981]]. Further work is necessary to explore this alternate hypothesis.

3.6 Recommendations

The four risk perception and communication studies described above lead to the following recommendations for the design of systems incorporating risk-based mobile multi-factor authentication:

- Explore whether user trust could be improved by making network and application security information readily available within the app or web site being used. Approximately one third of participants in Studies 1 and 2 said they would use network and application security information in decision-making.
- Less than one third of Personal Banking participants mentioned the risk of shoulder surfing. This lack of awareness, coupled with the anticipated use in malls, restaurants and other public places, argues in favor of authentication methods that are not vulnerable to observation. Today's PIN and password-based systems typically provide visual feedback that is vulnerable to shoulder surfing attack.
- In mobile transactions where users need to authenticate, tailoring the level of authentication to the location could fit well with user perceptions of risk, especially if demands are reduced in home and office locations that users perceive as relatively safe. Our results suggest that such tailoring would need to consider shoulder surfing risk, to align well with user perceptions of risk.
- Communicate awareness of threats that do not easily come to mind. Such threats include device loss, theft and hacking, and unauthorized access by insiders (e.g., friends, family, colleagues). With increased awareness of risk, users may be more likely to take security precautions such as locking their phones.

- Presenting short-term costs and benefits in an authentication prompt offers no advantage over a plain call-to-action.
- Provide information about risks where this is possible without compromising security.
- Include a location tracking preference.
- Communicate an application's protective measures against replay attacks and network snooping, to increase user confidence in security.

4.0 HUMAN COMPUTER INTERACTION

User acceptance is critical for the adoption of strong identity and risk-based authorization. Risk-based authorization seeks to adjust the authentication requirements based on: the value at risk, the situation and history. In some circumstances this may lead to an increase or reduction in the burden on users by adjusting authentication requirements. However, users' perception of risk and the actual security risks computed by a risk-based authorization system may not be aligned, potentially negatively impacting user acceptance.

In the previous section we described user studies that explored risks perceived by individuals using online banking and credit card purchases, and technology workers using company information. We also consulted security experts for their assessment of actual risks. Analysis of this data lead to several design recommendations for usable mobile authentication, and for areas where risk communication may be beneficial. This section describes the interface design iterations, informed by the psychometric studies described in the previous section, and the usability studies that were performed to refine and improve these designs.

4.1 Methods

User interface design and evaluation proceeded in several steps:

1. Initial design informed by Studies 1 and 2
2. Heuristic evaluation of initial design, leading to design improvements and recommendations.
3. Revision of design, informed by Studies 3 and 4, and the heuristic evaluation.
4. Usability testing with running prototype
5. Efficiency analysis, to assess the time required to authenticate in different contexts, and with different interface designs.

4.2 Assumptions

In this project we did not gather biometric information from target users, protecting our human subjects from concerns about providing biometrics and having their locations tracked for research purposes. This limited the realism of our evaluation studies. We performed usability evaluations using screen images and running prototypes on our own devices. This assumes that a user's reaction to the user interface and risk communication features in an artificial setting will be consistent with their needs and preferences in real use. Usability should be further tested in a field deployment.

4.3 Procedures

The initial design of a user interface for multi-factor biometric authentication on mobile devices, and heuristic evaluation of that design are reported in detail in Technical Report 4 “Heuristic Evaluation of Mobile Authentication and Design”. In brief, the design was instantiated as an HTML mockup, including risk communication and anti-phishing features. We conducted heuristic evaluation of this design with two usability experts who are not members of the project team. Each expert had over 25 years of experience in usability evaluation. In each heuristic evaluation session, the user interface expert walked through the proposed design, assessing it against a set of established usability heuristics. Our evaluation used Nielsen and Molich’s set of 10 heuristics [MN90, NM90, N94], included as Appendix A in Technical Report 4. This resulted in design recommendations for the prototype.

A running prototype was implemented, incorporating many of these design recommendations, with a focus on the risk communication and design of the authentication prompts. Summative usability testing was performed with five novice users using a simulated banking app with the prototype integrated. Participants were asked to use the banking app to perform three tasks, each requiring a higher level of authentication than the last. Participants were given a short description of the banking app’s authentication scheme, and what they would have done to enroll. They were given no instruction in how to use the prototype, in order to test whether the interface was intuitive to use, and the prompt messages were clear. Where confusion was observed, changes were made to the prompts before the next participant’s session. The biometrics were not enabled for this test. Participants went through the steps of providing a photo or voice sample just as they would in a real authentication, but these samples were not saved or analyzed, and the authentication was set to always succeed. After completing the three tasks, participants were asked to fill in a brief standard usability survey – the System Usability Scale instrument [Sauro 2011] - and then debriefed.

The efficiency analysis applies formal modeling to predict authentication time for skilled users. Using these predictions, we assess the potential impact of a risk-based multi-factor biometric system on the time spent authenticating, under various scenarios. The analysis procedure and results are described in more detail in the report “*Formal Analysis of Authentication Time with Risk-Based Multi-Factor Biometrics Using Keystroke-Level Models*”

4.4 Results and Discussion

The heuristic evaluation produced many design recommendations, summarized in Technical Report 4. Most relevant to risk communication, we found:

- The initial prototype used a lock icon as in web browsers, to indicate secure data transmission. The meaning of this icon was not clear, and the experts recommended removing it.
- To prevent errors and make the system status visible, it is useful to have a visual indication that authentication is going to take more effort before starting. Users may decide to move to a location that is better for the biometric methods, or defer authentication until another time.
- Risk indication should be done without giving too much information to attackers, and without alarming users. One expert did not notice the risk indicator until prompted.
- Users may not remember the icons for specific risk factors, or need that depth of understanding. Both experts felt it would be sufficient to indicate only the level of risk on the login screen.
- A clock icon does not convey 'risk', and neither expert could interpret its meaning. It conveys that the user may be under time pressure. Use standard warning icons.
- The meaning of the last login time is clear, but the value of this to detect an insider attack may not be apparent

In response to these findings, we made the following changes to the risk communication features:

- removed the lock icon for secure communication,
- changed the risk icon from a clock to a gauge, and
- provided three gauge levels (low, medium and high risk), to give useful information about the level of authentication needed without giving away information about how the system works, or explicitly stating how many steps will be required (not possible to give in advance because a biometric may need to be repeated or replaced). The gauge icons were selected by test users from a sample of many different alternative icons.
- Added fade-in-and-out animation and red coloring to the last login message when the previous login attempt failed, and changed the text to 'Failed Login'

We also changed the interface design. The final design is shown in Figure 4.1, in a configuration with three available biometrics: face, voice and fingerprint. In addition to the risk communication features mentioned previously, a central image provides

protection from phishing attacks. At enrollment time, users select an image to be displayed there. If the image is not displayed, the login screen may be fake and users should not enter their credentials.

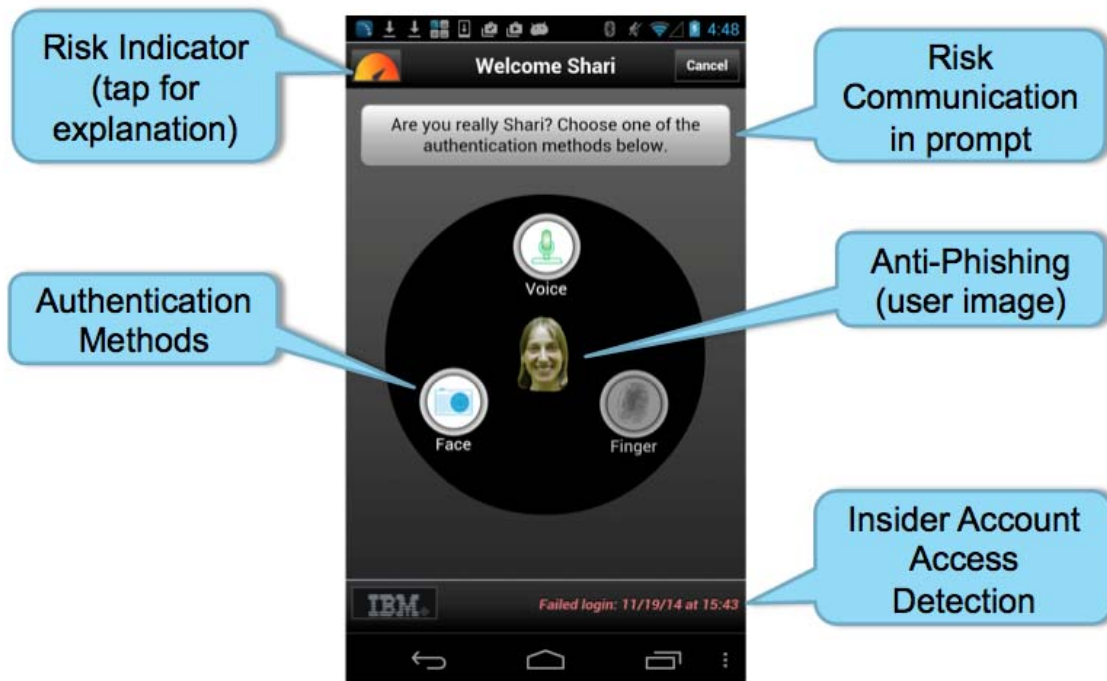


Figure 4.1. Final three-biometric authentication screen design with features highlighted

In the second usability test using the live prototype, we focused on the ability of novice users to perform with no training, just by following the instructions on the screen. All five participants were able to authenticate in all three tasks without assistance, with one authentication failure for one user. We modified the prompts as the study progressed to eliminate confusion. The average System Usability Scale score given by the participants was 74 (range 0 to 100), which is above average for SUS ratings in general.

In the efficiency analysis, we predicted biometric authentication times for the current prototype as 10.1 to 17.0 seconds, with fingerprint being the fastest. The 8-character password is predicted to take 13.6 seconds, which is longer than the single biometrics, but faster than providing fingerprint and another biometric. Figure 4.2 summarizes the

times for different authentication methods, for the current interface with and without system delays, and for an efficient interaction performed by a practiced user.

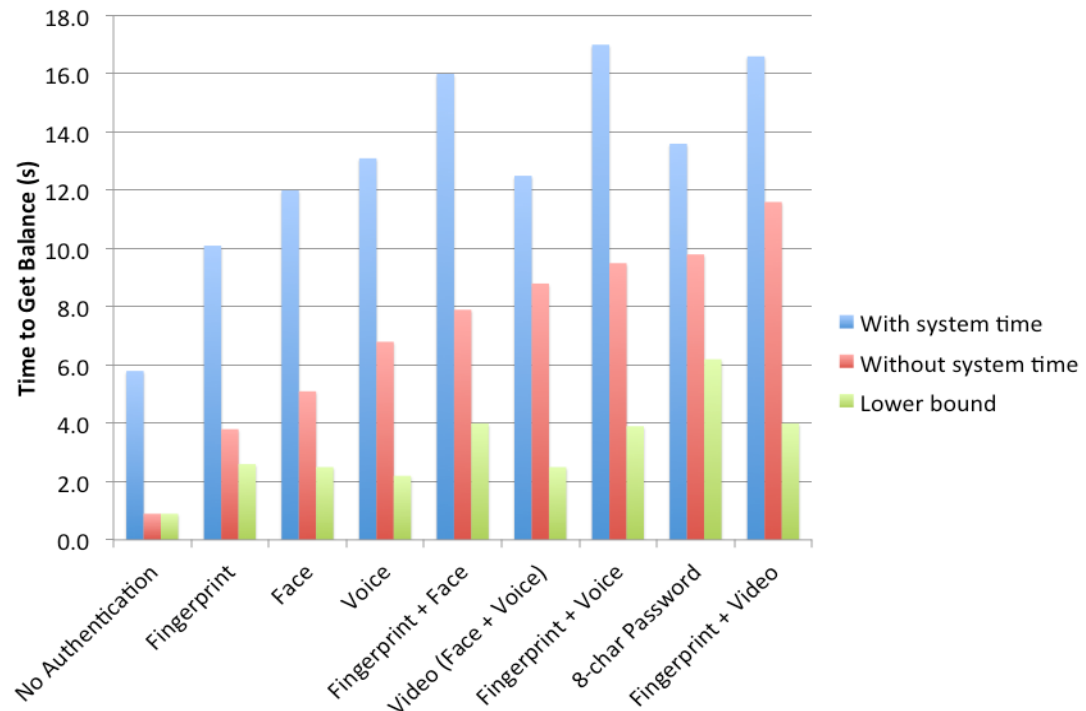


Figure 4.2. Predicted task times for 'Get Balance' task for the current authentication design with and without system/network delays, and an alternative lower bound efficient design, for different authentication methods.

Taking biometric false negative rates into account, we calculated the expected biometric authentication time for a low risk situation, excluding system delays, to be 5.3 seconds. This is a 46% time reduction compared to typing an 8-character mixed alphanumeric password, even with error-free typing.

4.5 Conclusions

Our current interface design for multi-factor mobile authentication balances efficiency with user control, is more efficient than typing a password, and could be expected to be usable by novices with minimal instruction and training.

4.6 Recommendations

The next step for the authentication app should be to test in real mobile contexts in a field deployment. This may highlight challenges and requirements not identified in the controlled environments we have tested in to date.

5.0 BIOMETRIC MULTI-FACTOR AUTHENTICATION

The focus of this work stream is to explore use of multiple biometrics to improve the identity confidence and usability. While biometrics has been in use in many law enforcement agencies, use of biometrics in unsupervised environments poses its own set of challenges. First, the users are not experts in biometrics and hence cannot decide which biometrics is best for an application. Second, because the user needs to be authenticated in many diverse situations, even cooperative users will find some biometrics to be unusable in many scenarios. For example, in an area with a darker lightning condition, face recognition may not be very useful. Similarly, in a noisy environment, speaker recognition may be challenging. Thus use of multiple biometrics and fusion of their result has been proposed to address this problem.

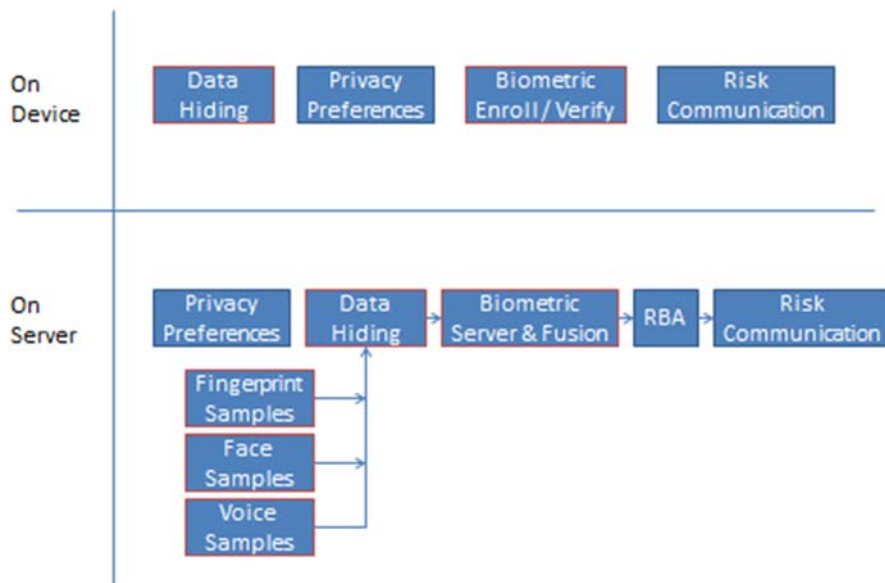


Figure 5.1 Biometric elements of the system

There are two major areas where biometrics has presence in the solution being developed as a part of the project: client side and the server side as shown in Figure 5.1. The client is the smart device in this case where the biometrics signal and basic quality assessment can be made. One of the important components of assuring security in terms of thwarting replay attacks through data hiding is also handled on the client

side. The server side architecture of the system for mobile multi-factor authentication is shown in Figure 5.2 with the biometrics components highlighted.

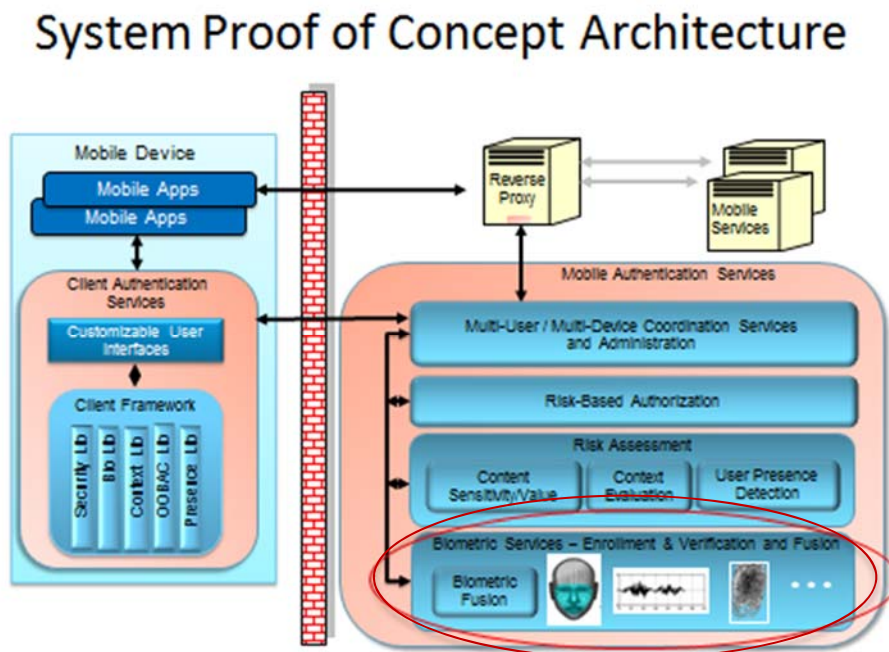


Figure 5.2 System Architecture

The other components of biometrics include biometrics signal acquisition and quality control.

5.1 Methods

The primary activity in this work stream involves exploring the use of multiple biometrics modalities that can provide uncorrelated identity measurements through the users' biometrics signals. It is believed that the multiple measurements will improve the overall confidence about the identity of the user while each measurement may have challenges to decide the user's identity. Our current selection of biometrics includes face, voice and fingerprints. We use a commercially available face recognition engine, speaker recognition engine and a fingerprint recognition engine installed and integrated with the authentication app. Finally, we fuse the results of these biometrics-based identity confidences in terms of scores produced by the individual engines to improve the overall security while complying with the security policies of the enterprise. The overall approach is shown in Figure 5.3. As desired by the policy, one or more biometrics signals can be submitted to the respective biometrics engine and the scores

communicated to the fusion engine which will produce the final score to be assessed by the risk based authorization engine. If the policy requirements are not met as determined by the risk assessment stage, the user will be encouraged to provide additional sources of identity information and the fusion engine will update the identity assurance score. This process either continues until the user is able to meet the desired level of identity based on the policy for the particular transaction at hand. If not, the system can fall back to an exception mode or reject the transaction.

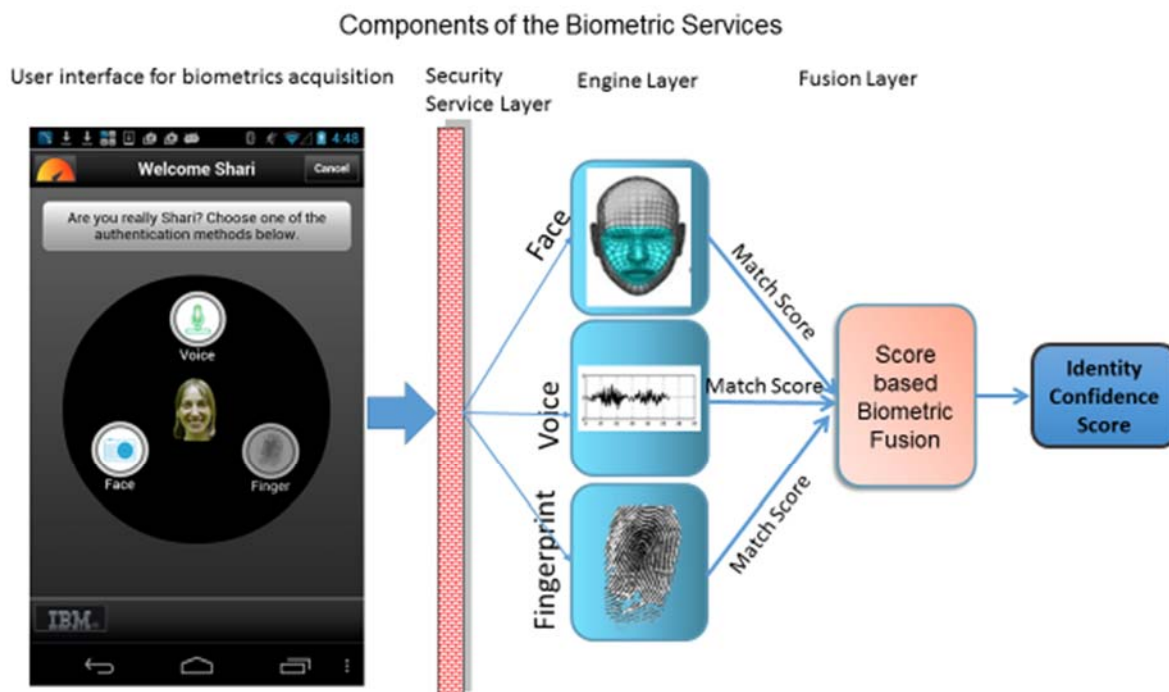


Figure 5.3 Flow for biometric acquisition and scoring

In order to support multiple biometrics, we needed to select a variety of biometrics modalities that can provide independent measurements of the individuality of the user. While in the forensic community, the popular choices are face, fingerprint and iris, in the smart device space, we have to assess what can be easily workable with the non-forensic environments where the biometrics data is collected without any supervision.

Recently, Apple introduced fingerprint scanners in their latest iPhone 5S. It is expected that the same may be extended to many other smart devices from Apple soon. The Android devices are not being far behind either. Samsung S5 has introduced a different

type of fingerprint scanner on the front side of the device, other vendors have placed fingerprint scanners on the back side of the device.

While the hardware manufacturers wrestle to find out the best fingerprint scanner for their latest models, we chose to proceed with our experimental evaluation with the externally connected fingerprint scanners. One of the reasons for choosing an externally connected fingerprint scanner is that the most popular iPhone 5S does not provide access to the device fingerprint sensor data for any other transaction beyond unlocking the devices and using it for iTunes purchases. We note that the fingerprint scanners in the current experiment are very different from what would be available in future. However, some applications will still benefit from the testing of these fingerprint scanners in the smart mobile devices when they are being used in secure environments. The second issue that guided our decision to pursue the use of external fingerprint scanners is based on the fact that the iOS fingerprint scanner is a much smaller fingerprint scanner than normally used in say laptops. Hence it requires several enrollment samples. The final reason for selecting the external scanners was driven by the NIST certification of the sensors. Many sensors have been certified for PIV program. Our choice was guided by the list of certified scanners for compatibility with the user base.

5.2 Assumptions

We aim to use sensors available on mobile phones to design multi-factor authentication based on a fusion of biometric sensors chosen for optimal performance. The success metrics for this component will be user acceptance and lowered friction in authentication while supporting compliance requirements of the enterprise through a policy.

While for face and voice, we can use the built-in sensors, for the fingerprint matching, we are relying on an external fingerprint sensor for iOS devices. The internal built-in fingerprint scanner is not accessible to applications in terms of providing the fingerprint signal at this time.



Figure 5.4 FbF mobileOne QuickDock

As can be seen in Figure 5.4, this fingerprint scanning device can be used with iPhone and iPads with the 30-pin connectors. This can be switched between devices. The fingerprint scanner is FIPS201/PIV certified and provides grayscale images at near FBI recommended 500 dpi. It also supports autocapture by detecting presence of a finger on the scanner automatically. The SDK also supports access to the fingerprint image for analysis on the smart device or the server.

In addition to the fingerprint devices, we also selected the engines for face, fingerprint and voice. Based on our approach to find vendors who perform well, are reasonably priced and provide access to the images and features, we chose VeriLook [VeriLook], VeriFinger [Verifinger] and VeriSpeak [VeriSpeak] engines. These engines are available easily across the globe and affordably priced with good accuracy performance.

5.3 Procedures

Biometric fusion can involve a wide range of techniques from a simple decision level fusion to very complex methods involving optimization techniques. The overall goal is being able to strike an optimal balance between usability and security. Several measurements are made during the process of biometrics signal processing and analysis. In addition to the final match score that leads to a decision, there are several other factors that are available. Earlier it was seen how quality can be measured from the biometrics signal. Often an acceptable quality is what the biometrics users expect for better results. Once the quality metric is beyond the acceptable threshold set by the system, the quality index is not used. Similarly, the knowledge of underlying performance of the biometrics engine is known to the system designed either by the vendor or by conducting a performance analysis on a collection of biometrics samples. In our fusion policy module, we employ these two measurements.

Quality of a biometrics signal can decide the impact of the biometric system performance. Even best known biometrics modalities can end up having very low performance when the signal quality is poor and vice-versa. Most biometrics systems employ a reject option to not take any further steps when a poor quality biometrics is detected. Once the biometrics quality is acceptable, the match score is computed leading to a match/no match decision. It can be easily seen that the degree of match is very high when the quality is very good. Hence using the quality as a part of the fusion policy would be a good idea. It can help us decide how much weight to be assigned to a particular modality.

The statistical error rate performance is used to decide which engines can provide good results. The error indices commonly used are: false accept and false reject (also

referred to as false positive and false negative respectively). The interplay between these two errors is often shown in a Receiver Operating Characteristics (ROC) curve. Once a threshold is chosen, both the errors get decided. It is quite well known that these two errors can't be optimized independently. Once the false accept rate (FAR) is chosen, the false reject rate (FRR) gets automatically decided. While false accept rate is related to security of the system, the false reject rate is related to the usability of the system. Thus both are important to the overall system experience. Researcher often have an operating point called as the equal error rate (EER) point on the ROC curve where the two errors are equal as an important point. At this error rate (by definition), $FAR = FRR$. We use this point in our fusion policy for its simplicity in representing the overall performance of the biometrics system. Normally a smaller EER would imply a better biometrics system in terms of accuracy. Another reason for choosing this point is based on the fact, it is not known a priori if the system is optimized for higher security or higher usability. Hence the fusion policy decision can be based on the EER. Additionally, it reduces one variable in place of dealing with two errors.

In order for the usability team to evaluate the user experience, it was felt that the project would benefit from having access to an integrated multi-biometrics service. To achieve that goal a multi-biometrics server was designed that can handle face, finger and voice biometrics. A web-API was designed for the service to support two functions: enrollment and verification. The enrollment function collects the biometrics signals of a subject and stores it for use during the verification service. In the verification service, in addition to the username, the test subject would provide the biometrics signals.

We have developed a web-based API for interacting with the biometrics engines available for the project. Each biometric modality has an interface to compute the match score and quality assessment through the interface. Based on the input signal/image submitted for verification against a claimed user id, a score is computed and returned for future use. In addition, a similar interface has been developed to handle more than one engine response to compute the fusion confidence. The resulting fusion score is converted to a confidence score that can be used in the final stage of risk evaluation along with other parameters. Our overall platform supports three biometrics modalities: face, finger and voice.

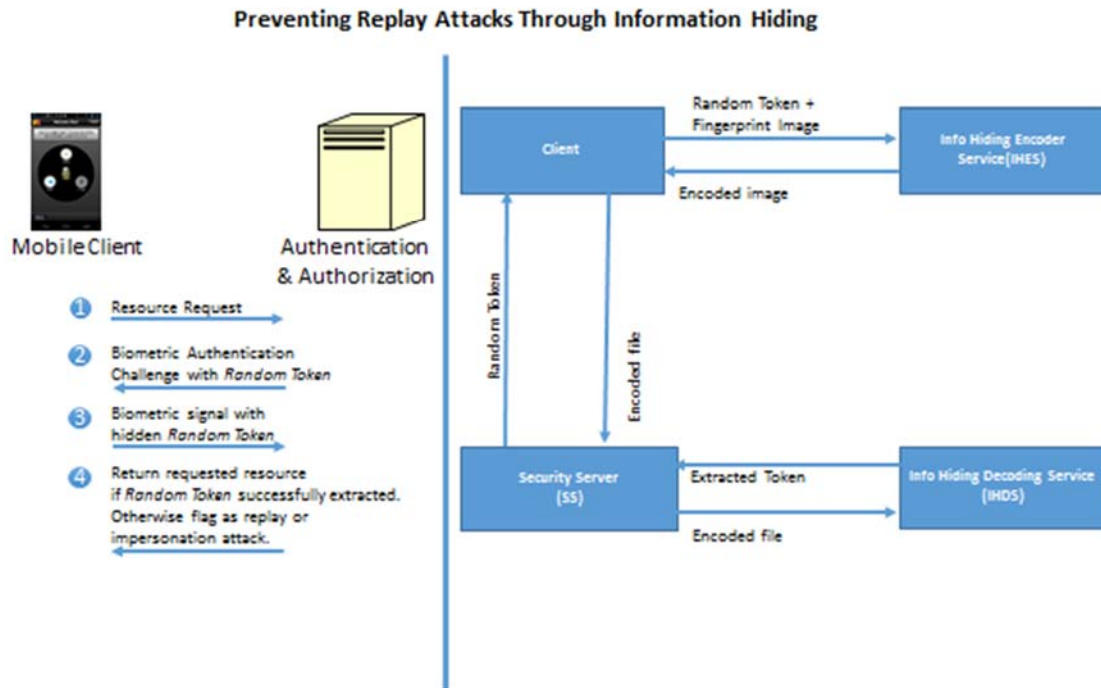


Figure 5.5 Preventing Replay Attacks Through Information Hiding

One of the key components of the project is related to the security enhancements in the biometrics in terms of thwarting replay attacks. As the biometrics is captured on the client device, we propose to invisibly watermark the signal using data hiding method. The signal can't be replayed to fool the server for another transaction. In this approach, the server decodes the hidden message before the signal is sent to the biometrics engine for further analysis as shown in Figure 5.5.

5.4 Results and Discussion

A multi-biometrics fusion system has been developed using three modalities of face, voice and fingerprint. Each modality can be independently used for authentication while more than one can be combined in a serial fashion to improve the authentication confidence based on the trust required by the transaction. For example, if face modality was used and obtained a confidence of 0.6 and the transaction needed a confidence of 0.7, the user can now provide fingerprint to boost the overall identity confidence. The fusion algorithm takes the match scores and the quality of the input signal to compute the overall fusion score. The fusion score is translated by the risk management system to the authentication confidence. The individual engine scores may fall below the expected identity confidence threshold while the overall fusion score can be expected to cross that to clear the transaction. The web interfaces for single modality and fusion are

shown below. As these run in the back end server, the user never sees these screens. The integration middleware uses the interface to compute the final result.

The three functions supported are: (i) enroll a user; (ii) verify an enrolled user; and (iii) delete an enrolled user.

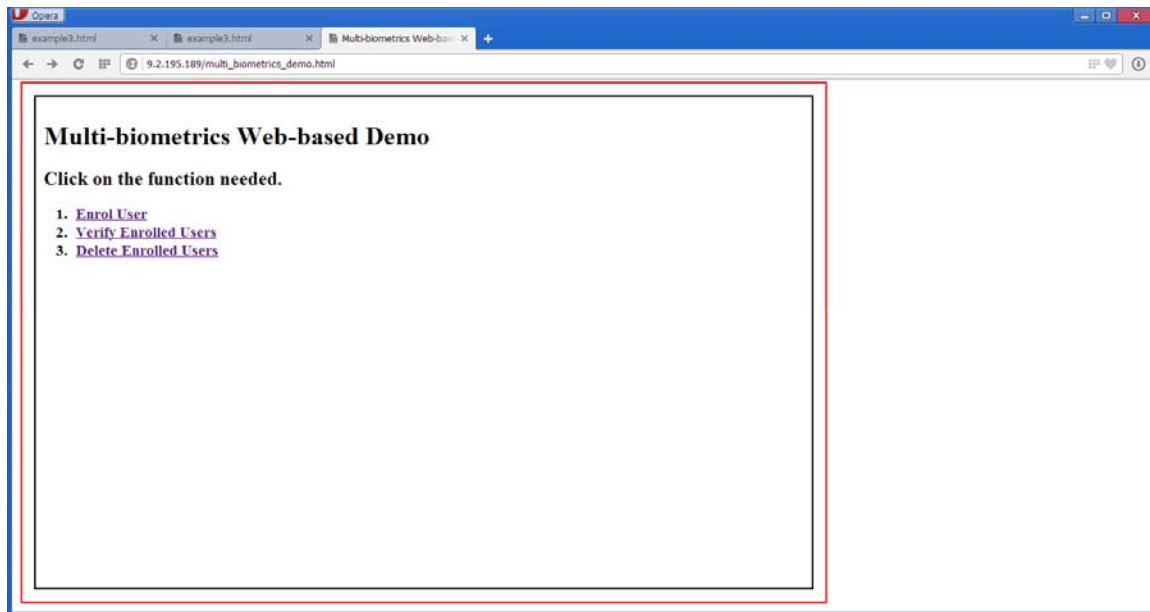


Figure 5.6 Multi-biometrics Web Interface

1. Enroll:
Input:

userid (6-8 characters without space or special characters)

Face file (jpeg), fingerprint file (tif) and voice file (wma)

The service creates a userid directory and saves the images and signal.

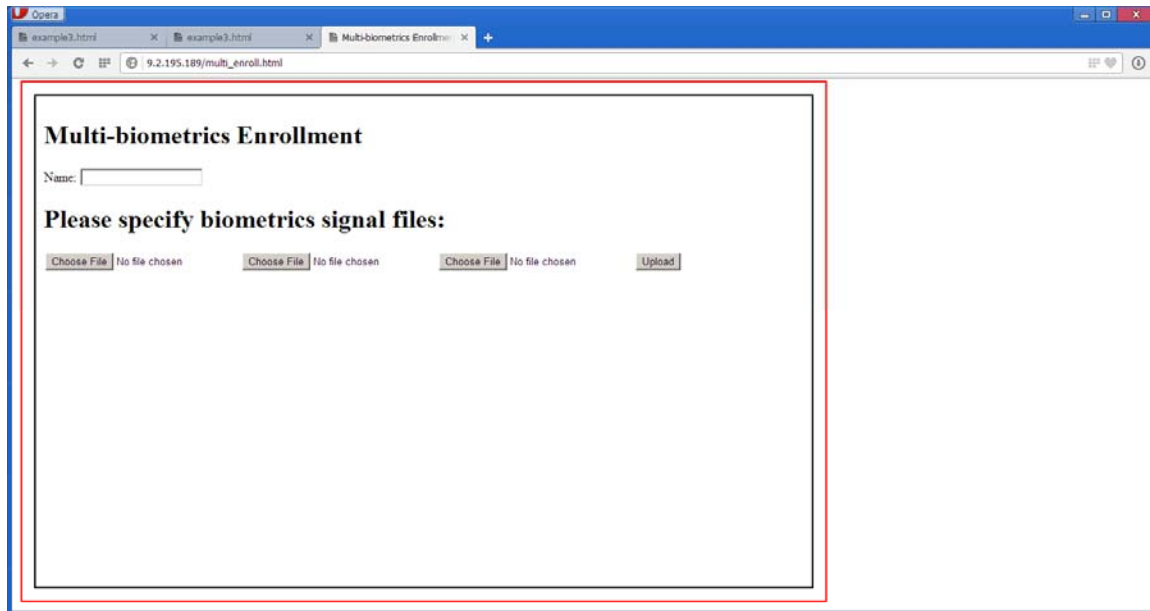
A screenshot of a web browser window titled 'Opera'. The address bar shows '9.2.195.189/multi_enroll.html'. The page content is titled 'Multi-biometrics Enrollment'. Below the title is a 'Name:' label followed by a text input field. Below that is the heading 'Please specify biometrics signal files:'. Under this heading are three file selection controls, each consisting of a 'Choose File' button and the text 'No file chosen', followed by an 'Upload' button. The entire form is enclosed in a red rectangular border.

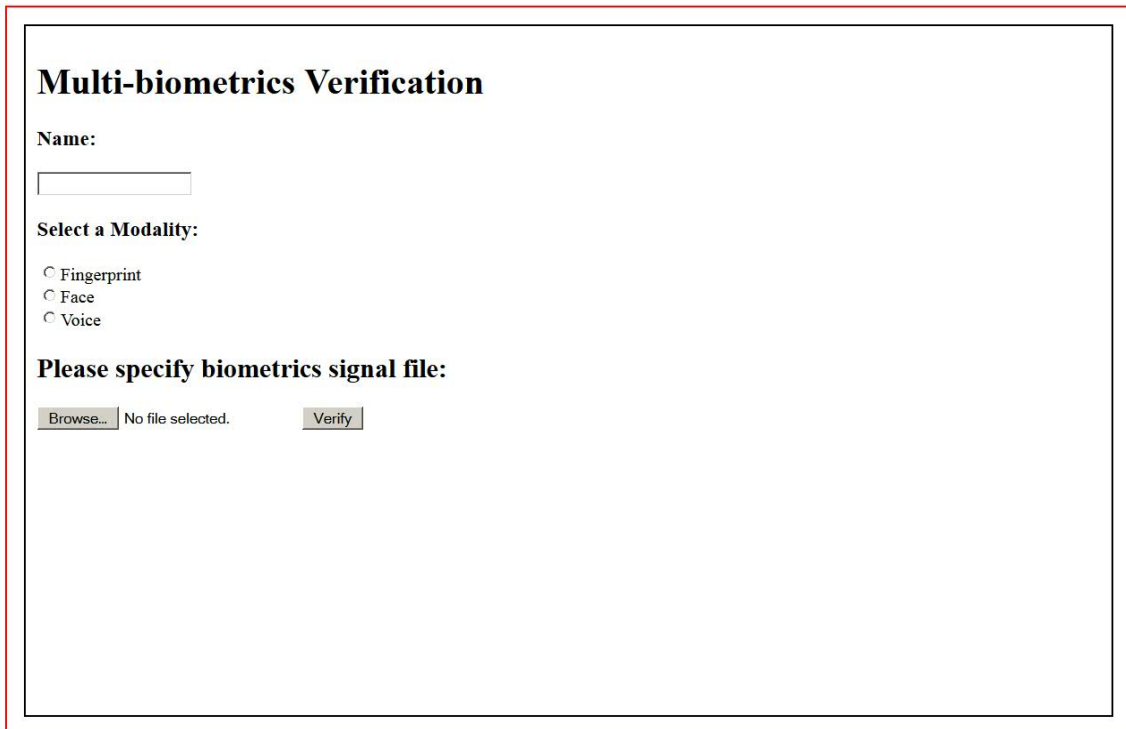
Figure 5.7 Multi-biometrics Enrollment

2. Verify:
Input:

User id (same as the one used during enroll)

Face file (jpeg), fingerprint (tif) and voice file (wma)

The service reports the scores of matching the query face, fingerprint and voice against the enrolled templates of the same person.

A web form titled "Multi-biometrics Verification" enclosed in a red border. The form contains a "Name:" label followed by a text input field. Below this is a "Select a Modality:" label with three radio button options: "Fingerprint", "Face", and "Voice". Underneath is a label "Please specify biometrics signal file:" followed by a "Browse..." button, the text "No file selected.", and a "Verify" button.

Multi-biometrics Verification

Name:

Select a Modality:

☐ Fingerprint

☐ Face

☐ Voice

Please specify biometrics signal file:

No file selected.

Figure 5.8 Multi-biometrics Verification

3. Delete:

Input: user id (same as the enroll)

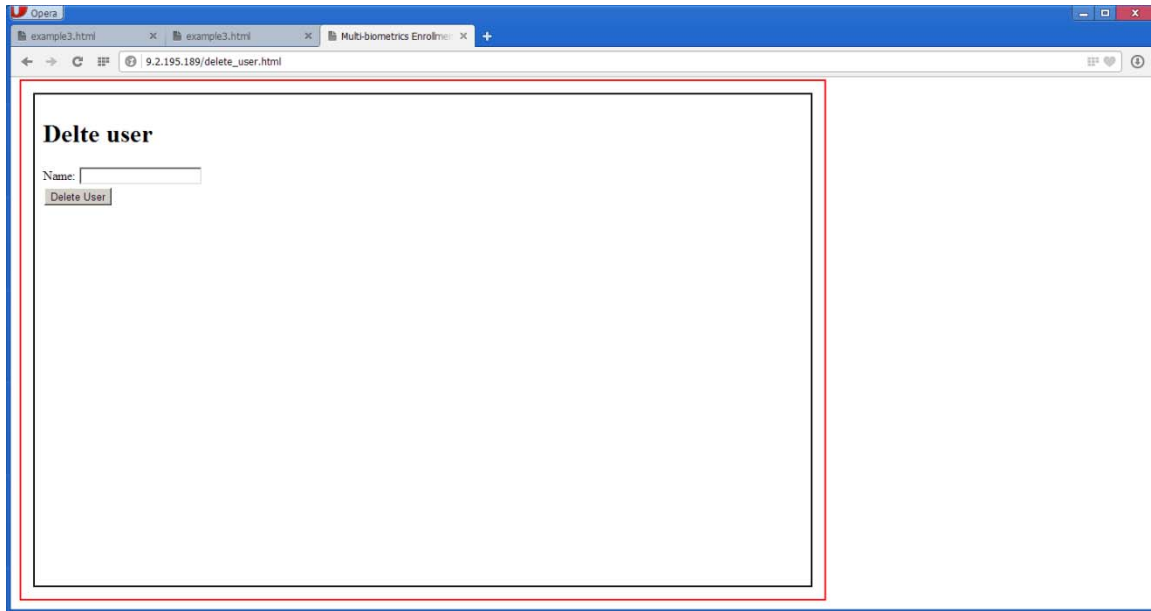


Figure 5.9 Delete user

< Modality: fingerprint>
< Software version: 4.5.0.0 >
< Fingerprint image quality value: 0.50 >
< Fingerprint pattern class is "TentedArch", confidence 0.73 >
< Fingerprint image quality value: 0.75 >
< Fingerprint pattern class is "PlainArch", confidence 0.70 >
< Fingerprint verification minimum match score = 0 >
< Fingerprint verification maximum match score = 2000 >
< Fingerprint verification match score = 0 >

Figure 5.10 Fingerprint Verification Sample Response

< Modality: face>
< Face Verification Software version: 4.5.0.0 >
< Face Min quality score: 0.0 >
< Face Max quality score: 100.0 >
< Face Min match score: 0 >
< Face Max match score: 1600 >
< Face location for image: 1 = (169, 242), width = 183, height = 183, Quality = 66.52 >
< Face location for image: 2 = (291, 358), width = 459, height = 459, Quality = 80.68 >
< Face verification matching score: 0 >

Figure 5.11 Face Verification Sample Response

< Modality: voice>
< Software version: 4.5.0.0 >
< Voiceprint Min Match score: 0 >
< Voiceprint Max Match score: 100 >
< Voiceprint verification match score = 5 >

< Voiceprint quality min score: 0 >
< Voiceprint quality max score: 100 >
< Voice quality score: 46 >
< Voice SNR score: 26 >
< template extracted from audio: 00:00:01.1680000, duration: 00:00:09.5520000 >

< Voiceprint quality min score: 0 >
< Voiceprint quality max score: 100 >
< Voice quality score: 50 >
< Voice SNR score: 62 >
< template extracted from audio: 00:00:00.5920000, duration: 00:00:08.0480000 >

Figure 5.12 Voiceprint Verification Sample Response

Multi-biometrics Fusion

User Name: Algorithm:

Modality	<input checked="" type="checkbox"/> Face	<input checked="" type="checkbox"/> Fingerprint	<input type="checkbox"/> Voice
Software Version	<input type="text" value="4.5"/>	<input type="text" value="4.5"/>	<input type="text"/>
Quality Score	<input type="text" value="35"/>	<input type="text" value="56"/>	<input type="text"/>
Quality Min Score	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text"/>
Quality Max Score	<input type="text" value="100"/>	<input type="text" value="100"/>	<input type="text"/>
Match Score	<input type="text" value="45"/>	<input type="text" value="65"/>	<input type="text"/>
Min Match Score	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text"/>
Max Match Score	<input type="text" value="100"/>	<input type="text" value="100"/>	<input type="text"/>
Environment Score	<input type="text" value="23"/>	<input type="text" value="45"/>	<input type="text"/>

< Fusion Modality: face fingerprint voice >
 < Face score : 10.000000 >
 < Fingerprint score : 20.000000 >
 < Fingerprint score : 30.000000 >
 < Fusion score = 0.137255 >

Figure 5.13 Multi-biometrics Fusion Web Interface and Sample Response

As mentioned earlier, we show the overall statistical performance of the individual biometrics engines and the fusion results in terms of a set of Receiver Operator Characteristic Curves (ROCs). The individual engine performance is directly from vendor reported results. The fusion ROCs are based on the individual engines used in the fusion rule.

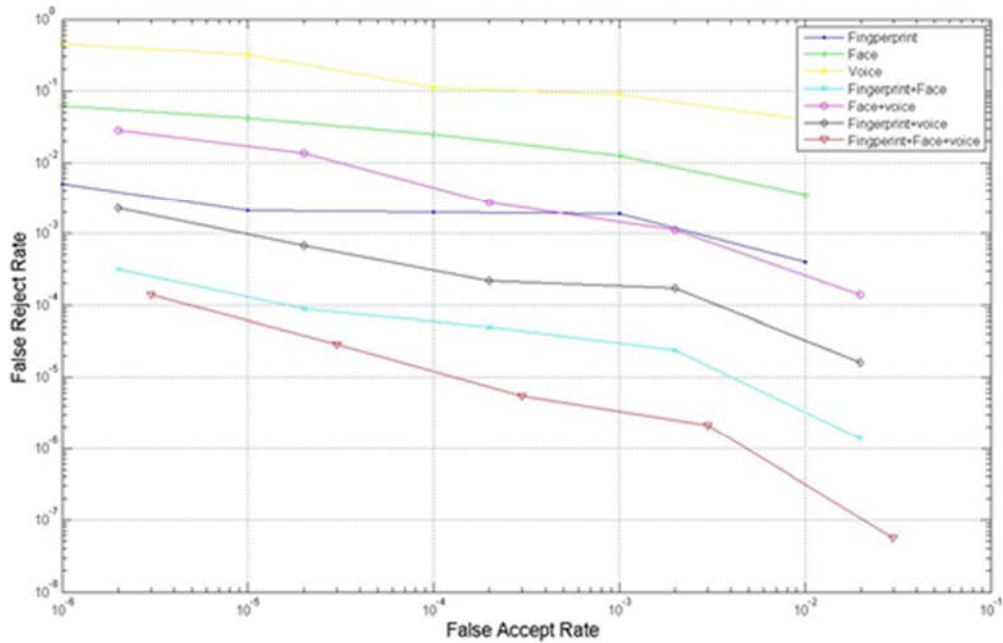


Figure 5.14 Fusion ROC

5.5 Conclusions

We have been successful in building a multifactor biometrics based authentication system that can provide usable and frictionless authentication in mobile devices. In order to improve the overall security of the system, we employ state-of-the-art data hiding methods to hide messages in the biometrics signal that can be decoded by the secure server.

The experimental system was demonstrated at the DHS sponsored cybersecurity conference and was extremely well received.

5.6 Recommendations

We believe the system is ready for a field trial where meaningful data can be collected both in terms of user experience and real biometrics signals to characterize the performance both at engine level, fusion level and usability level and compliance level against a specified policy. We also believe the data hiding method can also be tested in the field trial by suitably devising experiments where signals collected from a previous transaction can be resubmitted.

6.0 RISK-BASED AUTHORIZATION

Risk-Based Authorization (RBA) estimates risk from the contextual factors of an access request, and makes an access control decision based on the risk estimate, the trust on the user (the subject) and the value/sensitivity of the requested resource (the object). The goal of RBA is to take calculated risk to maximize legitimate access provided to the users while keeping risk within an acceptable level.

An access control system based on RBA would have the following properties:

- *context awareness*: A decision is based not only on the subject and the object, but also on the context of the access request.
- *non-binary decision options*: besides the usual *allow* and *deny*, there can be one or more decision options that prescribe risk mitigation actions. The goal of risk mitigation is try to reduce risk below certain level so an access request can be allowed. Although there is no guarantee that taking risk mitigation action(s) will result in the access request being granted.
- *multi-round decision making loop*: the system may take several risk mitigation actions over several rounds to see whether the risk can be reduced enough to grant the request. In each round after risk mitigation, the system will estimate risk to see if the request can be granted. For example, in our prototype, the system may ask a human user to provide biometrics samples in order to increase authentication confidence and thereby to reduce the risk. The user may be asked to provide face, voice, and fingerprint samples in different rounds. Both the system and the user have the discretion to either request/provide more samples or to abort the request in any round.

Another very desirable, but not strictly necessary property, is the ability to continuously learn and adapt, so as to improve the accuracy of risk estimation, to make better decisions, and to adapt to the ever changing environments and user behaviors. Our prototype does have the ability to continuously learn new user behaviors and to update user behavior profiles used in risk estimation.

It should be noted that RBA aims to manage the risk and does not attempt the impossible of eliminating the risk completely. In other words, RBA aims to balance security and usability, maintaining enough security and not inconveniencing the users too much.

6.1 Methods

The most fundamental concept in RBA is risk. For RBA, risk is the uncertainty that an access control decision made now may lead to future events with undesirable consequences. In other words, such future events will cause tangible or intangible damages.

6.1.1 Risk, Uncertainty and Time

RBA is mainly concerned with the following kinds of uncertainty:

- the probability of such a future event, which is usually neither 0 nor 1. We estimate such probabilities using data, experience and/or expert opinions. And we must emphasize that *data, experience and expert opinions only came from the past* (history).
- the uncertainty about the probability estimation. Estimation is based on the *past*; but the future may resemble, but will not be exactly the same as past. So there is always this unknown difference between the past and the future. This difference is a major source of uncertainty. A *black swan*, a rare but catastrophic event may not be preset in the data or experience, but it may happen in the future.

As a side note, accounting for the possibility of black swans has been the major feedback we received since we started research on risk based security [Chen07] in 2004.

- the uncertainty in the input data to risk estimation. All sensor data have inherent inaccuracy. Some inputs, such as anomaly scores, are basically (semi-)subjective indices.
- the uncertainty in the risk estimation model, its structures and parameters. The model can only be an approximation to real risk; and we believe no one knows exactly what real risk is.
- The uncertainty in the impact/magnitude of the potential damage.

In summary, RBA tries to manage the future risk by making decisions and taking actions accordingly in the present. Neither we, nor anybody else, can know the future exactly. So we try to make our most educated guess about the future while taking the uncertainty into account, using the data and knowledge from the past and the current context.

We must note that in this project, the risk estimation focuses on estimating the probability of damage. The impact/magnitude of the damage, trust on the user, and the risk are handled as three separate entities in our *Trust-Value-Risk* based policy model that is to be discussed later. In this policy model, the values / sensitivity of resources correspond to impact.

6.1.2 Uncertainty

6.1.2.1 Where Is Uncertainty?

- Profiling human behavior patterns. Normal human beings have fluctuations in their behaviors.
- Quantifying/measuring a phenomenon/concept against which there is no objective metrics. Anomaly scores are a good example. In general, such a quantity/measure is a (semi-)subjective index which is derived from data in a principled, but (semi-) subjective way. Bio-metric fusion scores are another example.
- Using expert opinions based on knowledge and experience but not real, objective measurements.
- Using objective measurements that have inevitable inaccuracy. GPS coordinates are a good example [GPS13][GPS14][GPSa].
- Using data to predict the future, such as predicting the probability of a future event. Any data must have come from the past, but the future will not exactly repeat the past. Also, no data set has perfect, complete and un-skewed coverage.

6.1.2.2 Reasons to Account for Uncertainty

- allowing the normal fluctuations in human behaviors, and distinguishing these fluctuations from anomalous behaviors.
- allowing the normal, inevitable inaccuracy and incompleteness in sensor data, such as GPS coordinates (latitude and longitude), and distinguishing drift in data points caused by such inaccuracy from real difference among data points.
- allowing for the inevitable inaccuracy in data analytics.
- allowing for possibilities that are not covered or contrary to (semi-) subjective quantities, expert opinions, knowledge and past experience.

- allowing for the possibility of future black swans that are not covered by data, expert opinions, knowledge and past experience.

6.1.2.3 Handling/Modeling Uncertainty

We used the concepts and tools from Fuzzy Logic [JSM97][LL96] and Bayesian Statistics [Bol07][CJB11] to handle and model uncertainty, and to estimate risk. More details are provided in Section 6.3 (Procedures section).

6.1.3 On Accuracy of Risk Estimate

Because risk can only be estimated, there are always concerns about the (in)accuracy of the risk estimates and its effect on making access control decisions based on risk estimates. The concerns we heard most often are:

- *Can a reasonable, informed decision be made based on a risk estimate?* In a computer system, there is typically only a *finite, discrete set of decision options* to choose from when making an access control decision. This implies a range of risk estimate will be mapped into a single option. In other words, the whole range of possible risk estimate values will be quantized to match the available decision options. A risk estimate only needs to be accurate enough to be within the right range. Of course, some risk estimates may not be within their right ranges, and this brings up the next concern.
- *Will my system run wild if its RBA makes a decision based on a risk estimate that is far off?* It is possible that a risk estimate is far off. To address this problem, we can use a traditional access control system (such as ACL) whose policy specifies what must never be allowed. And decisions made according to this policy can veto the decisions made by RBA. In other words, a big *sandbox* can be erected around RBA to block access control decisions that are seriously wrong. Either the RBA or the sandbox can deny an access request, but the two must agree to grant a request. Our prototype does not implement the sandbox.

6.2 Assumptions

Our RBA prototype makes the following assumptions:

- People are generally creatures of habit and often exhibits regular behavior patterns. This assumption is useful because it allows us to build behavior profiles to assess how anomalous a user's behavior is when he makes an access request. This assessment is represented as an anomaly score which is input to risk estimation.

- There is only a finite, discrete set of decision options to choose from when making an access control decision.
- RBA can be coupled with a traditional access control system that can veto access requests granted by RBA.

6.3 Procedures

6.3.1 Handling Uncertainty

We use the concepts and tools from Fuzzy Logic [JSM97][LL96] and Bayesian Statistics [Bol07][CJB11] to handle and model uncertainty.

6.3.1.1 Handling Normal Fluctuations, Inaccuracy and Incompleteness

We use Fuzzy Logic concepts to account for the normal fluctuations in human behaviors, inaccuracy in sensor data and analytics, and incompleteness of data sets. For example, we developed our own geo clustering technology to find the locations a user frequents from the user's location/GPS data. A location is modeled as a centroid with a roughly rectangle bounding box around it. Due to inaccuracy and incompleteness of GPS data, human behavior fluctuations and inaccuracy in geo clustering, such a box is only an approximation. So when we build the user's geo-location-time profile based on how often and when the user is at these locations, we consider at-a-location as a non-boolean, fuzzy property. We treat a location as a fuzzy set and compute the fuzzy membership of a geo data point (GPS coordinate) in this location: how much this point is in this location. When building the user's geo-location-time profile, the frequency count of a location is the sum of the geo data points' memberships in this location. The fuzzy union of (1-membership) over a point's memberships in all locations, is how much the point is at none of the locations, and is used for accumulating the frequency count of the special *all-others* bucket in the profile.

6.3.1.2 Handling Possibilities Not Covered by Past Data and Experience

We use concepts/practices from Bayesian statistics to account for possibilities not covered by data, experience or expert opinions.

Basically, we change an input point value into a Beta Distribution [NIBE] over the range of the input. Such a distribution covers possibilities that are not the input point value.

Below are some excerpts from [NIBE] on Beta Distribution.

The general formula for the probability density function (*pdf*) of the Beta Distribution is

$$f(x) = \frac{(x-a)^{p-1}(b-x)^{q-1}}{B(p,q)(b-a)^{p+q-1}} \quad a \leq x \leq b; p, q > 0$$

where p and q are the *shape parameters*, a and b are the *lower and upper bounds*, respectively, of the distribution, and $B(p,q)$ is the *beta function*.

The beta function has the formula

$$B(\alpha, \beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1} dt$$

The case where $a = 0$ and $b = 1$ is called the **standard beta distribution**. The *pdf* for the standard beta distribution is

$$f(x) = \frac{x^{p-1}(1-x)^{q-1}}{B(p,q)} \quad 0 \leq x \leq 1; p, q > 0$$

The *location* and *scale* parameters can be defined in terms of the lower and upper limits as follows:

$$\text{location} = a, \quad \text{scale} = b - a$$

The *mean* for the standard distribution is

$$\frac{p}{p+q}$$

The *variance* for the standard distribution is

$$\frac{pq}{(p+q)^2(p+q+1)}$$

Figure 6.1 Beta Distribution

For our purposes, we note the following:

- The location parameter shifts the standard distribution along the x-axis; and the *scale* parameter *stretches* the standard distribution along the x-axis. The general formula for the *pdf* can be expressed in terms of the standard beta distribution:

$$\left(\left(\frac{x-a}{b-a} \right)^{p-1} \left(\frac{b-x}{b-a} \right)^{q-1} / B(p, q) \right) \times \frac{1}{(b-a)}$$

So the general *pdf* formula can be obtained from the standard formula through scaling and shifting; basically by replacing the x in the standard formula with $(x-a)/(b-a)$ and adding the scaling factor $1/(b-a)$.

- For the standard distribution, the mean only depends on the ratio p/q . And if both p and q are greater than 1, then the distribution is a bell-shaped curve with its peak at the mean.

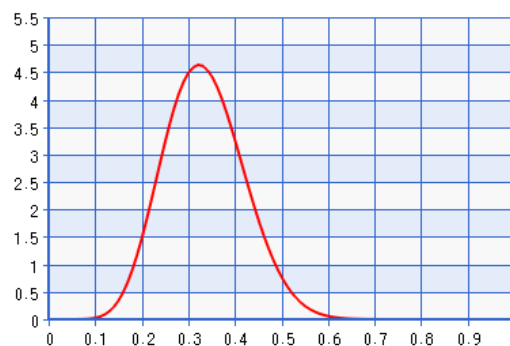


Figure 6.2 Sample Beta Distribution Curve

- *The larger the sum ($p+q$), the smaller the variance, and vice versa.*
- For the standard distribution, the shape parameters p and q can be derived from the mean and the variance.

$$p = \text{mean} \times ((\text{mean}(1 - \text{mean}) / \text{variance}) - 1)$$

$$q = p \times (1 - \text{mean}) / \text{mean}$$

And therefore the *standard distribution can also be specified with mean and variance*. In our work, we found it is sometimes more convenient to specify the standard distribution with mean and $(p+q)$.

- We can fix the p/q ratio and therefore the mean while decreasing or increasing the variance by scaling up or down the the sum $(p+q)$.
- The larger the variance, the higher the degree of uncertainty and vice versa.

When we change an input point value to a beta distribution, we view this point value as the most likely value and use it as the mean of the distribution, and determine the variance based on how (un)certain we are about the point value. The more uncertain we are, the larger the variance (and therefore the smaller $(p+q)$). The variance is usually a (semi-) subjective value based on experience (or "expert opinion") and must be greater than 2, since both p and q must be greater than 1 to have a bell shaped distribution. This technique is learned from [CJB11].

In our work, we would usually choose a larger variance if we want to slightly over-estimate the potential risk the input implies. And we "*normalize*" the input ranges to be $[0,1]$ to make it more convenient for further processing.

We must note that due to the inherent inaccuracy/uncertainty in input point values and the way variances are determined, these beta distributions are only approximations of the uncertainty in inputs.

Indeed, if uncertainty could be accurately quantified, then it would not be uncertainty.

To simplify the computation and our implementation, we further quantize a Beta distribution into a *probability mass function (pmf)*: a discrete pdf. We divide the input range ($[0,1]$) into a few ranges/quanta, and assign each quantum a probability mass which is the integration of the pdf over the quantum. We usually use five quanta: *very low, low, medium, high, very high*.

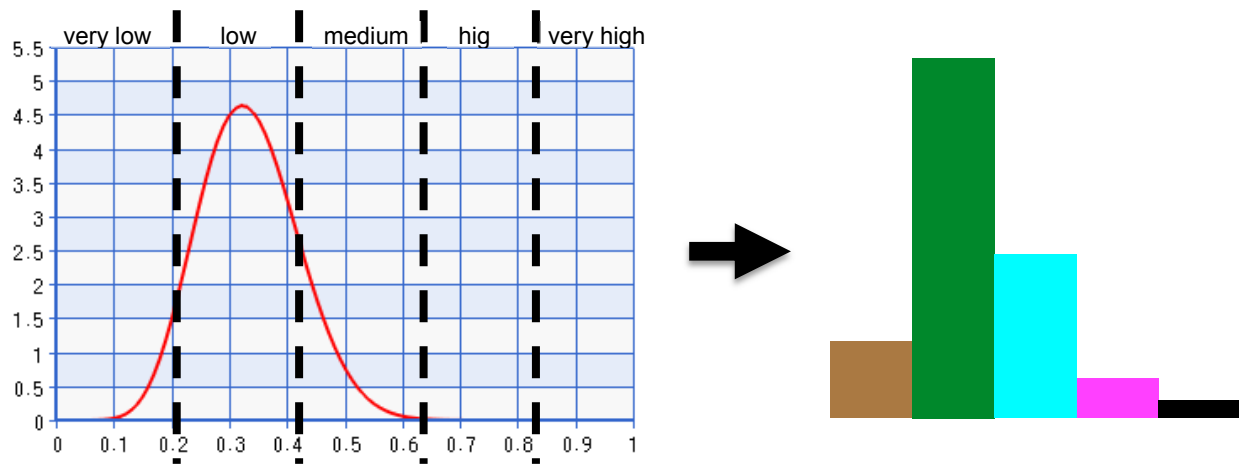


Figure 6.3 Conversion of *pdf* to *pmf* example

For example, the quanta of an anomaly score input would be named: very normal, normal, neutral, anomalous, very anomalous. The exception to the five-quantum practice is when the input is inherently *boolean*, such as the result of a password match, in which case we use *only two quanta* : *true* and *false*, *match* and *mis-match*, etc. We believe such quantization is justified because each *pdf* is only an approximation and does not carry the accuracy that is usually implied by a smooth, continuous function.

6.3.1.2.1 A Performance Optimization

We implemented an approximation of the quantization described above. We used the *org.apache.commons.math3.distribution.BetaDistribution* class for Beta distribution implementation. To use this class, we need to create a new object of this class for every Beta distribution with an unique (mean, variance) tuple. This means we would need to create a new object for every new input value. As we shall see, a Bayesian Network risk estimator has several inputs. The performance penalty could be high if we need to create a new object for every new input value received. This is especially so for a large production system that serves many user requests at any given time, and each request would provide several input values.

To address this performance issue, our code creates just one *BetaDistribution* object for each numerical input node of the risk estimator. Such an object implements a symmetric standard Beta distribution, $p = q$, $mean = 0.5$. And the p value is determined from a standard deviation value that reflects the perceived uncertainty in the particular input. Let std be the standard deviation, then

$$p = q = (1/(std^2 \times 8)) - 0.5$$

Conceptually, we first shift the distribution so the mean of the shifted distribution equals an input value, then we quantize the shifted distribution over the quanta. In actual implementation, we shift the quanta in the opposite direction by the same amount and then quantize the un-shifted distribution over the shifted quanta. And thus we can reuse the distribution object.

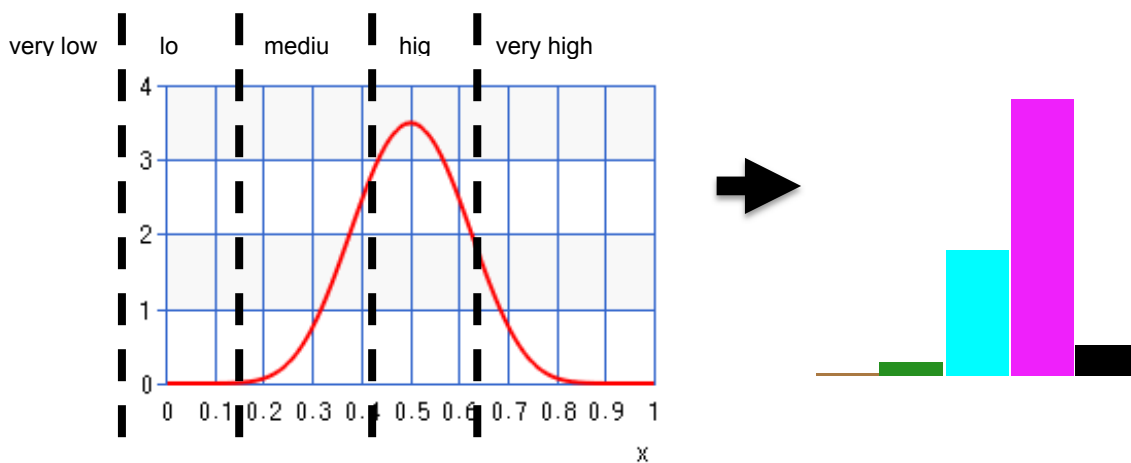


Figure 6.4 *pdf* to *pmf* Conversion with Shifted Quanta

6.3.2 Estimating Risk

We have discussed how we account for the uncertainty in the input to risk estimation, by changing input point values into *pmf*'s. To estimate risk, we need to merge these *pmf*'s to produce a single risk number. The requirement is to preserve the uncertainty in the inputs so the risk number accounts for the uncertainty. And we use *Bayesian Network* (BN) [Dar09] to meet this requirement.

6.3.2.1 Bayesian Network Overview

The BN we constructed is a tree structure. The root of the tree is the final output node that outputs the risk number. The leaves are input nodes which accept input point values and convert the point values into *pmf*'s. Each of the intermediate nodes between the input and output nodes accepts one or more *pmf*'s as inputs and merges these

inputs to an output *pmf*, which is used as input to another node. The output node functions like an intermediate node, but its output has only two quanta : *bad* and *not bad*. And the probability mass assigned to the *bad* quanta is the risk number.

The topology of the BN network should represent our perception on how the inputs relate to and interact with one another to produce the risk estimate. The parameters for a node determine how the inputs are merged. We know of two ways to determine the topology and the parameters for each node.

- *Data Analytics*. If there is a large enough data set on the input/output relationships of the nodes, then analytics such as Markov Chain Monte Carlo (MCMC) [GL06][LLC10] can be used to discover the node parameters [Dar09][Bol11] and even the topology.
- *Expert Opinions*. We pretended to be the experts and used this method because we have no such data set.

It should be noted that it is possible that there are only data sets on a subset of the nodes. In which case data analytics and expert opinions can both be used to construct the BN. This is one nice attribute of using a BN.

Figure 6.5 depicts the topology of the BN we used for estimating risk. The dash-lined components are currently not implemented.

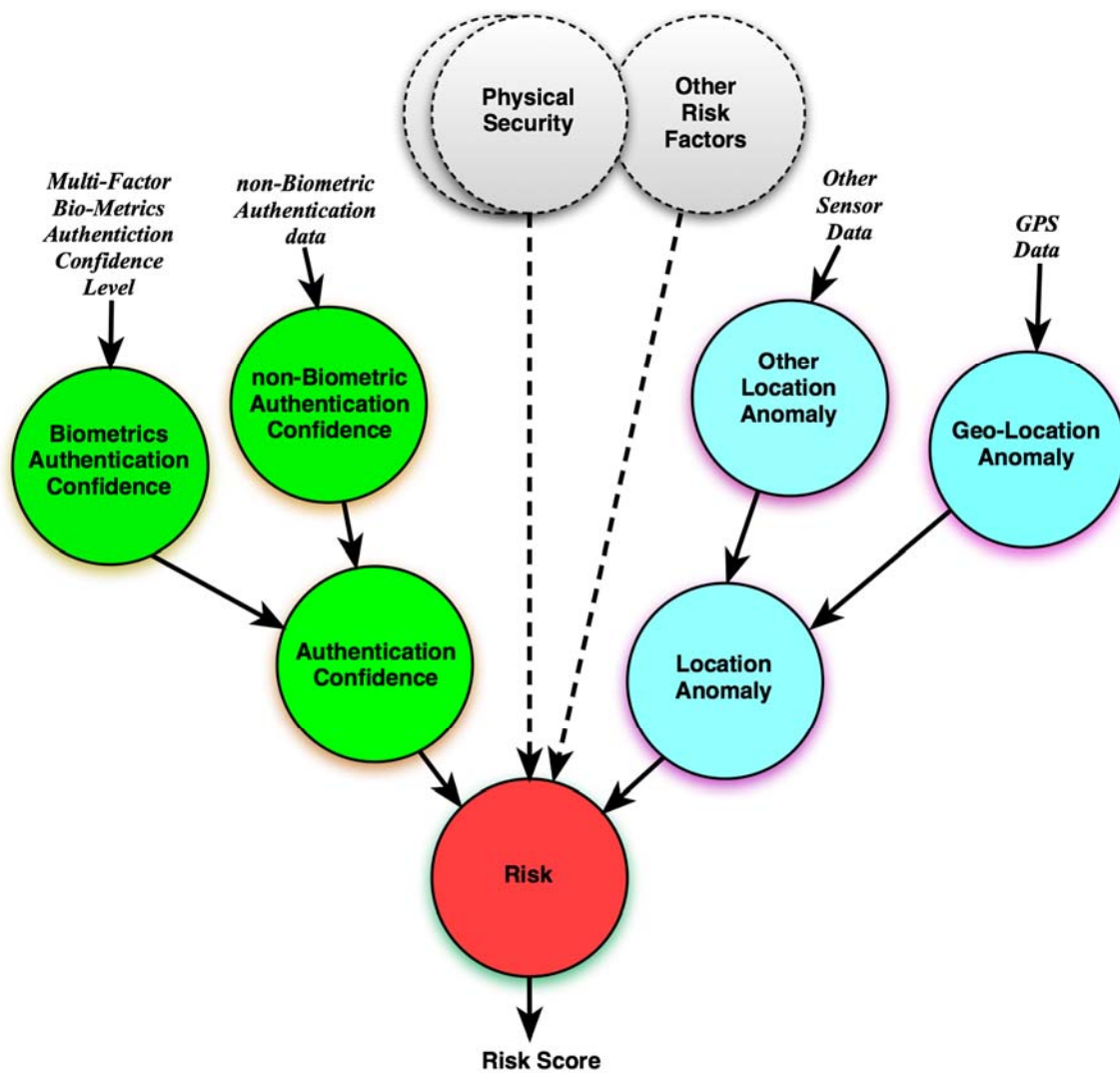


Figure 6.5 Risk Estimator Based on Bayesian Network

6.3.2.3 Using A Bayesian Network Node To Merge Uncertainty

A BN node merges its input *pmf*'s to produce an output *pmf*. The merging process is controlled by its *node parameter*: a *conditional probability table* (CPT).

Each quantum in an input/output *pmf* is considered a state of this input/output. The probability that an input/output is in a particular state is the probability mass assigned to the state (quantum). A condition is a tuple of states from all inputs.

We now explain the CPT with an example.

A BN node with two inputs A and B and an output C

A with m quanta a_1, \dots, a_m

B with n quanta b_1, \dots, b_n

C with l quanta c_1, \dots, c_l

there are $m \times n$ conditions (a_i, b_j) , $1 \leq i \leq m$, $1 \leq j \leq n$

the node's CPT is a table of the conditional probabilities

$$Prob(c_k \mid (a_i, b_j)) , \quad \forall ((a_i, b_j), c_k) , \quad 1 \leq i \leq m, \quad 1 \leq j \leq n, \quad 1 \leq k \leq l$$

Every input condition (a_i, b_j) is mapped to an output *pmf* with l qanta.

The probability that the output is in state c_k is

$$Prob(c_k) = \sum_{i=1}^m \sum_{j=1}^n Prob(c_k \mid (a_i, b_j)) \times Prob((a_i, b_j))$$

We note that:

- In general, a node's CPT establishes a mapping from each and every input condition to the probability of each and every output state under this input condition; in other words, a mapping from each and every input condition to an output *pmf*.

- $Prob(c_k)$
 - enumerates over all possible input conditions; i.e., all possible combinations of input states.
 - Bayesian Network assumes the inputs of a node to be independent of one another, so rewriting $Prob(c_k)$

$$Prob(c_k) = \sum_{i=1}^m \sum_{j=1}^n Prob(c_k | \{a_i, b_j\}) \times Prob(a_i) \times Prob(b_j)$$

- $Prob(a_i)$ and $Prob(b_j)$ are the probability masses assigned to the corresponding input quanta.
- can be generalized to any number of inputs.

The node computes the output *pmf* by computing $Prob(c_k)$ for all k in the range $[1, l]$. We can see that each and every $Prob(c_k)$ covers all possible combinations of input states. This is how a Bayesian Network node merges uncertainty in its inputs and preserves the merged uncertainty in its output.

6.3.2.4 Configuring a Bayesian Network Node

Configuring a Bayesian Network Node is mainly about determining its CPT, and we use the following procedure to do it.

1. *Define the node's output*
 - a. identify the node's *output type*. Intuitively, this type is a concept such as location anomaly or risk.
 - b. translate this output concept into a *finite, continuous range of scores*. The range is usually $[0, 1]$.
 - c. divide the output range into a few *quanta*. We usually use five quanta as we do in processing the BN's input. The one exception is the output of the final output node, the "Risk" node, which has only two quanta: *bad* and *not bad*. The lower and upper bounds of the quanta are determined by our subjective "expert" opinions.

2. identify the node's inputs, their types and quanta.
3. determine how each and every input affects the output. If an input's probability mass moves toward higher quanta, should the output's probability mass move toward higher or lower quanta and by how much?
4. determine how the inputs would jointly affect the output. This usually means determining a rough order of dominance of the inputs' effects on the output.
5. determine the conditional probabilities in the node's CPT. For each and every input condition (a possible combination of input states), determine the conditional probability of each and every output quantum under this input condition. We do this step in the following way.

For each and every input condition,

- a. assign an output score, a point value, to the input condition,
- b. convert the output score to a bell shaped Beta distribution *pdf* over the output score range, with its mean being the output score, and assign a $(p+q)$ value to the distribution.

To slightly over-estimate risk, we use a smaller $(p+q)$ value to have a larger variance if the output score means "good", such as low degree of anomaly. The larger variance assigns more probability mass to the "bad" side of the score range. Likewise, we use a larger $(p+q)$ value to have a smaller variance if the output score means "bad"; so there is less probability mass assigned to the "good" side.

- c. turn the *pdf* into a *pmf* through quantization according to the output quanta determined in step 1, by assigning each quantum a probability mass which is the integration of the *pdf* over the quantum.

The rationale for converting the output score to a *pmf* is that the output score and the input *pmf*'s are all estimates, so we must account for the possibility that the "correct" output score may not be the one we choose.

This step establishes a mapping from input conditions to output *pmf*'s. We note that the mapping can be *many-to-1*: more than one input condition can be mapped to a output *pmf*.

We must emphasize the above procedure is really iterative in nature. We usually needed to go through it a few times get a CPT that looks "right" to us.

We will now give examples to show how this procedure is done.

6.3.2.5 Example: Configuring the CPT of the *Location Anomaly* BN Node

We have two ways to determine/mark a user's location: geo (GPS) data and wireless data. Geo data is a (latitude, longitude) tuple. Wireless data is a set of (MAC address, signal strength) tuples. For WiFi data, MAC address is the MAC address of a WiFi base station seen by the WiFi receiver of the user's mobile device, and signal strength is the station's relative signal strength as seen by the receiver. For Bluetooth, MAC address and signal strength are the Bluetooth MAC address and signal strength of another Bluetooth enabled device seen by the mobile device's Bluetooth receiver. In this example, we will only consider GPS and WiFi data.

We first use geo data points and WiFi data points to build a geo location-time profile and a WiFi location-time profile. Given the GPS data and WiFi data from the user's current location and the current time, each profile answers the question "How anomalous is it for the user to be at this location at this time?" by computing an anomaly score through a comparison of the current data and the profile. Two input nodes convert the geo and WiFi anomaly scores into a geo anomaly pmf and a WiFi anomaly pmf, each pmf is over the range $[0,1]$ and has five quanta *very low*, *low*, *medium*, *high*, *very high*.

For the Location Anomaly BN node:

1. This node has an output type "Location Anomaly" with a score range $[0,1]$.
The score range is divided into five quanta, *very low*, *low*, *medium*, *high*, *very high*, demarcated at 0.0, 0.2, 0.4, 0.8, 0.9, 1.0.
2. The node's two inputs are the geo and WiFi anomaly pmf's described above.
3. The output's probability mass should move in the same direction as those of the inputs.
4. The geo anomaly input should be more dominant than the WiFi anomaly input because GPS coordinates are usually more stable and reliable than WiFi signals.

A geo point's GPS coordinate does not change modulo the GPS errors [GPS13][GPS14][GPSa]; but a WiFi base station may be powered on and off, replaced with another base station with a different MAC address and/or signal strength, moved to another position, etc.

5. There are 25 input conditions. We decided to map each of these conditions to one of five output *pmf*'s. The choice of five output *pmf*'s was obtained through several iterations over steps 1 to 5. We index these *pmf*'s 0, 1, 2, 3, 4; the mean and $(p+q)$ values of their underlying Beta distributions are given in Table 6.1. We can see the mean value moves from being *normal to anomalous* as the index increases; also, the $(p+q)$ value changes from *more uncertainty (larger variance) to less uncertainty (smaller variance)* as the mean value increases.

Table 6.1 Beta Distributions for Location Anomaly Bayesian Network Node

Beta Distribution	mean	$p + q$
0	0.10	60
1	0.25	70
2	0.50	80
3	0.85	90
4	0.95	100

The mapping from input conditions to output *pmf*'s are given in Table 6.2; table entries are indices for the Beta distributions described in Table 6.1. It can be seen that the Geo anomaly input is more dominant than the WiFi anomaly input.

Table 6.2 Input-Output Mapping for Location Anomaly Bayesian Network Node

		WiFi Anomaly Input				
		very low	low	Medium	high	very high
Geo Anomaly Input	very low	0	0	1	2	3
	low	1	1	1	2	3
	medium	2	2	2	3	4
	high	3	3	3	4	4
	very high	3	3	4	4	4

	WiFi Anomaly Input				
	very low	low	Medium	high	very high

Table entries are indices for the Beta distributions in Table 6.1.

6.3.2.6 Example: Configuring the CPT of the Risk BN Node

The Risk node is the output node of our risk-estimation BN. Currently we use two inputs to estimate risk : location anomaly as described above, and authentication confidence with a range of $[0,1]$: how sure we are that the user is who he/she claims to be. In our prototype, the authentication confidence comes from the fusion of several biometric scores of different modalities : face, voice, fingerprint. The range of the fusion score may not be $[0,1]$. We do a pre-processing that maps a fusion score to a subjective probability that the user is *not an impostor*; and use this probability value as the *normalized authentication confidence* value. As usual, we convert the normalized authentication confidence value to a *pmf* with five quanta to be used as an input to the Risk BN node. We note that the normalization mapping would usually be non-linear and depends heavily on the biometrics matching/scoring mechanisms and the fusion model.

For the Risk BN node :

1. This node has an output type "Risk", or "badness", with a score range $[0,1]$.
The score range is divided into two quanta, *not bad* and *bad* demarcated at $0.0, 0.49, 1.0$. The probability mass in the *bad* quantum is the risk score.
2. The node's two inputs are the location anomaly *pmf* and the authentication confidence *pmf* described above.
3. Higher authentication confidence decreases the risk (badness). Higher degree of location anomaly increases the risk.
4. Authentication confidence should be more dominant than location anomaly.
Biometrics is part of what a user is. If a user is at a location/time where/when he is usually not, it may indicate the user is an impostor, but it may also indicate the user is traveling, is in a friend's house, moved to a new office, etc.

Table 6.3 Beta Distributions for Risk Bayesian Network Node

Beta Distribution	mean	$p + q$
0	0.10	70
1	0.15	75
2	0.50	85
3	0.85	90
4	0.95	100

There are 25 input conditions. We decided to map each of these conditions to one of five output *pmf*'s. The choice of five output *pmf*'s was obtained through several iterations over steps 1 to 5. We index these *pmf*'s 0,1,2,3,4; the mean and $(p + q)$ values of their underlying Beta distributions are given in Table 6.3 . We can see the mean value moves from being not bad to bad as the index increases; also the $(p + q)$ value changes from more uncertainty (larger variance) to less uncertainty (smaller variance) as the mean value increases.

The mapping from input conditions to output *pmf*'s are given in Table 6.4; table entries are indices for the Beta distributions described in

Table 6.3. It can be seen that

- the authentication confidence input is more dominant than the location anomaly input,
- the risk decreases as authentication confidence increases,
- the risk increases as location anomaly increases.

Table 6.4 Input-Output Mapping for Risk Bayesian Network Node

		Authentication Confidence Input				
		very low	low	Medium	high	very high
Location Anomaly Input	very low	4	3	2	1	0
	low	4	3	2	1	0
	medium	4	3	3	2	0
	high	4	4	4	2	2
	very high	4	4	4	2	2

Table entries are indices for the Beta distributions in Table 6.3.

6.3.3 Trust-Value-Risk Based Access Control Policy

Traditional access control policies are usually rigid and inflexible. In practical usages, this inflexibility results in (1) poor usability, (2) many exceptions being granted for allowing legitimate requests and work to proceed. To address this issue, risk-based access control (*RBA*) policy was introduced to provide much more controlled flexibility in access control, by taking calculated risk which is prohibited by the traditional access control policies but nonetheless allowed by the exceptions.

Earlier RBA policy models [Chen07] are usually simple, but for many practical application scenarios these simple models lack sufficient details to describe the potentially complex tradeoff and interaction among the value of the resource being requested (the object), the trust placed on the user/requester (the subject) and the context/environment of the request. These earlier RBA policy models lump together value, trust and context into one numerical risk estimate; such an estimate could be associated with many access requests of very different combinations of value, trust and context. The flexibility and clarity in specifying risk-based policy is restricted, which in term limits how much flexibility a RBA policy can provide.

The Trust–Value–Risk (TVR) based policy model aims to have *trust, value and risk* as three separate entities. For a particular access request, its risk, quantitatively estimated from the context, is used to modulate the trust on the subject, and the modulated trust and the value of the requested resource are then used to reach an access control decision. The decoupling of trust, value, and risk/context allows more flexibility and clarity in specifying risk–based access control policies. Also, since risk is estimated from the context of the request, the policy in effect is context–aware. In other words, the trust placed on a user is adjusted/modulated by the context. The context can include any factor that is considered relevant to security, such as physical location, security of the location, time, user behavior profile, security of the device the user is using, access history of the user, etc. Therefore this policy model entails a flexible access control policy that is context aware and context adaptive.

Making resource value a separate explicit entity in the policy also makes it *easier to handle access requests in an emergency*. In an emergency, the resource values can be adjusted lower accordingly to allow more access so as to handle the emergency. In other words, the values/importance of resources is diminished when being compared with the need to handle an emergency. Alternatively the modulated trust can be increased to address the urgent need in an emergency. In our prototype, the implementation of this emergency-handling concept could be viewed as either decreasing resource value or increasing trust on the subject. More details will be given later.

The TVR model introduces the notion of *intrinsic trust*. This is the trust placed on a user independent of any risk/context. In other words, this is the trust placed on a user when there is no risk from the context. It is this trust that is to be modulated by risk.

An important concept in the TVR model (and some other RBA policy models) is risk mitigation. If the current risk of an access request is too high, the request may still be granted if the risk can be mitigated down to an acceptable level. The risk mitigation concept entails non-binary decision options: *allow, deny* and *mitigate*. In general, there can be more than one mitigate option since there are many risk mitigation techniques, such as sandboxing, detailed auditing, increasing authentication confidence, privilege reduction, etc.. A risk mitigation measure changes some part of the context of the request: increased authentication confidence, auditing activated, etc..

In summary, TVR aims to mitigate just enough so as not to overburden the users or the system. In practice, it is nearly impossible to do just enough so the practical objective is to mitigate a little more than enough and not too much more.

Figure 6.6 shows the flow of making an access control decision using the TVR model.

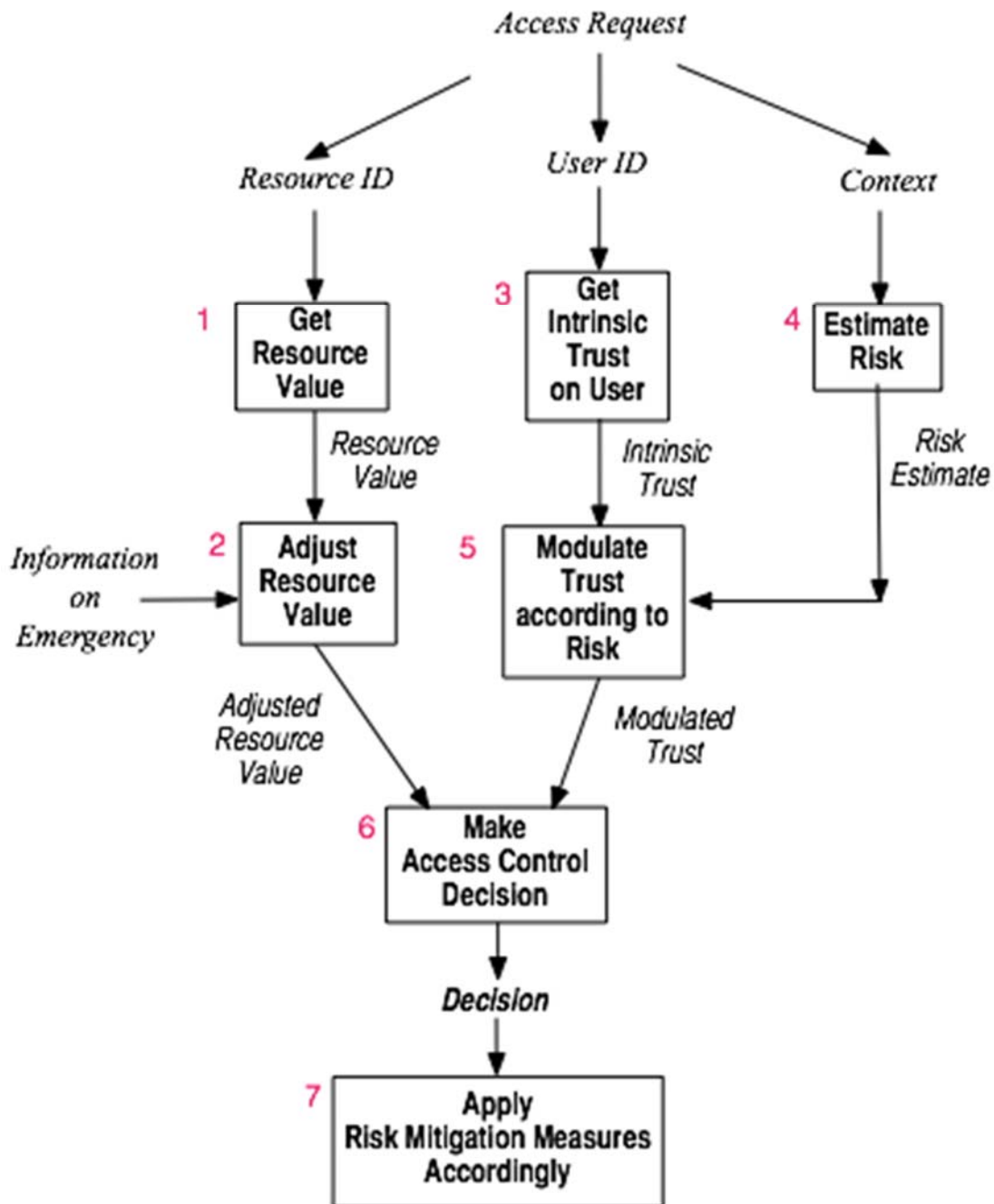


Figure 6.6 Making an Access Control Decision Using the TVR Model

Another important consequence of risk mitigation is the multi-round decision making concept when handling an access request. If an access request's risk is too high, the access control mechanism applies some risk mitigation measure. If the risk is still too high after applying the mitigation measure, apply more risk mitigation measures and see

if the risk is still too high. This “mitigate then check” step can be repeated until (1) the risk becomes acceptable, or (2) risk mitigation measures are exhausted, or (3) too many rounds of this step have been gone through. Figure 6.7 shows the flow of multi-round decision making with the TVR model.

It should be noted that it is not necessary to explicitly specify the “deny” decision option in the access control policy if multi-round decision making is used. A request will be denied if the risk mitigation measures are exhausted or too many rounds have been gone through. Depending on the particular application scenario, the “deny” option can still be specified if it is deemed that there are cases where the risk is too high and the request should be denied without trying to mitigate the risk.

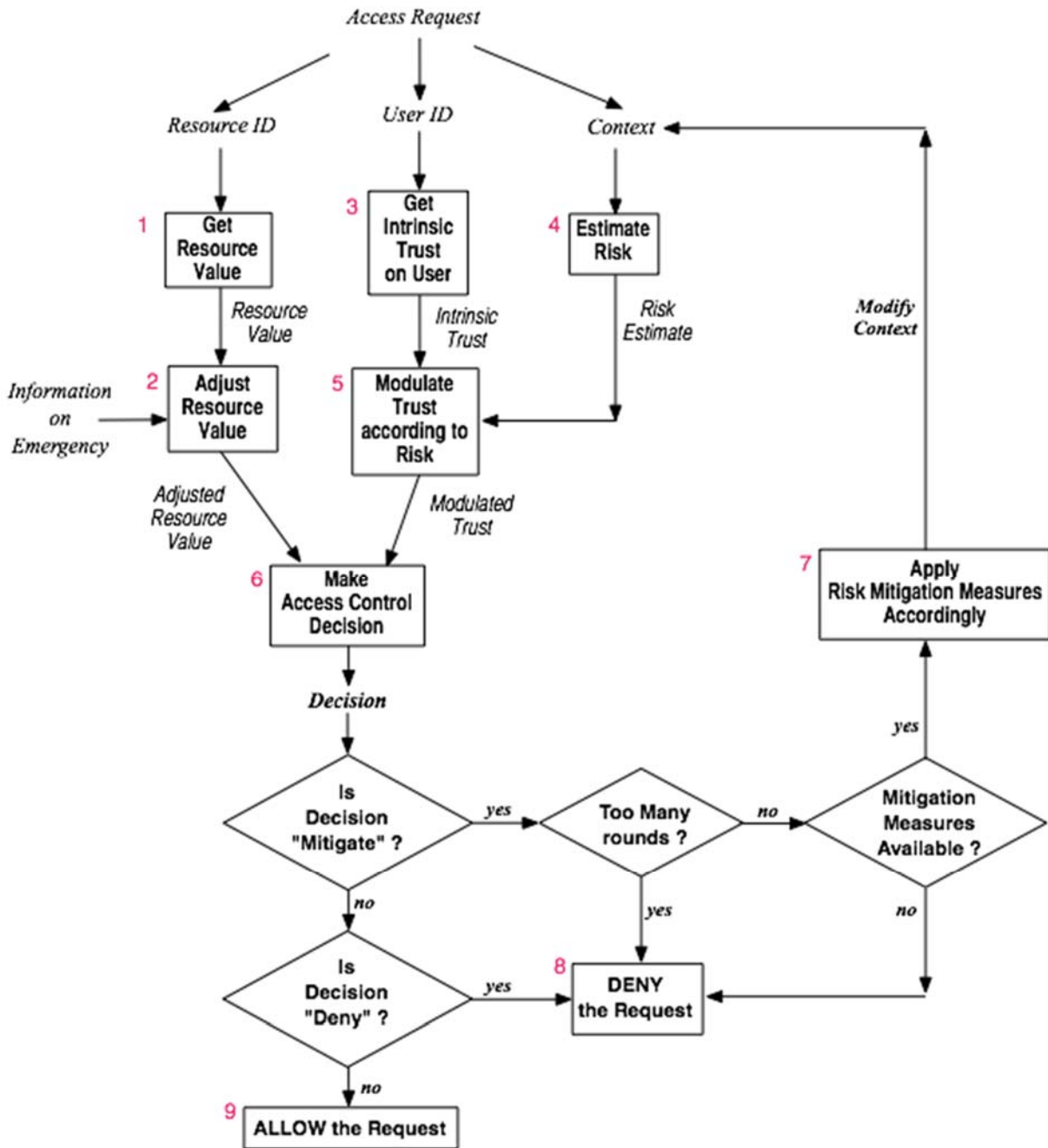


Figure 6.7 Multi-Round Decision Making with TVR Policy

Figure 6.8 depicts the implementation of the TVR model in our prototype.

- Policy rules are *decision boxes* placed along the trust and value axes.
- *Intrinsic trust* on a user is defined by a *Intrinsic Trust Curve*
- *Resource Value Adjustment* can be done by horizontally shifting the decision boxes, or shifting the trust curve in the opposite direction. In our implementation, we allow the trust curve to be right-shifted to increase the trust on a user during an emergency. Right-shifting the trust curve to increase trust is equivalent decreasing the resource value by the magnitude of the shift.
- Trust is modulated by down-shifting the intrinsic trust curve according to risk.
- The decision box where the modulated trust curve and the value line intersects gives the decision.

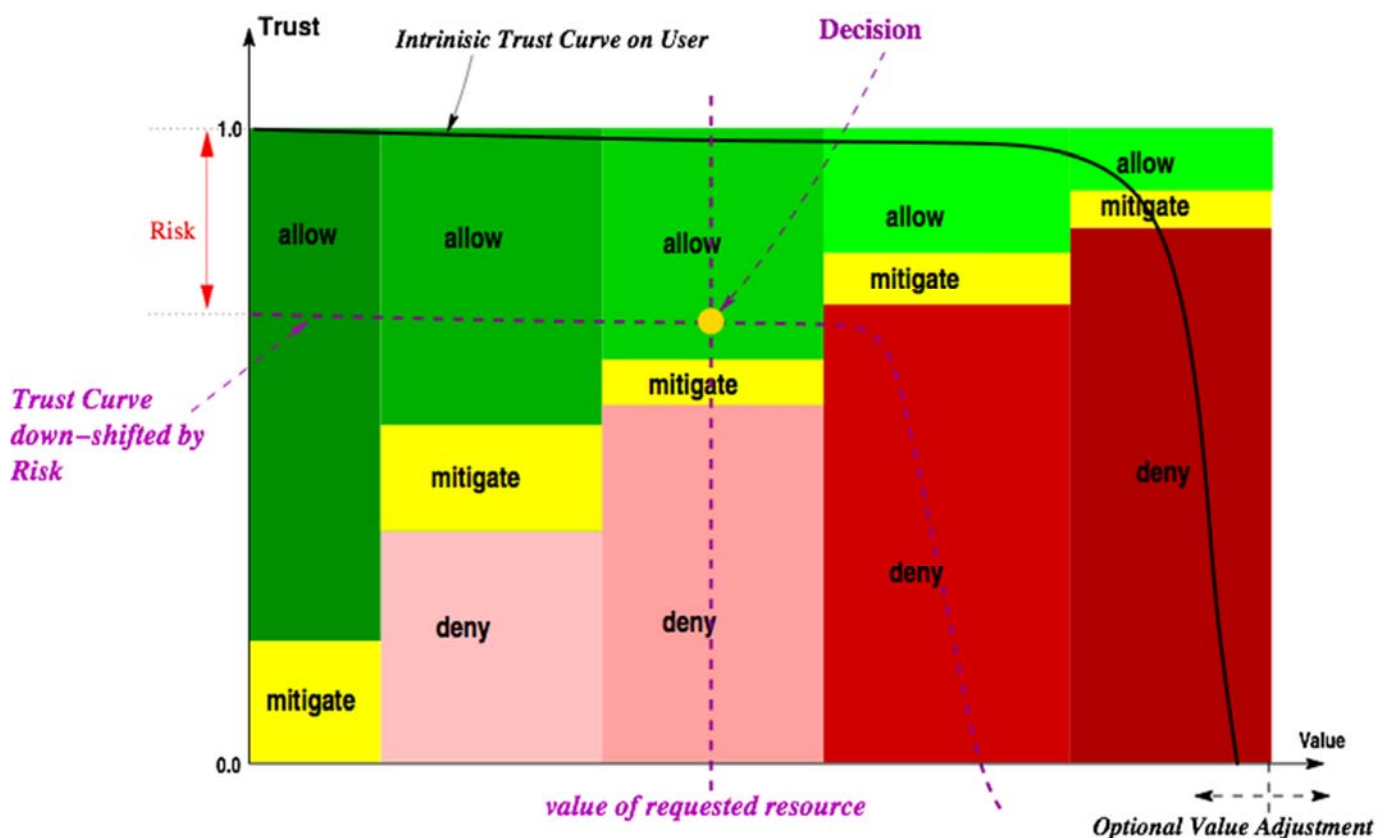


Figure 6.8 Our Implementation of the TVR Policy

6.3.5 Continuous Learning of Behavior Profiles

A behavior profile models certain aspect(s) of a user's normal behavior. For example, we build a user's time-location profile to see if a user is usually at the location at the time when he/she makes an access request. In general, a behavior profile is a good way to detect a possible impostor and therefore is a good tool when estimating risk. By comparing the user's current behavior against his/her profile, an anomaly score is generated which is used as an input to the risk estimator.

In practice, a user's behavior may change over time so there is a need to continuously detect and learn/discover new behavior patterns. Also, the system may not have a behavior profile for a newly enrolled user. In which case all the user's behaviors are new, and again there is a need to learn these new behavior patterns.

To detect what is new, we must first recognize what is old and what is new. Intuitively,

- what is considered *normal* by the existing profiles is *old*; and
- what is considered *anomalous* by the existing profiles is *new*.
- Everything is anomalous/new if there is no profile, or we can view this case as having an empty profile that does not cover any behavior patterns.

Our technique for continuous learning of behavior profiles includes three big steps :

1. detecting when there are new behavior patterns in the data. Data are usually collected and stored when the user makes an access request. We note that in this step we do not try to discover what the new patterns are, but only to *recognize signs for the emergence of new behavior pattern(s)*.
2. using analytical means to discover the new patterns from collected data. We note that the analytics may also discover old patterns that are still in the data.
3. building new behavior profiles based on the discovered patterns. After the profiles are built, they are used for anomaly scoring.

There is another important requirement when dealing with emerging new behavior patterns: *producing reasonable anomaly scores on data points from new behavior patterns even before the patterns are learned and covered by the profile*. This requirement is for usability. Data analytics need enough data points to identify a new pattern and to build new profiles; usually at least tens of data points are needed. Before the new profiles are built, the existing profile will produce a high anomaly score for a

data point from the new pattern. A high anomaly score would likely result in a high risk score, which may result in the denial of a legitimate request or putting extra burden on the user to mitigate the perceived risk. A human user would not tolerate such poor usability and wait for the system to collect enough data points. So we need to bridge the gap between the human demand for usability and the machine's need for enough data points. It turns out that bridging this gap is a natural by-product of recognizing signs for the emergence of new behavior patterns.

6.3.5.1 Recognizing the Signs of Emerging New Behavior Patterns

Enough data points are needed to form a pattern. So we assign a *newness score* to each data point and accumulate these newness scores. When the accumulated score is above a predefined threshold, we say there is most likely a new pattern in the data and start some data analytics in the background to discover/learn the new patterns and build new profiles.

To compute the newness score, we created a non-profile based mechanism to compute another anomaly score for a data point; this score is different from the anomaly score computed using the profile. This mechanism maintains a limited-size cache of recently collected data points and treats a newly received data point as a fuzzy set.

For this newly received data point and its corresponding fuzzy set :

1. for each and every *cached data point*
 - i. compute the distance/proximity between the cached point and the newly received point using a pre-defined distance/proximity measure.
 - ii. transform the distance to a value in the range $[0, 1]$ using a pre-defined, *monotonically decreasing* function. This value is the *fuzzy set membership* of the cached point. *The closer the cached point is to the newly received point, the larger the membership.*
2. compute the *fuzzy union* membership of the cached points' fuzzy memberships. The closer the cached points are to the newly received point, the larger the union membership. The larger the number of cached points that are close to the newly received point, the larger the union membership.

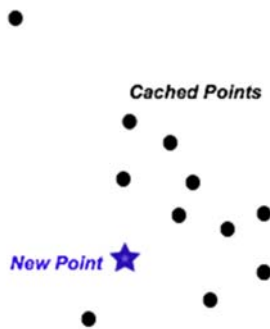


Figure 6.9 Data Point Cache for Learning Example

3. $anomaly\ score = (1 - fuzzy\ union\ membership)$. This anomaly score is called the *cache score* for the newly received point. The anomaly score computed using the profile (maybe empty) is called the *profile score*. Figure 6.10 is a conceptual diagram that shows how the cache score and profile score changes as points from emerging new behavior patterns are received over time, before the new patterns are learned and covered by the profiles. It can be seen that the cache score decreases as more points from the new patterns are received; but the profile score remains high since the new patterns are not covered by the profile yet. As the difference between profile score and cache score becomes larger, the sign for emerging new patterns becomes larger as well. If a point fits the existing profile but is not close to any cached point, then the point's profile score would be low and its cache score would be high; but Figure 6.10 does not show such a case.

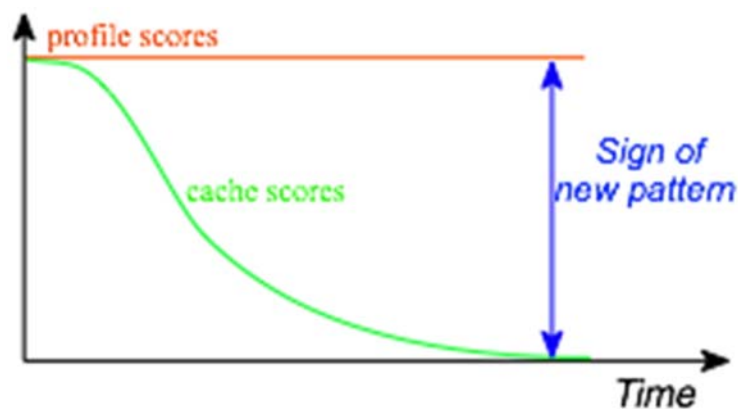


Figure 6.10 Profile and Cache Scores Over Time

The newness score is derived by comparing the cache score and the profile score:

- If $profile\ score < cache\ score$ then $newness\ score = 0.0$
The condition ($profile\ score < cache\ score$) indicates that the newly received point is similar to past points covered by the profile.
- otherwise $newness\ score = newnessMap(profile\ score - cache\ score)$,
where $newnessMap$ is a monotonically increasing mapping with range $[0, 1]$.
Figure 6.11 shows the mapping we used. This mapping is meant to de-emphasize smaller ($profile\ score - cache\ score$) differences and to emphasize larger ones. We feel smaller differences are insignificant because profile and cache scores are both semi-subjective.

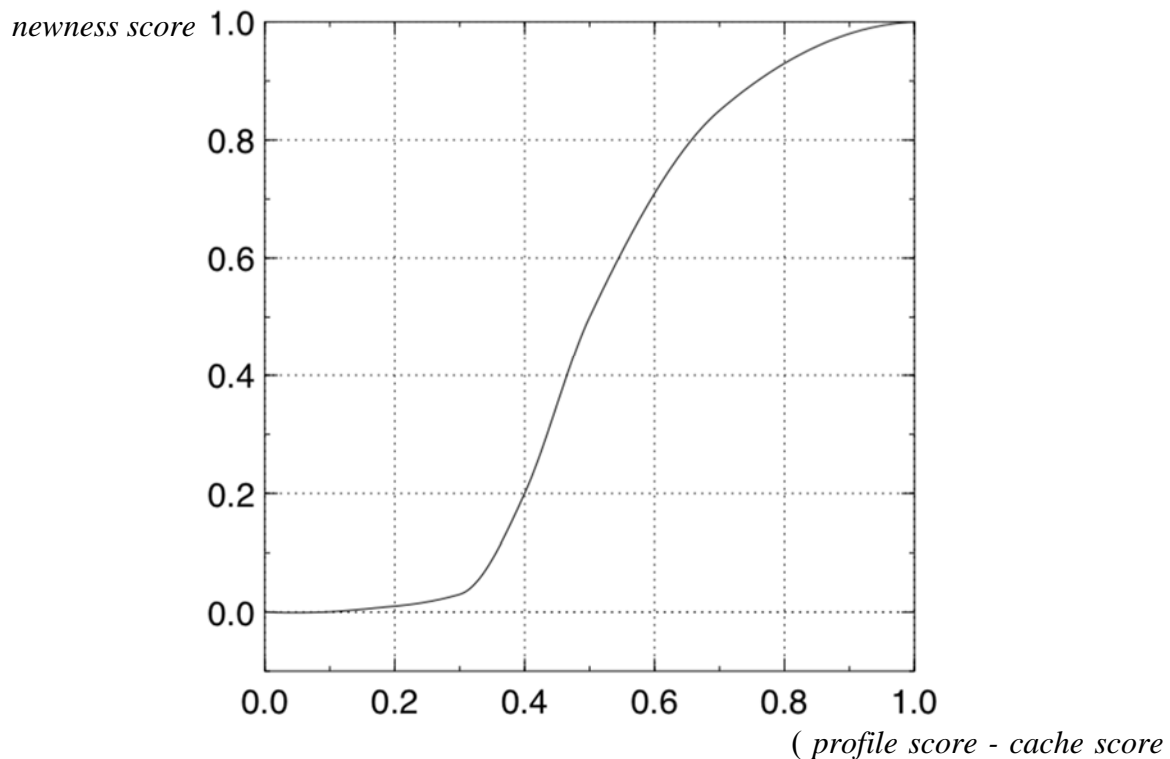


Figure 6.11 Newness Map

The newness scores are accumulated using simple summation. When the sum reaches a pre-defined threshold, an analytics application is started in the background to learn new behavior patterns in the data. A new profile is built according to the learned patterns. We choose the threshold to be 80. This choice is based on some trials and errors. This threshold value provides at least adequate number of data points for the analytics to work on.

6.3.5.2 Bridging the Gap Between Human and Machine

Human users are usually impatient, and demands quick and easy access to what he wants. The security mechanism of a system needs to balance such demand and the need to keep risk within an acceptable level, while trying not to in-convenience the users to much. Part of this balancing act is trying to mitigate just enough (or a little more than enough) by using the TVR policy to conduct risk-based authorization. This part *adapts to the user's context* when he makes an access request.

The learning mechanism is meant to adapt to a user's changing/new behavior. This adaption means the user's new behavior looks more and more normal to the system (machine) as more data points from this new behavior are collected; in other words, the anomaly score becomes lower and lower as more data points from this new behavior are collected. To implement this adaption, we fuse the profile score and the cache score to produce an anomaly score in the following way:

- If the profile score is less than or equal to the cache score, then use the profile score as the anomaly score.
- If the profile score is larger than the cache score, then the anomaly score should be somewhere between the cache score and the profile score; we designed and implemented a heuristics-based algorithm to fuse the cache score and the profile score to produce the anomaly score. The algorithm has the following properties:
 - The anomaly score will be in the range [cache score, profile score).
 - The profile score will pull the anomaly score higher.
 - The cache score will pull the anomaly score lower.
 - The smaller the cache score, the heavier the pull it exerts.

The algorithm is given in Figure 6.12.

Input : *profile score* and *cache score*, $0 \leq \text{cache score} < \text{profile score} \leq 1$

Let *bdist* be a standard Beta distribution with the following property:

$$p + q = 100$$

$$p = ((\text{cache score})/(\text{profile score})) \times (p + q)$$

Let r be the 95% quantile for *bdist* such that $\int_0^r \text{bdist}(x)dx = 0.95$ and let $v = (\text{profile score}) \times r$

Then $\text{anomaly score} = v$, $\text{if } v > (\text{cache score})$
 $\text{anomaly score} = \text{cache score}$, $\text{if } v \leq (\text{cache score})$

Figure 6.12 Merging Profile Score and Cache Score

As the cache score becomes smaller, the mean value of *bdist*, $(p / (p + q))$ becomes smaller. Also, the variance of *bdist* becomes smaller as the cache score becomes smaller. Therefore an ever heavier pull toward 0 is exerted on the anomaly score as the cache score becomes smaller.

6.3.5.3 Experimental Validation of the Learning and Scoring Mechanisms

We used a data set of 40318 time-stamped geo data points (latitude and longitude) generated by simulation to test the learning and scoring mechanisms. The story line is about a doctor who lives in Katonah, Westchester County, New York, practices at the Westchester Medical Center at Valhalla, New York and a clinic in Yorktown Heights, New York, shops at a supermarket in Yorktown Heights, and rides on the Metro North train to Grand Central Terminal in New York City during weekends. The data set covers a four-week period. We assumed the doctor is a creature of habit and exhibits repeating location-time pattern on a weekly basis.

We build a browser-based test client using Java Script. This client feeds access request, including the geo coordinate, to the server in which the RBA component is responsible for doing risk-based authorization and learning. Since our goal is to test the learning mechanism, we artificially set the authentication confidence to be high. So the

client always passed the authentication and its location-time behavior pattern was learned.

Figure 6.13 and Figure 6.14 are two screen shoots of the test client. It should be noted that a new data point is pushed on to the top of the table.

Figure 6.13 shows the early stage of the test, when the profile was not learned yet. It can be seen that the profile score remained high (1.0) but the cache score, the anomaly score and the risk score became smaller as more data points are learned.

Figure 6.14 shows the change in scores when the profile was learned. The transition happened between points 97 and 98. The profiles score changed from 1.0 to 0.0 and the anomaly score changed to 0.0 after the 1st profile was learned.

After running through all the 40318 data points, the learning mechanism learned the location the doctor frequents and the roads/paths he usually travels. Figure 6.15 shows these locations and paths on a map.

before profile is learned,
profile score = 1.0

Score History

Color Legend : Profile Score Cache Score Anomaly Score Risk Score

Point	Index	Location	Profile Score	Cache Score	Anomaly Score	Risk Score
21		Home	1.0000000000000000	0.1094189891315123	0.1646157148934402	0.2411999994791166
20		Home	1.0000000000000000	0.1215766545905692	0.1790725241261650	0.2423437394404792
19		Home	1.0000000000000000	0.1350851717672992	0.1949262943675544	0.2440231541270412
18		Home	1.0000000000000000	0.1500946352969991	0.2123164379874607	0.2464800048925833
17		Home	1.0000000000000000	0.1667718169966657	0.2313958467786752	0.2500486761478010
16		Home	1.0000000000000000	0.1853020188851840	0.2523319980859286	0.2551758029495618
15		Home	1.0000000000000000	0.2058911320946489	0.2753080633259310	0.2624309572158336
14		Home	1.0000000000000000	0.2287679245496099	0.3005239925481802	0.2724969875794770
13		Home	1.0000000000000000	0.2541865828329000	0.3281974995424908	0.2861263201051883
12		Home	1.0000000000000000	0.2824295364810000	0.3585648520127476	0.3040527091856812
11		Home	1.0000000000000000	0.3138105960899999	0.3918812956059009	0.3268601272846940
10		Home	1.0000000000000000	0.3486784400999999	0.4284208445230416	0.3548297049345494
9		Home	1.0000000000000000	0.3874204889999999	0.4684749808577814	0.3877964139680691
8		Home	1.0000000000000000	0.4304672099999999	0.5123494716549123	0.4250205992788936
7		Home	1.0000000000000000	0.4782968999999999	0.5603578534104161	0.4650062174379146
6		Home	1.0000000000000000	0.5314409999999999	0.6128087670705612	0.5051750438405104
5		Home	1.0000000000000000	0.5904900000000000	0.6699812030136642	0.5415923653670505
4		Home	1.0000000000000000	0.6560999999999999	0.7320736721268383	0.5696262088114823
3		Home	1.0000000000000000	0.7290000000000000	0.7990890055398622	0.5864588379308997
2		Home	1.0000000000000000	0.8100000000000001	0.8705226038138975	0.5937626371477625
1		Home	1.0000000000000000	0.9000000000000000	0.9441678211585370	0.5961753551669928
0		Home	1.0000000000000000	1.0000000000000000	1.0000000000000000	0.5968242880959416

new points
pushed on top

cache score, anomaly score and risk
score drop as learning progresses

Figure 6.13 Learning Screen Shot - Before Profile is Learned

after profile is learned
profile score becomes 0.0

Score History

Color Legend : Profile Score Cache Score Anomaly Score Risk Score

Point Index	Location	Profile Score	Cache Score	Anomaly Score	Risk Score
106	Home	0.0000000000000000	0.0423911582752161	0.0000000000000000	0.2386485197800671
105	Home	0.0000000000000000	0.0423911582752161	0.0000000000000000	0.2386485197800671
104	Home	0.0000000000000000	0.0423911582752161	0.0000000000000000	0.2386485197800671
103	Home	0.0000000000000000	0.0423911582752161	0.0000000000000000	0.2386485197800671
102	Home	0.0000000000000000	0.0423911582752161	0.0000000000000000	0.2386485197800671
101	Home	0.0000000000000000	0.0423911582752161	0.0000000000000000	0.2386485197800671
100	Home	0.0000000000000000	0.0423911582752161	0.0000000000000000	0.2386485197800671
99	Home	0.0000000000000000	0.0423911582752161	0.0000000000000000	0.2386485197800671
98	Home	0.0000000000000000	0.0423911582752161	0.0000000000000000	0.2386485197800671
97	Home	1.0000000000000000	0.0423911582752161	0.0797601499358158	0.2387914718178093
96	Home	1.0000000000000000	0.0423911582752161	0.0797601499358158	0.2387914718178093
95	Home	1.0000000000000000	0.0423911582752161	0.0797601499358158	0.2387914718178093
94	Home	1.0000000000000000	0.0423911582752161	0.0797601499358158	0.2387914718178093
93	Home	1.0000000000000000	0.0423911582752161	0.0797601499358158	0.2387914718178093
92	Home	1.0000000000000000	0.0423911582752161	0.0797601499358158	0.2387914718178093
91	Home	1.0000000000000000	0.0423911582752161	0.0797601499358158	0.2387914718178093
90	Home	1.0000000000000000	0.0423911582752161	0.0797601499358158	0.2387914718178093
89	Home	1.0000000000000000	0.0423911582752161	0.0797601499358158	0.2387914718178093
88	Home	1.0000000000000000	0.0423911582752161	0.0797601499358158	0.2387914718178093
87	Home	1.0000000000000000	0.0423911582752161	0.0797601499358158	0.2387914718178093
86	Home	1.0000000000000000	0.0423911582752161	0.0797601499358158	0.2387914718178093
85	Home	1.0000000000000000	0.0423911582752161	0.0797601499358158	0.2387914718178093
84	Home	1.0000000000000000	0.0423911582752161	0.0797601499358158	0.2387914718178093
83	Home	1.0000000000000000	0.0423911582752161	0.0797601499358158	0.2387914718178093

new points
pushed on top

anomaly score is 0.0 for
points covered by learned

Figure 6.14 Learning Screen Shot - After Profile is Learned

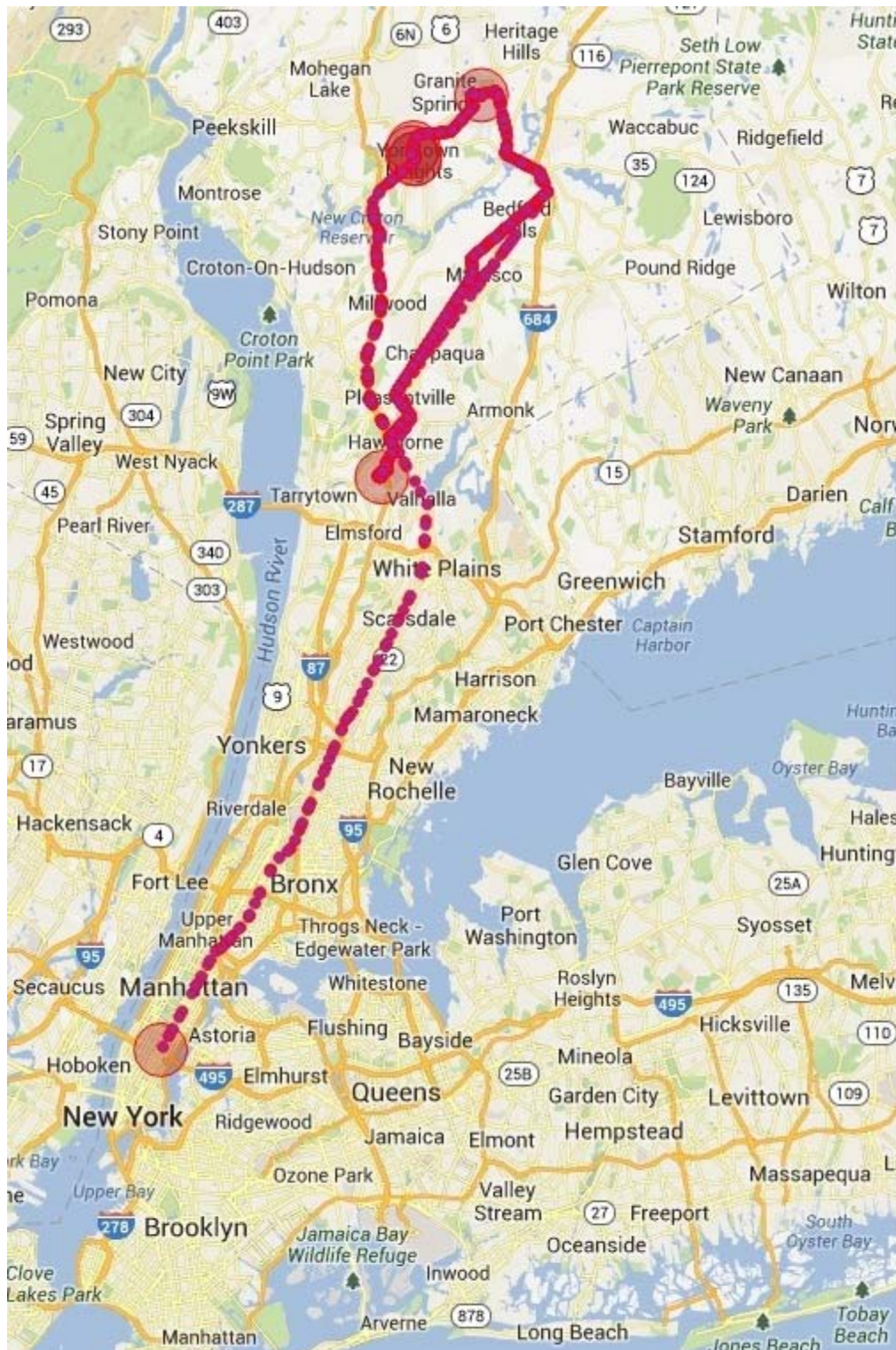


Figure 6.15 Learned Geo Locations and Paths

6.4 Results and Discussion

We have built a reasonable and working RBA prototype that can be deployed and field tested. We believe the key factor in successfully building the prototype is the understanding that risk is about uncertainty in the future and therefore risk estimate may never be very accurate, but reasonable risk estimate can be produced in a principled and systematic way, and these estimates can be used to make reasonable access control decisions.

Our prototype includes

- a risk estimator that estimates risk of an access request from its context while accounting for the uncertainty in input and risk modeling,
- a policy model that makes risk-based, context-aware access control decisions based on the risk estimate, the trust on the user, and the value/sensitivity of the requested resources.
- a learning mechanism/model that can learn user behavior patterns and profiles from scratch, and produces reasonable anomaly and risk scores while learning. We used time-stamped geo-location data to validate and to demonstrate this mechanism.

It should be emphasized that this ability to learn and score from scratch means that the RBA prototype can be used on newly enrolled users of whom there are no known behavior patterns and profiles. This ability makes RBA practical for human users of the system.

6.5 Conclusions

We demonstrated the feasibility that reasonable, numerical risk estimate can be produced to make reasonable access control decisions in practice, all in a principled and systematic manner.

The key lesson is that reasonable and useful risk estimates are possible, as long as we accept the fact that uncertainty/risk is inevitable, and the goal should be to manage, and not to eliminate the risk.

Our prototype is validated in a lab setting. It needs more validation, refinement and augmentation through practical deployments. We hope these deployments will provide more and different kinds of data for us to analyze and learn.

6.6 Recommendations

We recommend that more data be made available for R&D on mobile security. Data relevant to traditional information security are needed. Since mobility implies ever changing physical environments and a physically secure protective boundary can no longer be assumed, data that are relevant to the physical environments, such as the crime rate of a municipal area, type of a location, etc, are also needed.

7.0 MOBILE CLIENT AND SECURITY SERVICES

This section describes the high-level design of the mobile client component of our architecture and discusses the security offering of the system design. One major goal of our design is to balance security with usability. With this goal in mind, we make additional usability considerations in our system design to make it more usable for the app developers and system administrators. We hereby discuss details of the methods and techniques that we leverage to build the client-side framework in order to realize our goals.

7.1 *Methods*

In order to achieve improved usability in our system design, we make two major considerations. First, our design supports easier app development and administrative control. With ever reducing time-to-market requirements for mobile app development, this feature becomes increasingly important for making our design more desirable. Second, our design minimizes the real-world deployment hurdle by supporting multiple robust integration options. Such options include flexible integration support for legacy applications with no to minimum application modifications.

To realize these goals and considerations, we designed and developed the following elements as part of our end-to-end system offering:

- Platform-independent client-side application design: Instead of developing the application for each target platform (iOS and Android), we developed our applications to have a hybrid design in which a major component of the application is written using platform-independent web technologies (i.e., HTML, JavaScript and CSS) and subsequently packaged and executed on the mobile device as platform-specific native application.
- Modular application development: We built our design to be modular and extensible in order to incorporate new (and future), biometric and non-biometric authentication credentials. Additionally, our design allows easy replacement of one biometric engine implementation with another. The modular client design correspondingly enables replacement of credential capture modules without modification to the underlying core framework.
- Robust and extensible client-server communication primitives: The communication design between the client and server exposes a set of primitives that our authentication system builds on in order to implement its core client-server protocol. Such primitives enable the system to be extensible and robustly

adapting to any new requirements (e.g. addition of a new credential as an authentication factor).

- **Passive context-based authentication:** We collect and leverage raw data from mobile device sensors to define context for passive authentication. We utilize geolocation sensor data to model user's normal behavior in correlation with time and location. We use other sensors, such as accelerometer, to determine if the user has lost control of his device.
- **Inline and out-of-band authentication:** Different applications that seek advanced authentication using our system potentially have different usability and security requirements. On the one hand, they desire smooth integration with minimum modifications to the existing apps, and on the other hand, they seek a consistent and smooth user experience between their own application and our authentication system. We support out-of-band authentication mode to separate the business application from our authentication system thus enabling easier integration and providing added protection to the authentication system from the application. Inline mode enables close integration of the authentication into the business application by offering a set of simple interfaces for the business application to explicitly seek authentication in its transaction flow.

7.2 Assumptions

7.2.1 Hybrid Application Design

To facilitate rapid application development for mobile and to bridge the gap between different development platforms (such as iOS and Android), new programming frameworks have emerged to enable a “develop once and deploy everywhere” paradigm. In this paradigm, apps are developed using web technologies of HTML, JavaScript and CSS and subsequently wrapped with the hybrid framework, that, in turn, enable automatic deployment of these apps as native packages onto multiple platforms (Figure 7.1).

These *hybrid* application frameworks render the application in a chrome-less browser UI component typically called a WebView. This UI component is available in most mobile frameworks with slightly different names, such as Android WebView, and iOS UIWebView. The HTML, CSS and JavaScript code base runs in WebView that is wrapped in a native app. The underlying frameworks expose a set of APIs to the applications in order to perform native operations, such as accessing the device's sensors. Several such hybrid frameworks are available, such as PhoneGap and Cordova.

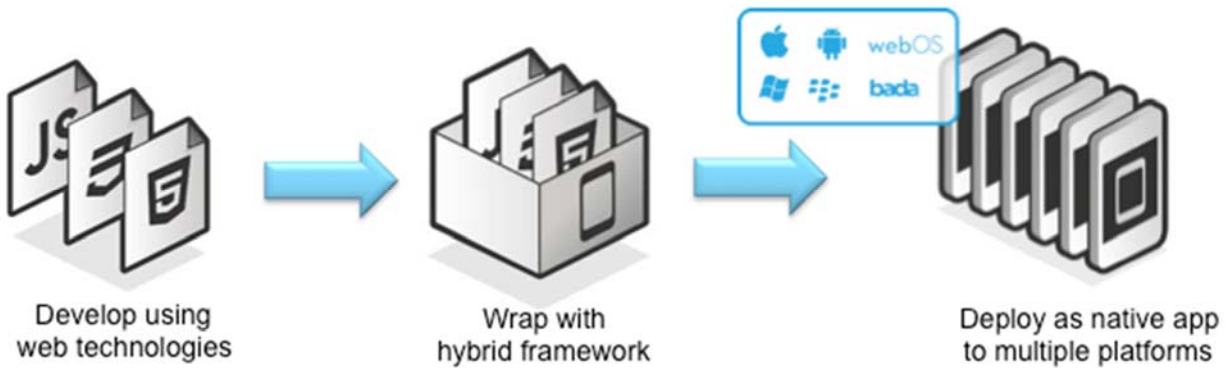


Figure 7.1Development lifecycle for hybrid applications

For our implementation, we used PhoneGap as our proof-of-concept framework. The concepts are readily portable to other frameworks, whether native or hybrid. The use of a hybrid design for our application enabled us to minimize our effort in supporting our two target platforms: Android and iOS.

7.2.2 Context-based Security

Modern mobile devices come fitted with a variety of sensors that act as a rich data source for determining a device's situational, environmental and behavioral context. For example, the presence of a large number of Bluetooth devices (determined by neighborhood scanning using the Bluetooth sensor) and/or Wifi networks represents a crowded location. The mobile context can be defined by the status of different environment/sensor variables (e.g. location, time and movement), the presence of other devices, a particular interaction between the user and the phone, or a combination of these.

We leverage mobile context to determine situational risk for *passively* making step-up/step-down authentication decisions. For example, a crowded location is riskier than a non-crowded one, and therefore additional authentication may be required. Note that different apps may have different security requirements and functionalities, and define their own particular authentication policies, even under the same circumstances. For instance, the online banking apps have higher security requirements than the social network apps, and hence might require more authentication confidence in crowded location than the social networking apps.

In our work, we also leverage mobile context to model and determine risky patterns and situations. More specifically, we leverage the accelerometer sensor readings for *user presence detection*, i.e., to determine if a user is in control of the mobile device. This determination can help the business application to take proactive steps to provide additional protection to its data and state. For example, assume a user is viewing a company's confidential information on his phone at a restaurant and forgets the phone on the table. Our detection mechanism would determine that the user has lost control of the device and can automatically sign out the user from the application, thus protecting the confidential information from public or targeted view. Alternative solutions could automatically lock the device on receipt of the loss-of-control trigger.

If the accelerometer reading is zero for a specific period of time, it represents idleness or lack of user presence. However, raw accelerometer readings on mobile devices are very sensitive and prone to environmental noise that could potentially lead to false triggers in user presence detection. This sensitivity varies based on the phone hardware. Therefore, we have a configurable parameter to discount environmental noisy movements. In future, this parameter could be automatically learned for each device hardware and user movement behavior. Additionally, every user or business application has different security requirements, so we enable them to configure the idleness period (where accelerometer reading is close to zero) after which the loss-of-control trigger is invoked.

7.2.3 Flexible Integration Options

One of our usability design goals is to enable easy integration of our solution into a variety of business applications with different usability and security requirements. Some applications desire close integration of the authentication solution with the business logic while others desire minimum to no modifications to their existing apps. To satisfy these contrasting goals, we support both inline and out-of-band integration of our authentication solution with the business. In the inline mode, the step-up authentication requirement is received by the business application in its transaction flow and it, in turn, calls the authentication application to perform the required authentication steps. In out-of-band mode, the business application is oblivious to the authentication requirements and instead such requirements are received out-of-band by the authentication application itself.

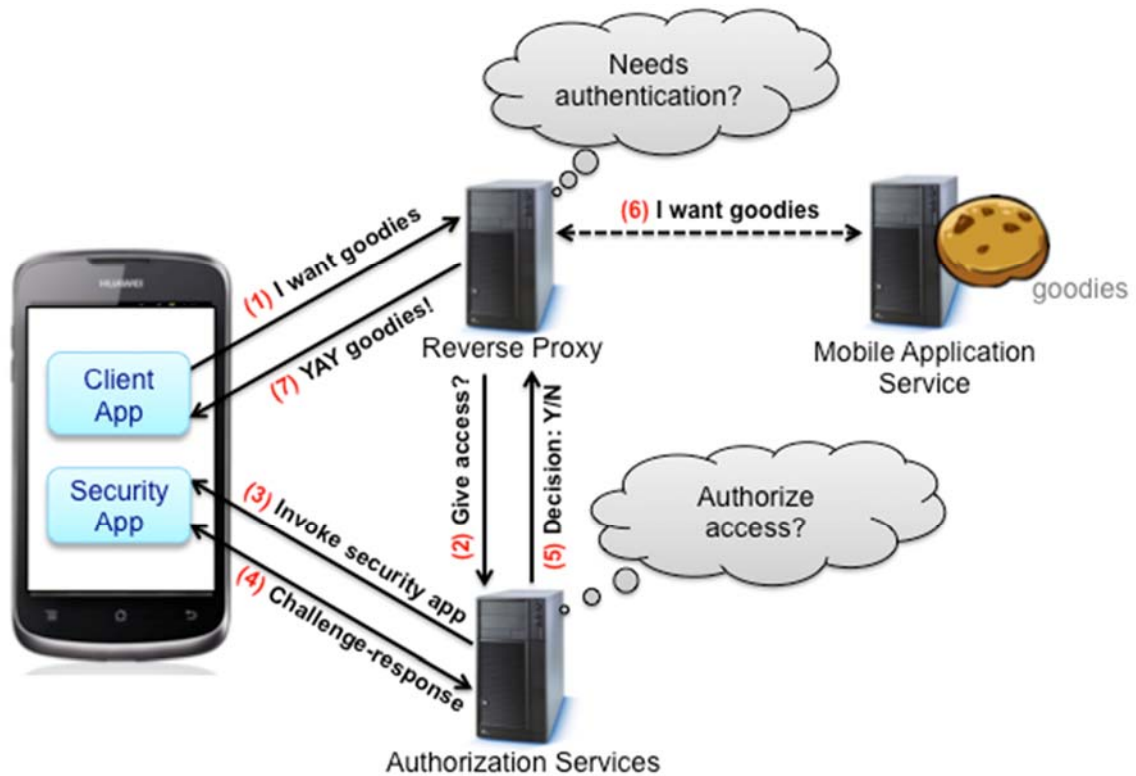
Out-of-band communication. In modern mobile platforms, application development has become much more distributed with a growing number of applications being developed by third parties. These third-party application developers often do not have a formal relationship with the network services. The application typically consumes the interfaces exposed by these network services. For example, FriendCaster is a popular

third-party mobile application that consumes Facebook data by communicating with Facebook servers using the user's credentials. This scenario presents several concerns. First, the third-party application, FriendCaster, must be trusted with the user's Facebook credentials. This can be quite risky given the untrusted landscape of mobile applications. Second, any modifications or enhancements to the Facebook's authentication mechanism, such as inclusion of biometric credentials, require modifications to all applications that use the Facebook APIs.

Figure 7.2 depicts a representative high-level architecture of our framework when configured with a reverse proxy service that enforces authentication and authorization. When an application makes a network services request, the reverse proxy intercepts the request and invokes the authentication and authorization service, which, in turn, sends an authentication challenge to the user (*Step 2*). Instead of sending the challenge over the primary communication channel, we create a secondary out-of-band channel of communication between the authorization service and a specialized security application on the client. The security application stays dormant on the client device and is only triggered by a security token sent over the push notification channel to the device (*Step 3*). The security application on the mobile client uses this token as a unique identifier when performing the appropriate authentication services as requested by the service. All authentication services are subsequently performed over this secondary channel (*Step 4*).

The primary communication channel is being blocked by the reverse proxy while the authentication is performed over this secondary channel. When the authentication protocol completes on the secondary channel, the primary communication channel resumes the communication. If authentication fails and authorization is denied, then the primary communication reports the authorization failure. If authorization is successful, the request initiated by the mobile application is forwarded to the network service for processing as usual. Aside from the reporting of an authentication failure (access denied), the client application is unaware of the authentication being performed on the secondary channel.

It should be noted that while Figure 7.2 shows the communication in steps (3) and (4) as bypassing the Reverse Proxy, in practice the actual communication may flow through the Reverse Proxy, but still remain on a secondary communication channel since the primary communication channel remains blocked by the Reverse Proxy.



The flows are as follows: (1) The Client App makes a service request to the Application Server, (2) the Reverse Proxy intercepts the request, determines if additional authorization is needed and calls the Authorization Services, (3) the Authorization Service invokes the Security App by sending authentication challenges to the mobile client, (4) the challenge- response protocol ensues between the Security App and the Authorization Services, (5) the Authorization Service communicates its decision to the Reverse Proxy, (6) if the request is authorized, it falls through to the Application Server for further processing, (7) the response is sent back to the Client App.

Figure 7.2. High-level architecture of out-of-band authentication

We support an out-of-band authentication mode that is driven by the network services. The network services define the authentication requirements for any request for a resource. Such requirements are imposed in the form of authentication challenges sent by the network services to the client. For example, let us assume that Facebook were to require a new form of knowledge-based authentication when the service deems that there is higher risk, such as authenticating from more than one location at the same time. If authentication were to be embedded into every application calling Facebook APIs, all such applications would need to update their code to handle this new API or face the wrath of unhappy users who are no longer able to log into their Facebook accounts from these applications.

While the use of a Reverse Proxy is a typical deployment pattern for supporting common authentication / authorization services for multiple applications, there are many mobile application services that directly perform the authentication and authorization. Instead of the Reverse Proxy initiating the challenge-response protocol, the Mobile Application Service invokes the Authentication Services in the same way that the Reverse Proxy invoked these services (Figure 7.3).

Finally, we note that it is sometimes desirable to communicate the user identity from the Client App to the Security App. From a usability perspective, if the user enters their identity into the Client App then it becomes frustrating and annoying to reenter the same information as part of the security credentials. The communication of the user identity, as well as any other relevant application state, can be passed to the Security App via existing inter-process communication mechanisms. Aside from this usability enhancement of passing along the userid to the Security App, there is no other communication between the mobile applications and the Security App, thus minimizing the integration and deployment.

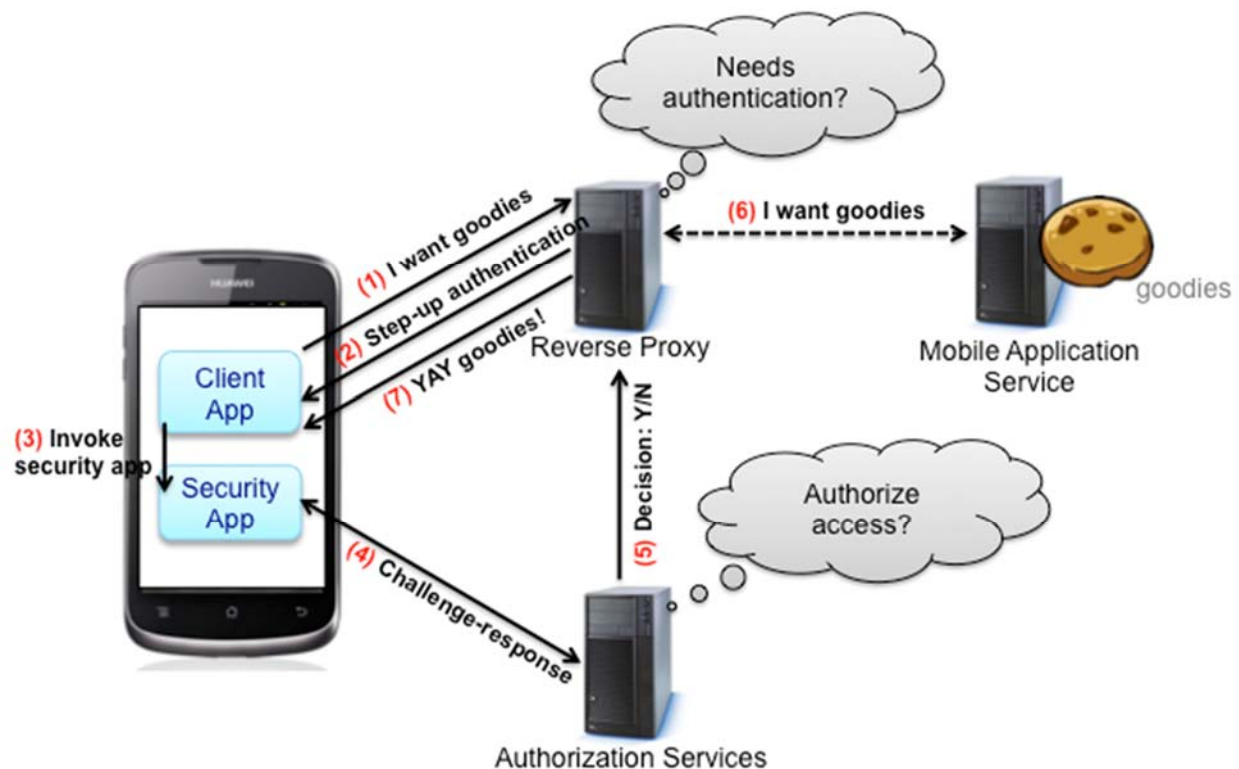


The flows are as follows: (1) The Client App makes a service request to the Mobile Application Service, (2) the Mobile Application Service determines if additional authorization is needed and calls the Authorization Services, (3) the Authorization Service invokes the Security App by sending authentication challenges to the mobile client, (4) the challenge-response protocol resumes between the Security App and the Authorization Services, (5) the Authorization Service communicates its decision to the Mobile Application Service, (6) if the request is authorized, the Mobile Application Service continues for further processing and the response is sent back to the Client App.

Figure 7.3. Proxy-less out-of-band authentication architecture with inline communication

While out-of-band authentication provides better security isolation and required minimal app modifications, its operation relies on the best-effort push notification mechanism of the corresponding platform. While it mostly works well in practice, it has intermittent performance delays that impact overall user experience. Moreover, our current implementation uses IP address (received as part of the initial resource request) as the common token to connect the initial transaction request to the mobile device being targeted for push notification. However, IP address is not a reliable token for devices that are behind a Network Address Translation (NAT). To address these challenges, we also support inline integration mode for the business application to seek additional authentication in its transactional flow.

Figure 7.4 shows the high-level architecture of the inline authentication mode. As can be seen in the figure, the step-up authentication challenge is sent over the main communication channel (*Step 2*) instead of the out-of-band secondary channel. The Client App handles this challenge by invoking the Security App using platform-specific inter-app invocations (*Step 3*). Since the authentication challenge-response protocol is initiated by the Security App at the client (*Step 4*), the role of IP Address as the connecting token between the two communication sessions is no longer required.



The flows are as follows: (1) The Client App makes a service request to the Application Server, (2) the Reverse Proxy intercepts the request, determines if additional authorization is needed and sends step-up authentication challenge to the Client App, (3) the Client App invokes the Security App, (4) the challenge-response protocol ensues between the Security App and the Authorization Services, (5) the Authorization Service communicates its decision to the Reverse Proxy, (6) if the request is authorized, it falls through to the Application Server for further processing, (7) the response is sent back to the Client App.

Figure 7.4. Architecture of inline authentication

7.3 Procedures

We designed our framework to be modular and extensible. A modular design allows easier customization of the system to a variety of applications and use cases. For example, we separated the user interface component of the framework from its core functionality logic. This separation enables a business entity to customize the look-and-feel of the app as per its own business and user requirements, while leveraging the core multi-factor authentication functionality offered by our framework.

We also envision that as mobile devices continue to evolve, there would be new sensors that would be made available in these devices. We are already noticing such trends with support of barometer sensor being made available on some new smartphones. Our framework is flexible in supporting new biometric and non-biometric modalities. To add a new modality in the client-side framework, the logic for tapping into the sensor and interact natively with the underlying platform is implemented as a hybrid *plugin*. A plugin is a package of injected code that allows the app's WebView to communicate with the native platform on which it runs. Plugins provide access to device and platform functionality that is ordinarily unavailable to web-based apps. Plugins comprise a single JavaScript interface along with corresponding native code libraries for each supported platform. In essence, it hides the various platform-specific native code implementations behind a common JavaScript interface.

Figure 7.5 shows the high-level architecture of the client-side design. The core component of the architecture is the client framework that is built over PhoneGap. It contains the core functionalities of our solution that are modularized into multiple library components:

1. *Biometrics Library*: It provides the APIs for capturing the biometric credentials – face, voice and fingerprint – and in turn interacts with the platform-specific native plugins for collecting these credentials from the device.
2. *Security Library*: It encapsulates the client-server communication primitives to construct a secure communication protocol between the client and the server. It includes transfer of biometric data from client to the server over multi-part MIME communication, and handling and responding to the authentication challenges received from the server.
3. *Context Library*: It provides the APIs for on-device context collection. It collects raw sensor data to build appropriate context to address multiple security use cases (Section 7.2.2). For example, it collects accelerometer data to detect whether a user has lost control of a device at any point. A business application

using our framework can take appropriate action when such trigger is received from this library.

4. *Out-of-band Authentication (OOBAC) Library*: Our system supports both inline and out-of-band integration to the authentication framework (Section 7.2.3). The OOBAC library handles all the communication over the out-of-band notification channel. For example, it receives the trigger for step-up authentication from the server via push notifications from the server and accordingly invokes the authentication application while passing the required state parameters.

An authentication application, with its own user interface (UI) customized to the business/government unit, can be built to consume the APIs exposed by the client framework. For our prototype, we have built a representative version of such a UI. The client framework also acts as an end point for the secure communication with Authentication and Authorization service hosted on the server and the UI and business logic of the application is masked from the client-server communication details.

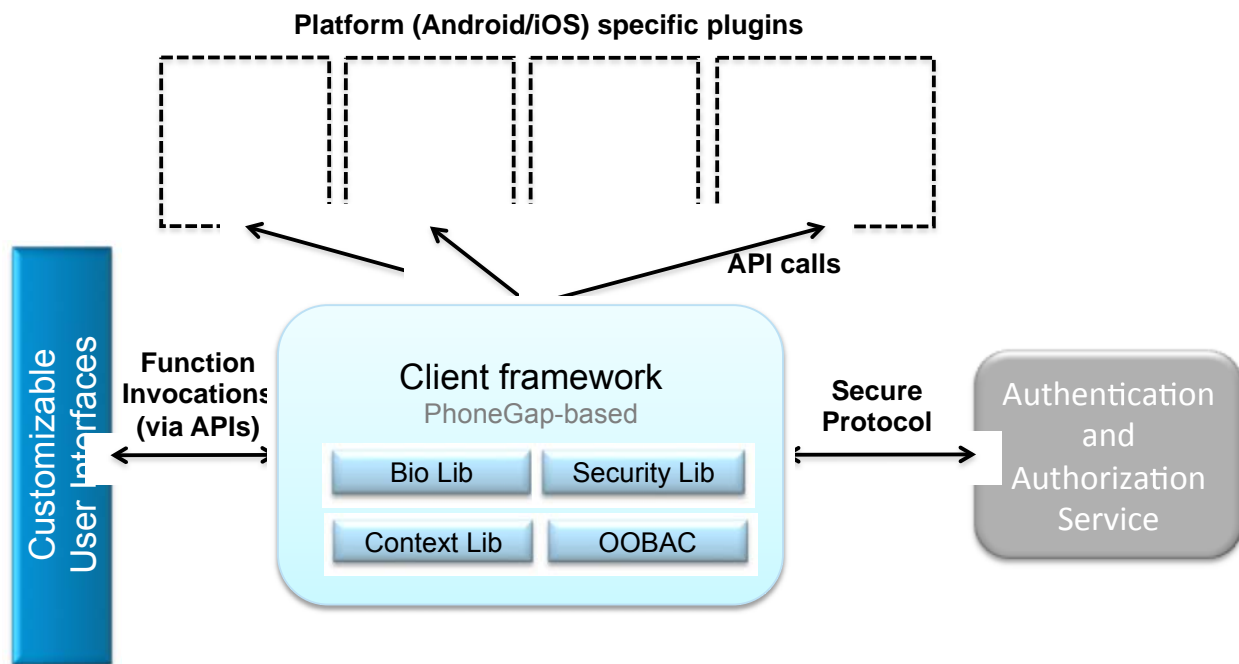


Figure 7.5. High-level Client-side Architecture

We discuss the implementation details of each of these major client-side components in the next section.

7.4 Results and Discussion

In this section, we present further details of various components of our client-side architecture and demonstrate how our solution can be transitioned to practice by showing a representative integration with a popular IBM product.

7.4.1 Core Client-side Components

Our prototype is built to support both Android and iOS platforms. Though our solution is designed as a hybrid application to support platform portability (Section 7.2.1), our implementation still required several plugins to be written specifically for the platform (i.e., Java for Android and Objective-C for iOS). For example, biometric capture code is specific to the platform as it leverages the platform APIs to interact with the device sensors, and therefore, it needs to be written as a platform-specific PhoneGap plugin.

7.4.1.1 Native Plugins and Wrappers

Our implementation provides wrappers to the underlying native plugins. The wrapper provides a platform-independent generic interface to the platform-specific plugin. Our wrappers are written in JavaScript and leverage PhoneGap's JavaScript-to-Java (for Android) and JavaScript-to-Objective-C (for iOS) bridge to call the plugins. In turn, our hybrid client-side framework utilizes these wrappers. The primary wrappers are as follows:

- **AppControl** – Provides APIs to gracefully close the authentication app (e.g. clear cookies, etc.) and switch control back to the business app with appropriate authentication response.
- **Bluetooth** – Provides APIs to retrieve information on neighboring Bluetooth devices both inline (on demand) and periodically.
- **CustomCamera** – Provides APIs to take a picture using the device's camera and exposes a custom native interface for image captures.
- **DetailedDeviceInfo** – Provides APIs to get meta-information about the device, such as device ID, device model and platform.
- **FingerprintScanner** – Provides APIs to get fingerprint reading from our supported FbF Mobile One fingerprint scanner and invoke Touch ID support in iOS.

- **MultifileTransfer** – Provides APIs to upload multiple credential files to the server as response to the server challenge.
- **WavAudio** – Provides APIs to collect audio samples using the device's microphone in WAV format.

Each of these wrappers calls the corresponding native plugins as described below.

AppControlPlugin. It exposes functionality to clear app state and gracefully transition to the calling business app. For the iOS version, it uses a custom URL scheme to invoke the business app. For Android, it returns to the business app via the callback to the `startActivityResult` function call invocation from the business app (for inline mode only).

BluetoothPlugin and BluetoothService. These two plugins are used to collect neighboring Bluetooth device information for context-based security. While BluetoothPlugin exposes functionality to invoke the data collection on demand and responds back, BluetoothService periodically (with configurable time interval) collects the data and writes it to a file. This data is subsequently provided to the hybrid framework layer on demand. Note that while Android native APIs allow collection of meta-information that includes neighboring phones, iOS limits its collection to only certain Bluetooth peripheral devices (without programmatically exposing neighboring Bluetooth-enabled phones).

CustomCameraPlugin. This plugin enables capture of one (for verification) or multiple (for enrollment) face images for biometric authentication. We opted to build our own custom face capture functionality instead of relying on the default camera application of the device for multiple reasons. First, custom camera enables us to have a user interface for the face capture screen to be consistent with other parts of the application. Second, it gives control and flexibility to capture image at a specific resolution (as required by the verification engine). Finally, it does not rely on an external (potentially untrusted) application and instead includes the custom camera into one cohesive authentication app. Note that our prototype also supports custom user interfaces for both phone and tablet form factors and such interfaces for the custom camera are statically defined in our implementation.

DetailedDeviceInfoPlugin. It calls the underlying platform APIs to collect meta-information (device id, platform, etc.) on the device that are exposed to the hybrid framework layer via the wrapper.

TouchIDPlugin and FbFPPlugin. These two plugins collect biometric data for the fingerprint credential. We provide fingerprint support using two technologies: the de-

facto Touch ID (for newer iOS devices) and our representative FbF Mobile One fingerprint scanner. iOS's native APIs for accessing Touch ID do not expose the raw fingerprint data and instead provides the result of the on-device fingerprint authentication (i.e., whether the fingerprint input matches the registered fingerprints on the device). Therefore, in case of Touch ID, the client communicated the authentication result itself to the server. On the other hand, FbF Mobile One fingerprint scanner enables direct access to the raw fingerprint capture that is uploaded to the server for server-side verification.

MultifileTransferPlugin. Our client-server protocol supports multiple credentials to be transferred in one response to a server challenge. We use multipart MIME encoding for the transfer to enable diverse biometric and non-biometric credentials to be transferred at tandem. This plugin marshals the multipart MIME payload with the relevant credential data, uploads to the server and handles the server response.

CustomAudioPlugin. This plugin enables capture of one (for verification) or multiple (for enrollment) voice samples for biometric authentication. We opted to build our own custom voice capture functionality instead of relying on the default microphone application of the device for reasons similar to the ones previously discussed for CustomCameraPlugin. For our prototype, voice is sampled at 8000 Hz to support our verification engine. However, this value is easily configurable in our implementation.

7.4.1.2 Hybrid layer

The PhoneGap-based hybrid layer of our authentication solution consists of the UI component and the client-side framework. In addition to providing all functionality required for authentication, including credential capture, context collection and communication with the server, the client-side framework also translates any errors and server responses into a more consumable format for the UI.

UI Component

The UI of the authentication app is implemented in HTML with the dynamic behavior defined in JavaScript. There are several useful features in the UI implementation:

- The UI screen is adaptive to the form factor of the device (i.e., phone or tablet).
- The main authentication screen customizes itself to the number of authentication factors being supported, i.e., the buttons on the screen realign to provide appropriate screen coverage.
- Our solution also enables server authentication for protection against phishing attacks by using a user-chosen image as the authentication factor. The UI

automatically downloads the appropriate image from the server and renders it on the screen.

- Risk associated with the current situation is communicated to the user by means of a semi-gauge that reflects low, medium and high risk.
- Other meta-information, such as last login time, is communicated to the user by the UI.

Application Settings. A device user can configure several configuration parameters at runtime using the application's settings screen:

- *User Configuration:* A user can specify his *UserID* using this configuration. The use of a single UserID in the authentication app allows the application to be used as a single sign-on for multiple applications on the device.
- *Server Configuration:* It contains the server parameters to define the sever connection. This includes the *Protocol*, *Server Host*, *Port* and *Root Directory* for the REST APIs.
- *User Privacy Preferences:* As discussed in Section 7.2.2, we passively collect contextual information from the device for passive authentication and providing additional security. We do allow the user to opt out of such context collection by specifying his privacy preference settings. This includes enabling/disabling collection of Geolocation and Bluetooth data and use of anti-phishing image for server verification.

Client-side Framework

The client-side framework is the central “controller” that provides the binding between the UI component, the native wrappers and the communication with the server.

The client handles a typical step-up authentication challenge from the server in the following steps:

1. On a resource request from the client, the server sends a step-up authentication challenge to the authentication app in the form of a JSON response. The JSON contains information on the biometric credentials required for the authentication. Additionally, it contains any contextual information (such as geolocation) required by the server to make the risk-based authorization decision.

2. The framework receives the response and unmarshals it to a form consumable by the UI. Additionally, it passively retrieves the contextual information by calling the relevant native wrappers.
3. The UI renders the relevant authentication buttons based on the challenge received. For example, if it only receives “face” as a challenge, it enables the face capture button while disabling all others.
4. When the user presses a button for biometric capture, the UI calls into the framework that captures the credentials by calling into the corresponding native wrapper.
5. After a callback from the native wrapper, it uploads the captured credential to the server using the multi-file upload along with the corresponding File Type Map (FTM) information (see below). Subsequently, it deletes the captured credential from the client.
6. Once an authentication success is received from the server, the authentication app gracefully shuts down while passing control back to the business app. The business app is given access to the requested resource.
7. If authentication failure is received from the server, the authentication app gracefully shuts down while passing control back to the business app with authentication failure as response code. The business app shows an appropriate failure message to the user.

File Type Map (FTM) contains meta-information about the file being uploaded from the client to the server. For example, it specifies if the file being uploaded is a “face” file, or a “voice” file. Since multiple recognizers could use the same file type, the server cannot just rely on just the file type. Instead, the server handlers leverage the FTM to select the files to be processed. FTM provides a one-to-one mapping between its entries and the files being uploaded in the order in which the files are being uploaded. For example, to upload a voice credential and two face credentials, the FTM would be in the JSON form {“voice”, “face”, “face”}.

Context-based Security. Contextual data on the device is collected either on demand (from the server) or periodically. All data, except the accelerometer for user presence detection, is sent to the server for risk-based authentication and authorization decisions. While the geolocation data is collected on demand from the server, we collect Bluetooth data periodically and send the latest snapshot of the data when Bluetooth information is requested from the server. We opted for this implementation to reduce the client

response time, as Bluetooth device scanning is a time-consuming task and is therefore performed on a parallel thread in the background.

As discussed previously, user presence detection is a feature that is integrated with the business app itself. It runs as a background service that collects accelerometer readings from the device continuously and leverages them to detect if the user is present with the device. Once idleness is detected, it triggers a callback to the business app that can take appropriate action, such as user logout, to protect its sensitive data.

Authentication Libraries (biometrics.js and serverUtils.js). The libraries are exposed as JavaScript objects that provides all the core functionalities for authentication that includes biometric capture (using captureAudio, captureImage, captureFingerprint and verifyFingerprint interfaces), handling client-server protocol (including the challenges and error responses) (using interfaces, such as setBiometricsForVerification) and out-of-band authentication trigger.

Out-of-band authentication leverages the push notification mechanism of the underlying operating system. In order to handle and consume the trigger, we provide our custom implementation for didReceiveRemoteNotification method for iOS. For Android, we implement the client and server side components of the Google Cloud Messaging (GCM) Service. On the client side, we received the notification trigger by implementing our Intent Service that extends the GCMBaseIntentService.

7.4.2 Representative Business Application: Banking

To demonstrate and evaluate the multi-factor authentication in a representative setting, we developed a hybrid mobile application for the banking use case. The application contains typical banking functionalities, such as account balance (checking and savings), deposit, scheduling account-to-account transfer and adding a new payee.

Each of these banking functionalities represents a different level of risk and accordingly requires different level of confidence in the user's identity. For example, adding a new payee is more risky operation than checking the bank balance, and correspondingly requires step-up authentication. Relevant parameters of an operation also determine the authentication required for the operation. For example, transfer of a higher amount represents high risk for the user/bank in comparison to a lower amount, and hence requires higher confidence in user's identity.

We also utilize the banking application to demonstrate flexible integration with our multi-factor authentication solution. As discussed earlier, our solution is prototyped as a hybrid application (called Authentication application). Our prototype shows both the out-of-band authentication mode (where no (Android) to minimum (iOS) change is required

to the banking application) and the inline mode (where application received the step-up authentication challenge and subsequently invokes the Authentication application). For out-of-band mode in iOS, the banking application needs to expose a custom URL scheme that can be called by the Authentication application during callback. This change is required because unlike Android, iOS does not automatically pass the control back to the last running application (i.e., banking application) when Authentication application finishes its task.

To leverage user presence detection to provide additional security, the application can integrate it into its own code by using the corresponding APIs provided by our solution. User presence detection is started as a service with three parameters: (1) idleness noise discount, (2) idleness timeout period, and (3) the callback function to be called when idleness is detected. The application can choose its own action when idleness detection is triggered. For the banking application, we do not logout the user completely, but instead hide all the confidential data on the screen (e.g. account balance) and reduce the authentication confidence. As a result, the user (or attacker) has to provide additional credentials to meet the authentication confidence requirements for access.

7.4.3 Transition to Practice: Integration with ISAM

We integrated our framework into a popular reverse proxy product – IBM Security Access Manager (ISAM) for Web and Mobile – to demonstrate that our design will work with a real-world authentication solution. ISAM is based on a scalable security reverse proxy server that has been in use for 20+ years by many large enterprises in a broad range of industries, including telecom, finance and technology. It has a comprehensive suite of security features, such as support for existing hardware and software tokens, including one time passwords, as well as support for the leading industry standards, including OAuth, OpenID and SAML.

We opted in favor of ISAM as our representative use case for multiple reasons. First, ISAM provides an appropriate network enforcement point that is key for our proxy-based, server-enforced authentication solution. Second, it is enterprise-ready and is scalable. Finally, it already enjoys a large customer base that is seeking new authentication technologies to support mobile, web and Internet of Things (IoT).

Figure 7.6 shows a high-level view of our integration with ISAM. The server-side component of our solution is integrated into the Authentication/Authorization service of ISAM. The resource request is proxied by the ISAM proxy for Web, which queries ISAM's Authentication/Authorization service for authentication decision making. This triggers our server-side code that, in turns, uses ISAM's notification service to send out-of-band authentication trigger to the client. The communication between the

Authentication app on the client and the server-side component of our solution remains unchanged.

For our integration, we needed to satisfy some core requirements of ISAM into our client framework:

- We integrated OAuth support into the system workflow of our Security app and used the OAuth token as the secure session identifier.
- We improved our original error handling to make it robust to consider app runtime states and error conditions.
- We extended our Logout protocol to additionally clear states in ISAM.

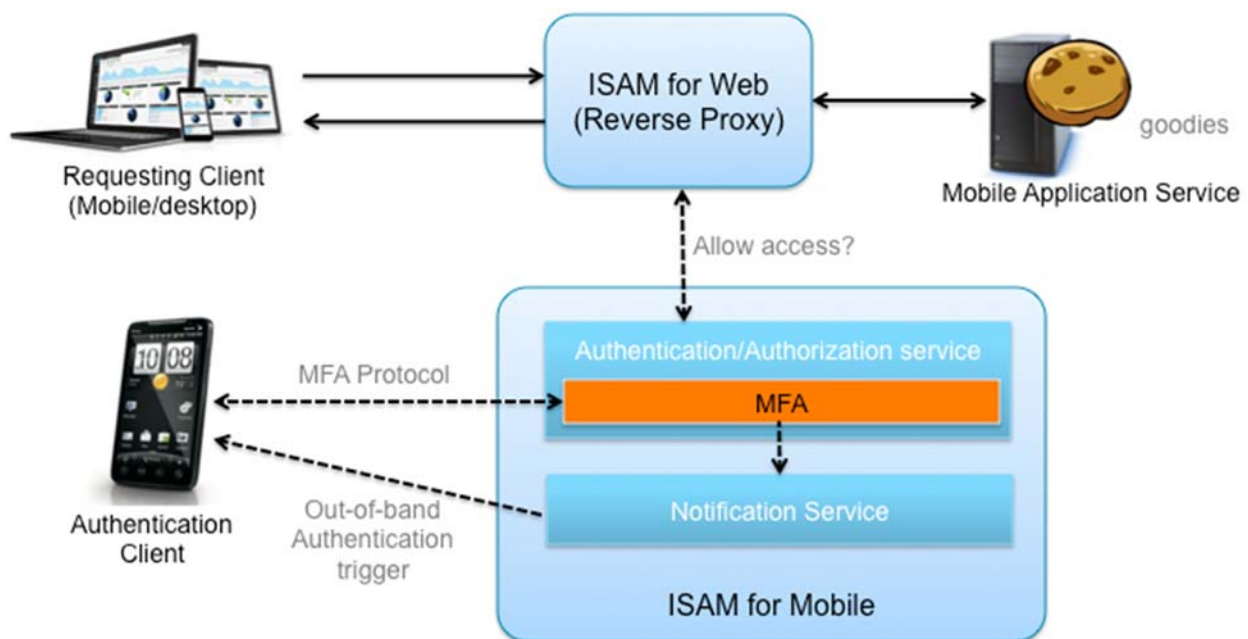


Figure 7.6. Representative Integration with ISAM

7.4.3.1 Multi-factor authentication for web apps

Integration with ISAM also enabled multi-factor authentication support for ISAM existing use cases. ISAM is also used extensively for moderating access to web services from the browser for both desktops and mobile devices. Our solution readily enables multi-factor authentication support for web applications.

A user initially registers one or more mobile devices (with sensors for collecting biometric credentials) with ISAM. When an access request to a web service is received for that particular user, ISAM sends an out-of-band authentication trigger to one of the user's registered devices. After the challenge-response protocol is completed between the Security App on the device and our server-side component, the decision is communicated to the reverse proxy. The proxy, in turn, enables or disables access to the web resource.

7.5 Conclusions

The client-side component of our multi-factor solution is designed to be modular, extensible and portable to multiple platforms. We used a hybrid application design that supports multiple (inline and out-of-band) integration options and hence is easily portable to a variety of mobile platforms and applications. We evaluated our design by integrating it with a representative banking application that was built in-house, and showed that minimal changes are required in the business application to support our multi-factor authentication solution.

In addition to the active biometric and non-biometric authentication, our solution also collects and avails mobile contextual factors, such as geolocation and Bluetooth, for passive risk-based authorization and other factors, such as accelerometer, for user presence detection for additional security.

We further highlight the value of our solution by integrating it with the popular ISAM reverse proxy that shows its potential path to large-scale productization. We believe that such packaged offering in addition to the salient features, such as portability and extensibility, would appeal to a large set of existing and future business applications.

7.6 Recommendations

In this section, we provide recommendations on how a business app can apply multi-factor authentication with minimum integration effort.

Which integration option to consider? If the business app and the authentication app are developed by mutually trusting parties, it is recommended to use the inline mode to achieve better performance (as the push notification mechanisms for both Android and iOS can be unreliable). Inline mode should also be considered if the mobile device is behind Network Address Translation (NAT); Out-of-band mode is limited by the NAT as IP address is used to connect the session on the main channel with the authentication challenge-response on the secondary channel.

Out-of-band mode should be used if the authenticating device is different from the host requesting access to a resource. This is especially beneficial for the requesting host,

such as a desktop, that does not have appropriate sensors to collect biometric or contextual credentials. Out-of-band mode provides a “move-in ready” solution for business apps (web or mobile) that seek to use multi-factor authentication with no to minimum modifications.

Considerations for context-based passive authentication. Performance and privacy are two major considerations for collecting and consuming mobile context for authentication. Continuous collection of context is detrimental to the device’s battery performance with certain factors, such as Bluetooth scanning, being more expensive than others, such as geolocation. On the other hand, large sampling interval could potentially result in missing relevant context for making security decisions. It is recommended for the user to consider his own usage patterns to decide on the sampling rate. For example, if battery life is more valuable to the user, he can opt for higher sampling interval or have adaptive sampling based on multiple factors (e.g. increase Bluetooth scan interval when in less risky geolocation).

It is recommended to allow the user to opt-out of context collection at any time, so that he can decide on his own privacy under different circumstances. Note that the service providing the resource can optionally disable access if no context is available, thus encouraging users to opt-in.

How to include new credential type? A new credential could be easily supported in our design by utilizing the extensibility of our client-side framework. A platform-specific plugin is first developed to capture the credential by calling the platform APIs for the corresponding device sensor. A JavaScript wrapper is then written to expose the native interface directly to the hybrid framework that is subsequently used by the framework to capture the credential. The hybrid framework, in turn, handles the success and failure responses from the native layer and provides consumable codes to the corresponding UI. The UI can include an additional button for the new credential.

8.0 NETWORK AUTHENTICATION AND AUTHORIZATION SERVICES

In this section we describe our architecture and design of the *Network Authentication and Authorization Services* (NAAS). An important practical aspect of the project is the demonstrable feasibility of integrating the multiple elements of this project – risk communication, multi-factor biometric authentication, risk based authorization and client interaction – with an existing network based authorization service. We describe the assumptions around creating such a service and how we implemented a proof of concept.

We used an iterative design process whereby we were able to construct an operational version of the system, incrementally adding functionality and services over time. We report here the current status of the NAAS prototype.

8.1 Methods

The overall goal is to provide network based authentication and authorization services that are both easier to use than traditional userid / password, yet provide stronger authentication based on multi-factor biometrics. To make the system more usable, we want to limit authentication to those times when it is necessary and only challenge for security credentials (e.g., biometrics) to sufficiently mitigate the risk associated with the transaction. We hypothesized that communication of the risk at hand would aid in gaining end-user acceptance for stepped up authentication. We do so when the risk assessment determined that additional authentication would be required. To make end-user authentication more usable, we provide for multiple types of authentication; within this project we focused on biometric authentication, although non-biometric authentication techniques can be easily integrated.

We have identified the following system elements that are required to address the requirements of a system to meet these objectives:

- Integration with an application server, or network based authentication and authorization service. We defined a shim layer that allows the core set of NAAS services to interface with any number of systems assuming that the authentication protocol can be supported. Since the system we created is general and extensible, we have demonstrated its integration with a network-based reverse proxy server (IBM's Security Access Manager [ISAM]) that provides authentication and authorization services. We have also demonstrated that our services can be performed in the absence of such a reverse proxy server.

- Secure communication between a mobile client and a network based authentication and authorization service. If this were to be a solution that is only based on local area networking, there would be less concern about networking overhead. However, since 3G and 4G networks can be far slower and with greater latency than gigabit local area networks, we need to be cognizant of the number of messages and quantity of data being transmitted.
- *Inline* and *out of band* processing modes to either make it easier for programmers to integrate our technologies, or provide a more robust and tightly integrated user experience. *Inline* processing means that the authentication logic is integrated as part of a callable library with the business logic of the application (app) on the device. *Out of band* processing separates the authentication into an app separate from the business logic / app.
- Processing of biometric samples, both for “enrollment” of the end-user into the system, as well as for verification of the user. In many ways the handling of biometric samples from the client and handing them off to the biometric engines for enrollment and verification are similar. We have designed a general purpose extensible framework for this purpose. As with mobile communication, there can be a significant cost both in time and processing requirements for biometric operations. It is well known in the literature and through anecdotal evidence that end-user perception of response time is very important in the overall usability of the system. To address this, we support parallel processing of biometric verification to reduce the overall latency when multiple biometric authentication samples are concurrently received from the client to be processed.
- Risk based authorization determines how much authentication will be required based on the value at risk. We consider a few risk factors and apply a policy, form an assessment, generate appropriate authentication challenges.
- Authentication challenges are generated that are both interactive and non-interactive. Interactive challenges are those where the end-user needs to interact with the system, such as speaker, face or fingerprint biometric verification. Non-interactive authentication is where no end-user interaction is required, such as geolocation anomaly detection (is the user where we expect them to be based on GPS coordinates reported by the mobile device).
- Risk communication is based on the risk assessment, taking into account a variety of risk factors. These may include unusual time / location for the resource request, high value asset request, etc., and will result in appropriate messages needing to be presented to the end user at the mobile device.

- Administrative functions to configure and manage the NAAS services. This includes the configured and enrolled users, devices and their capabilities, *PushNotification* registrations, authentication policies and configuration, and user authentication status.

8.2 Assumptions

The three types of users of the system – end-user, security administrator, and programmer – are all impacted by the design of the NAAS. We can break down the assumptions as they are related to each of these three classes of users:

End users: These users are motivated to achieve their primary task, which is typically not security or authentication related. In general, we are unaware of the semantics of their primary task. However, it is important to minimize the time required for end users to perform the authentication tasks. If there are multiple apps on the same device that use the same authentication domain, then single sign-on is important to minimize the total number of authentication challenges requiring user attention.

Security administrators: Administrators need to configure the system, identify and track the end-users of the system, the mobile authentication devices and their capabilities, and configure authentication policies. Related to this is configuration of risk communication messages. One of the key challenges for security administrators is that mobile applications (apps) typically operate in a complex environment involving a range of network topologies and service providers, new and legacy network services, multiple apps, and a range of mobile platforms and versions. The mobile devices have a non-uniform set of capabilities that can be used for authentication. Some devices may have fingerprint readers, while other devices may not have front facing cameras.

Programmers: There are several touch points depending on which part of the system they are performing integration services. There is the interface to the mobile application services in the network or the reverse proxy security server, integration of biometric engines (enrollment and verification), and non-biometric context / risk factor assessment services.

Representative Network Control Points

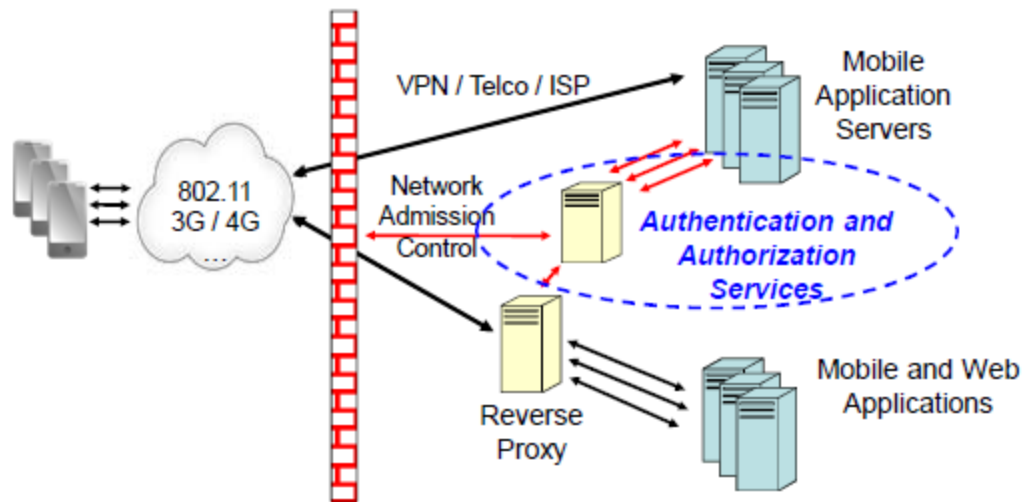


Figure 8.1 Representative Network Control Points

8.2.1 Integration with network authentication / authorization services and network services providers

As shown in Figure 8.1, there are multiple security control points in the network where the NAAS can be inserted: at the VPN gateway into an enterprise, attached to the mobile application servers / services, or at a reverse proxy authentication / authorization service. We assume that these are all legitimate network security control points. The NAAS architecture and design need to support all of these. Some of these use cases are HTTP / HTTPS based. Other protocols may be used, such as RADIUS [RADIUS]. To do so, we will separate the set of core NAAS services from the interface to these network control points. This is done through a shim layer that we describe below.

In our prototype, all interaction between the mobile device and our authentication services may be routed either through the reverse proxy server / application server, or communicated directly between the mobile device and the NAAS. For the discussion here, we will assume that all communication is securely routed through the reverse proxy server / application server over SSL. This simplifies security administration via a single network control point and SSL.

Whether our services are integrated with a reverse proxy or application server, there is no observable difference to app programmers or the end user. However, the security administrator typically will have tasks that are specific to the kind of network control point. There will also be differences when identity administration is self-service (e.g.,

consumer facing services) vs. managed identity (e.g., enter[prise identity management). In this paper, we will primarily focus on the reverse proxy and enterprise scenarios.

8.2.1.1 Reverse Proxy

In the reverse proxy use case we assume that the reverse proxy server will keep track of the end-user identity (when known) and the authentication session identifier. We also assume that the reverse proxy will manage or connect to an appropriate identity registry, such as LDAP or Microsoft Active Directory. User identity management is outside the scope of the services we provide.

At the conclusion of the authentication and authorization process, the reverse proxy needs to know (1) the authenticated identity of the user and/or (2) whether the authentication and authorization was successful. If authentication and authorization is successful, we return the user's identity to be communicated to the network-based services that are being protected by the reverse proxy. The identity returned must be a known / legitimate identity that is recognized by the reverse proxy. If authentication fails, an error condition is raised and the reverse proxy will return an error condition to the calling app – e.g., HTTP status code *401 Unauthorized*.

A key advantage for using a reverse proxy server is that it can support single sign-on for multiple apps that use the same reverse proxy.

8.2.1.2 Mobile Application Service

In the mobile application service use case, all of the communication between the mobile app and our service is routed through the mobile application server, typically over HTTPS. A user registry is required and may be managed by the application server or a 3rd party service, such as LDAP or Microsoft Active Directory. As such, an interface is required to confirm that the userid being asserted by the client is in fact legitimate.

At the conclusion of our authentication and authorization process, the application server wants to know (1) the authenticated identity of the user and/or (2) whether the authentication and authorization was successful. If authentication and authorization is successful, we either return the user's identity to the application server so that the server can keep track of the identity for the current session, or make a call to the server's runtime where we assert the session's identity. If authentication and authorization fails, an error condition is raised and an error condition is sent to the calling app – e.g., HTTP status code *401 Unauthorized*.

If an application server is hosting, or proxying, multiple mobile application services, then it has the potential for providing single sign-on services for the multiple apps and/or mobile application services.

8.2.2 Secure operation and efficient communication protocols

We now turn our attention to the security of our design and the efficiency of the communication protocol between the client and the server.

8.2.2.1 Secure operation

Security of the system needs to be considered at all of the layers of the system. We consider this here rather than break it up throughout the document.

Specifically, we will look at the following elements of the system at the security considerations: Network and protocol, mobile device, NAAS and biometrics. As previously noted in the Risk-Based Authorization discussion (see Section 6.0), it is possible to use a traditional role-based access control policy as a backstop in case RBA miscalculates the risk.

8.2.2.1.1 Network and protocol security

Communication between the mobile device and our security services needs to be over a secure communication channel, such as SSL or TLS, to ensure at least integrity and confidentiality. For the purposes of the system we constructed, we will assume the use of HTTPS, and rely on the reverse proxy to establish the appropriate cipher suite that is appropriate for the organization.

One of the problems that arise with mobile devices, whether it is with web browsing or mobile apps, is that there is no *chrome*, including the web site and traditional desktop security indicators. By *chrome* we mean the borders, title bar, status bar and other visual icons that include security status indicators. The device user is left with no obvious way to check whether the browser or app is securely connected to the web site of interest. Even with desktop browsers, the end-user can be duped into a man-in-the-middle situation, for instance, by means of a phishing attack. Recently it was discovered that the Gogo in-flight (airline) Wifi service was issuing their own certificates so Gogo could throttle or block high bandwidth services [Felt]. With mobile apps, there is no chrome or other means for the end-user to determine whether there is a man-in-the-middle attack. The only effective means for thwarting such an attack is for the app on the client device to verify the server-side certificate.

There is the question of whether an arbitrary app may be allowed to authenticate through the NAAS. There are different perspectives on this, based on the organization's security policies. Less risk adverse organizations will allow 3rd party apps to be used to gain access to the network services. More risk adverse organizations will want to know which app is attempting access. Authenticating the business app itself is not the focus of this work and our current prototype assumes the former perspective.

We do afford protection of the user authentication process by isolating the third-party business app from the authentication app. In the latter case, an app identifier can be added to the app to prevent arbitrary apps from attempting to access the services via the NAAS. One simple approach is to generate random keys that are injected into the app prior to download, and are verified / registered by the NAAS on first use on a device / by a user. Think of this as a random password or token used to authenticate the app instance. A variant of HOTP [HOTP] may be a viable alternative. The sophistication of the scheme used is relative to the level of security desired.

Related to the network security is the security of the protocol between the mobile device and NAAS. In the current prototype using the reverse proxy, network traffic is protected by HTTPS, and we use Servlet HTTP sessions to keep track of the set of interactions between the authentication app and the NAAS. HTTP cookies are set by the NAAS shim layer and the authentication app (or SDK library for the *inline* case) uses this cookie to maintain the authenticated session. The cookies are protected by SSL / TLS when communicated over the network, and access to the cookies is limited to the authentication app or business app. Implementing the authentication app with PhoneGap / Cordova allows using its WebView component (HTML/JavaScript/CSS rendering) to automatically do the HTTP session / cookie management. As long as the authentication UI does not engage in 3rd party web site content download, the HTTP session cookies should be protected as well as any app's private state.

When using the out-of-band authentication mode, there needs to be a way for the authentication app to establish a secure session with the NAAS. The PushNotification service on the mobile device triggers the launch of the authentication app on the device. Since the establishment of the authentication session may not be on the same IP address as the original application request – the device suspended and changed network connections – we use a random session identifier to create an association between the authentication app and the original resource request (e.g., business app). The details of this protocol and design can be found in our related publication [SinghKoved].

As part of the integration of the NAAS with a reverse proxy, we added OAuth support into the authentication app. When the authentication app is initialized for the first time, it prompts for the user's userid and password. The reverse proxy returns an OAuth token that is then included as a header in all subsequent HTTPS calls to the NAAS. Once the user is authenticated, the reverse proxy will set this OAuth token as a cookie in the business app's HTTP header so that it will be authorized for subsequent resource requests. The use of an OAuth token allows the re-use of an existing set of security services that are already supported by the reverse proxy server rather than inventing a new set of security protocols.

8.2.2.1.2 Mobile device security

The security of the mobile device is out of scope for this project. However, there are a few best practices that should be observed, especially around mobile app code signing and launching of the authentication app.

Access to content on mobile devices is usually dictated by the digital signature on the code being deployed. If the code for two or more apps is signed with the same digital certificate (e.g., developer key), then the apps are considered to have the same principal and generally have access to application specific content that has been written to secondary storage. If there is sensitive content, including biometric data or session keys, then it is unwise to share the signing key of the app with other apps, especially if they may be less trustworthy.

As noted below in Section 8.2.3, there are two basic approaches to incorporating the authentication logic into a mobile app. The first approach is to use a SDK to pull the logic into the mobile app itself. The second approach is to use a separate app to perform the authentication. In cases where the development / deployment is fully trusted, and the risk of app compromise is low (e.g., no 3rd party content is incorporated into the app), then using an SDK has relatively low risk due to loss of biometric data or compromised context risk factor data collection and reporting. In cases where the business app has higher risk, including cases where the app is developed by a 3rd party, then separating out the authentication into a trusted app is more appropriate and reduces the overall security risk. This is further discussed in published our previous work [SinghKoved].

We rely on the security guarantees of the underlying platform (Android and iOS) to enable secure launch of the authentication app by the business app. This app launch is based on inter-app communication mechanisms (Intents for Android and custom URL scheme for iOS) that only allow limited data to be communicated only in text format without any sharing of state between applications. Assuming that the authentication app is trusted and not vulnerable, it would not be passing any sensitive information to the untrusted third-party business app, thus ensuring the security of the launch process.

8.2.2.1.3 NAAS and biometrics security

The primary mechanism for protecting NAAS is to keep all of the APIs, including the shim layer and administrative functions, behind the reverse proxy or other protection mechanism. With an HTTP server, such as Tomcat or a reverse proxy server such as

ISAM, it is possible to configure the Servlets and other APIs to require SSH and to require authentication to perform protected operations.

One of the key areas of security concern is around the biometric data. For reasons of good engineering, scalability and security, we separated out the biometric processing into a separate web-based service. Communication with this service could also be protected via authentication and SSL / TLS. By keeping the biometric data processing separate, we can establish multiple physical servers to enable scale-out processing to meet peak processing demand. This also affords us the ability to sandbox around the service so that any compromise of the service can be limited in its ability to leak data through various means, including setting up firewall rules. Similarly, since the NAAS does not have access to the biometric templates used for biometric verification, compromise of the NAAS will limit the ability of the attacker to collect biometric sample data.

To provide additional security to the biometric data on the client, our framework deletes the credentials after they are successfully uploaded to the server for authentication.

8.2.2.2 Efficient communication protocols

The communication bandwidth between the mobile device and the NAAS varies depending on whether the mobile device is connected to a local area network or a wireless service provider over a 3G / 4G network. When attached to mobile / cellular networks, the bandwidth is expected to be much lower with a latency that is higher than on a Wifi network. As such, we need to be aware of the performance (and cost) limitations and design a message flow that tries to minimize the number of messages between the mobile device and the NAAS. Specifically, we need to minimize the number of messages between the *Security library* on the mobile device and the *Reverse proxy* in the (enterprise) network.

We will look at three scenarios to understand the basic message flows. Then we will discuss possible optimizations.

In the next three figures we have the following entities:

Entity	Purpose
Customer Client App	This is the business app / logic that is trying to access the network resources.
Security	This encapsulates all of the client runtime services for handling

library	authentication and context challenges. It handles the communication protocol between the client and the NAAS (as will be described below).
Client bio library	Biometric data acquisition library on the mobile device.
3rd party client code	Any 3 rd party code that may be incorporated as part of the Customer client app.
Firewall	Enterprise firewall.
Reverse proxy	Mediates resource requests from the client, forcing authentication and authorization based on security policies. Passes through calls to NAAS services as required. NAAS will make authentication and authorization recommendations that will be enforced by the Reverse proxy.
NAAS	This includes all of the NAAS API services as will be described below. Makes authentication and authorization recommendations that are to be enforced by the Reverse proxy.
Bio Fusion Engine	Combines biometric authentication scores to create a combined / fused score that is used by the authorization components of the NAAS.
Server Bio engines	Takes biometric samples and performs biometric enrollment and verification of user identity.
Server app	The target system containing resources requested by the Customer Client app.

The simplest scenario is a mobile app request to an unprotected resource. Figure 8.2 depicts the basic message flows. In this figure, and the subsequent two figures, the bold horizontal lines are the flows of primary concern since these are the messages that are likely to flow over a slow network. Other messages in these flows are either internal to the client, authentication service, or enterprise network.

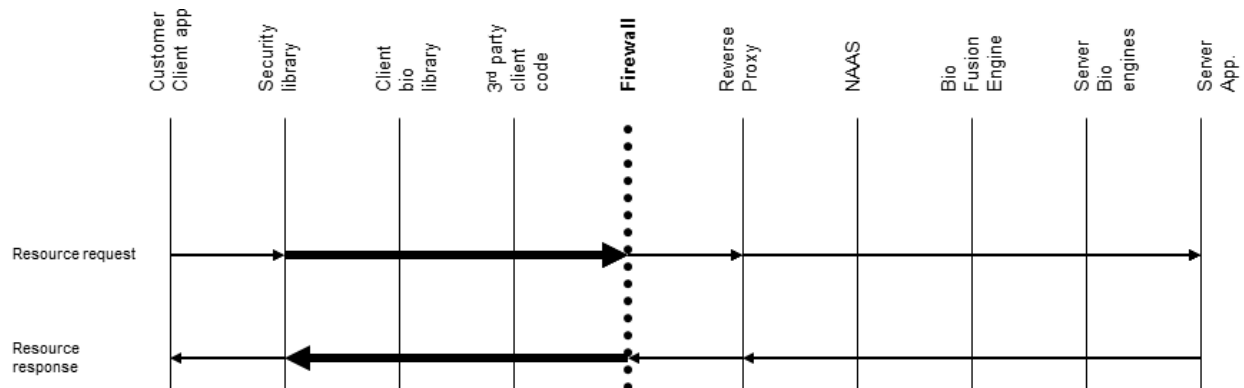


Figure 8.2 Unprotected Resource Access Message Flows

For an unprotected resource access, there is a simple request and response flow.

Protected resources have a more complex flow as depicted in Figure 8.3.

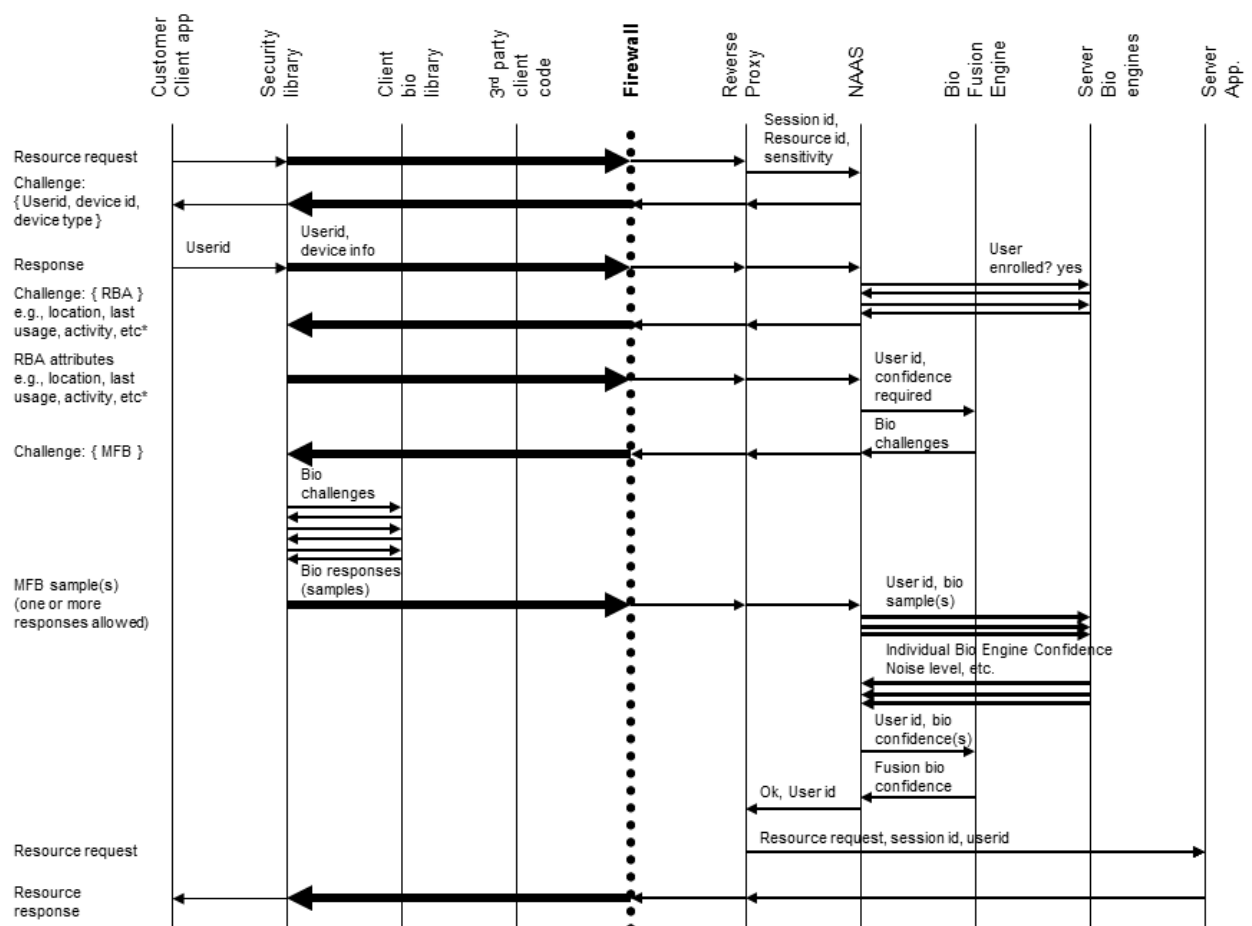


Figure 8.3 Protected Resource Access Message Flows

On the first request from the mobile device, the device will be challenged for the user's claimed identity (e.g., userid) and device information. This is the second line labeled *Challenge: {Userid, device id, device type}*. Registration of the user or device would take place at this point if not previously completed. Otherwise the security library returns this information to NAAS.

As will be described below, we are interested in obtaining the context before deciding whether any authentication challenges are appropriate. These are labeled as *Challenge: {RBA}* and *RBA attributes*. Based on the context values, biometric and non-biometric authentication challenges will be returned to the client. If the risk is low enough, when taking the context into consideration, then the request can simply be authorized.

Should (step-up) authentication be required, an authentication challenge is issued. This is labeled *Challenge: {MFB}*. At this point, depending on policy, one or more biometric and non-biometric authentication samples are collected and sent back to NAAS. After successful authentication, the original resource request is forwarded to the Server application to be fulfilled.

Rather than requiring the user to provide authentication using all of the authentication techniques every time authentication is required, we want to authenticate just enough to allow for access to the protected resources. The observation is that authentication is an interruption to the user's primary task. We are trying to minimize the disruption to the user's work flow. To achieve this goal, we first collect context and then prompt for authentication until there is sufficient confidence that the risk has been sufficiently minimized. This may require multiple rounds of interaction with the user to collect authentication data (e.g., biometric or non-biometric credentials). Figure 8.4 depicts this iterative interaction with the user to collect authentication data.

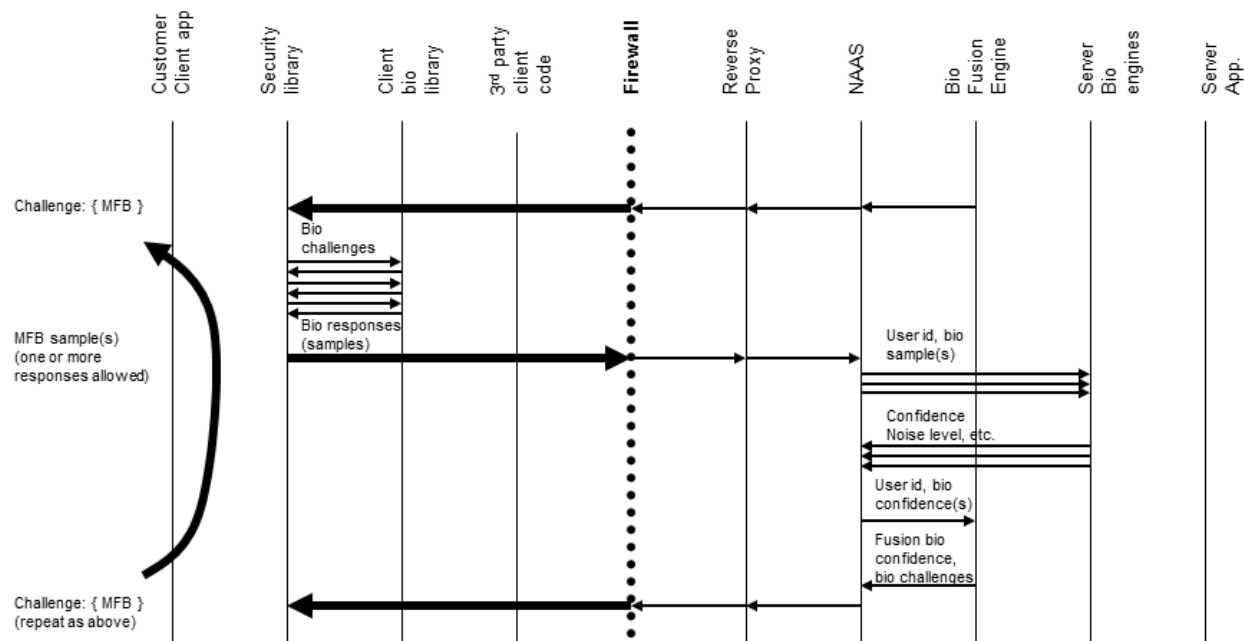


Figure 8.4 Biometric Authentication Iteration

If there is insufficient confidence (e.g., result of biometric fusion), then the client is challenged to send another authentication credential. This repeats until authentication is successful or there are no more authentication modalities available that would enable successful authentication.

There is a cost associated, both to the end user and at the network layer, for using an iterative approach for sending authentication challenges and responses. Each challenge / response iteration requires interaction with the NAAS to get an authentication challenge, and the client must respond with authentication credentials. If, either by policy or through advanced analytics, it can be determined that two or more authentication credentials will be required, then this iterative loop can be optimized to collect a sufficient number of credentials in a single round before sending them to the NAAS.

The policy in the current prototype focuses on a purely iterative approach to authentication challenge response. However, the protocol that we implemented between the NAAS and Security library supports the request of one or more authentication credentials to be sent by the client (via multi-part MIME) to the NAAS. Thus, when there is a more sophisticated NAAS policy, the client can respond with multiple credentials per authentication iteration.

8.2.3 Inline / out of band processing

We identified multiple use cases for the mobile authentication. These included:

- **Inline SDK** – this mode of operation incorporates the authentication in the mobile app. The authentication protocol is performed over the primary communication channel between the mobile app and the NAAS.

This is the most invasive approach for integration since it requires active involvement of the app programmer. The business logic /app must recognize that there is an authentication challenge (e.g., *HTTP 401 Unauthorized*) and call the appropriate library to handle the authentication challenges – biometric & non-biometric. After the authentication library exits, repeat the request for the resource and/or handle any error conditions that may arise.

Any changes to the authentication SDK will require recompilation / linking / deployment of the mobile apps.

From the end user's perspective, since the authentication is part of the mobile app, it appears as a tightly integrated / seamless service. This is a highly desirable usability characteristic.

- **Inline Authentication App** – the mobile app (business logic) needs to recognize that authentication is required (e.g., *HTTP 401 Unauthorized*) and launch the mobile authentication app. Authentication takes place over a secondary (*out of band*) communication channel between the mobile app and the authentication service. Since multiple apps can use a common authentication app, it is straightforward to support single sign-on across the multiple apps on the same device.

If multiple apps were to concurrently require authentication, then the results from a single authentication attempt, whether it be successful or unsuccessful, can be applied to all outstanding authentication requests. This improves overall usability of security across these apps.

There is minimal programmer effort to recognize the need for authentication and invoke the authentication app via a library call (SDK).

If the SDK is upgraded and the upgrade is mandatory to continue to use the NAAS, then the business app will need to be recompiled / linked. However if the functional changes can be contained within the authentication app, only the

authentication app needs to be updated. We feel that this latter case – just updating the authentication app – will be the most common event, with less likelihood of needing to update the business apps.

From the end user's perspective, it is noticeable that there is a separate app for authentication versus the business app since there is only a loose coupling. The visual appearance and/or user experience of the authentication app may be different than for the business app. Depending on the mobile platform and configuration, there is the possibility of additional steps to upgrade / maintain the separate authentication app.

- **Out of band app authentication** – an unmodified mobile app requires authentication services. The advantage from the programmer's and system administrator's perspectives is that the mobile business app remains unmodified. The user's perspective is pretty much the same as with the *inline authentication app* scenario above. The primary disadvantage is that this mode of operation currently does not support devices that are sitting behind network services / devices that perform Network Address Translation (NAT), such as routers used at home or when going through Virtual Private Network (VPN) gateways.
- **Out of band web** – a request is made via a web browser for a network resource requiring authentication, and the authentication app is launched to perform out of band authentication. This is largely the same as *out of band app authentication* except that
 1. the request is being made from a web browser, or other web-based application, instead of from a mobile app,
 2. the web user needs to have a validated identity (e.g., userid), and
 3. instead of launching the authentication app on the same device as the mobile app, the authentication app is launched on all registered / configured devices for the authenticated user. A typical use case is for two factor authentication where the user typically first authenticates on the web and the mobile authentication app is used for one or more authentication factors.

Depending on the design of the libraries and the operating environment, the security administrator may need to configure the reverse proxy / application server with the above operating modes to be used for each of the requested resources. This could be for a specific resource or a set of resources. Preferably, it is best if these operating

modes can be automatically detected via a protocol between the client and security services so that configuration is not required by the administrator.

8.2.4 Biometrics – enrollment and verification

Biometric identify verification is an evolving area, especially for mobile applications. There are many existing vendors for these technologies. We expect more such vendors to emerge as the sensors and processing on mobile devices increases. We assume that the set of biometric verification technologies is extensible and designed the system to accommodate new verification technologies.

The biometric enrollment and verification may be performed on the mobile device, such as Apple's Touch ID for fingerprint verification, or performed in the network. For this project our primary focus was on biometric processing in the NAAS. We did, however, add Apple Touch ID support as well for on-device authentication.

For simplicity, in our current implementation, we are assuming self-enrollment. This would be true for many online (e.g., consumer facing services) services. The enrollment process covers both user identity and biometric enrollment. For simplicity, we also assume that a user will be required to enroll for *all* of the configured biometric modalities. This could be adjusted so that the user or security administrator could choose which enrollments are required and which are optional.

Verification can be performed by sending challenges to the client device for multiple biometric samples to be scored. Or biometric verification samples can be required one at a time, each time checking to see if there is sufficient biometric authentication confidence before sending additional biometric authentication challenges. While we have implemented both approaches, we focused on the latter – one challenge at a time.

There is an independent question as to the lifetime (*time to live*) of a biometric authentication result. Once a biometric sample has been verified, how long should the NAAS authorization algorithm trust this result? If the time interval is too short, we may interrupt the end-user too often for re-authentication. If the interval is too long, an attacker will gain an advantage if s/he can steal the device. Our implementation addresses this dilemma in two ways: (1) user presence detection that signals the NAAS when there is a presumption that the user is no longer in possession of the device, and (2) a scaled *time to live* that automatically degrades the biometric confidence score after a set interval, discarding the biometric authentication result when the confidence score reaches a lower bound threshold. Other *time to live* schemes are feasible.

8.2.5 Risk Authorization

A number of contextual factors are useful for providing robust authentication and risk-based authorization. This general approach is not new. If you use an online social network service, such as *Facebook*², and you were to log into your account from an unusual location (e.g., London instead of Washington), then the Facebook authentication services will present a set of additional authentication challenges to confirm your identity.

We assume that contextual factors influence the risk assessment. As such, the NAAS collects and processes several contextual factors that can be used to influence the timing and number of authentication challenges to send to the mobile device.

Contextual factors remain valid for a period of time. For example, location could remain valid as long as the device does not move a substantial distance. Or, the device is being held by a person is valid for as long as it has not been put down to rest. Since it would be very expensive to continually stream the contextual factors to the network security service, contextual factors have a *time to live*. Our *time to live* for context implementation is similar to the approach taken with biometric authentication, as described above.

Collection of contextual factors has been in use for quite a while in the web environment. What is new is that mobile devices have a richer set of sensors and processing capabilities. We expect the set of sensors useful for context to continue to increase. Similar to what we have done for the biometric data processing, we have made the contextual factors configurable and extensible.

8.2.6 Authentication Challenges

Authorization decisions are based on the value at risk, risk assessment, evaluation of context and authentication factors. In this project, during the authentication process, we are given a value that represents the value at risk. Based on policy, we need to determine whether we have sufficient confidence in the users' identity or we need additional contextual information. From a usability perspective, interaction with a user is costly from a time and cognitive load perspective (e.g., [Trewin2012]). If we can avoid user interruption and interaction to gain confidence in the users' identity, we should have a better user experience. In this project all of the contextual factors we are considering are non-interactive – don't require user interaction. Since risk authorization

² <http://www.facebook.com>

is partially dependent on contextual factors, we focus on obtaining and processing context and risk factors prior to making any authorization decision.

8.2.7 Risk Communication

Communication of risk factors to the end user is dependent on the results of the risk assessment and the decision as to whether there will be any interactive authentication challenges to the mobile device. There may be multiple risk factors resulting in interactive challenges (e.g., biometric authentication requests). We would need to prioritize which risk factor(s) are to be communicated. In practice, due to space limitations as well as the results of our risk communication studies, limited communication of risk is sufficient.

8.2.8 Administration

Any system is not usable if it cannot be configured and maintained. We have users, their devices and capabilities, biometric engines / enrollments and risk factor evaluators. We created interfaces to configure and manage these system elements. This will be discussed further below.

8.3 Procedures

The NAAS may be deployed as an enterprise or cloud-based service to provide mobile authentication and authorization services. The following figure depicts a high level view of the security services as we envision it being provisioned with a reverse proxy server.

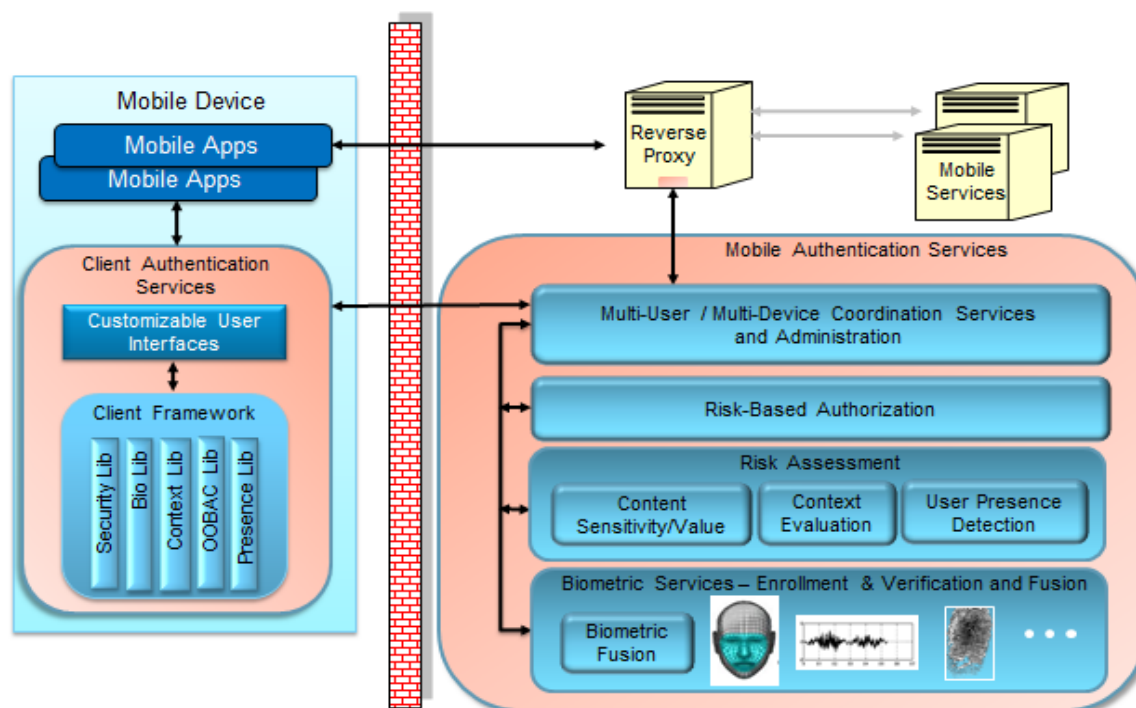


Figure 8.5 Reference System Architecture

The mobile device-based services were previously described. We discuss the details of elements on the right side of Figure 8.5 in the following subsections.

8.4 Results and Discussion

In this section we describe each of the elements of the NAAS and some of the motivation behind the design. The runtime itself can be viewed as having a shim layer to interface with the environment, whether that be a reverse proxy server, mobile application server, or RADIUS server. Behind the shim layer are the core NAAS services as described below.

8.4.1 Shim layer

The shim layer is called from the operating environment to perform the authentication and risk based authorization processing. This layer was implemented as a set of Java Servlets running in an Apache Tomcat 7 server running on Redhat Linux 6.3 or Microsoft Windows 7. IBM DB2 10.1 is used as the SQL database for persistent storage, including user and device registration data. The primary shim layer APIs are as follows:

- **EAI** – Called by the mobile application server or reverse proxy to identify the resource being requested and assign a resource sensitivity value. The sensitivity level can be a numeric quantity (e.g., 0 through 1000), or quantized into discrete security levels (e.g., low, medium, high).

EAI also can be called to test whether further authentication is required. If not, EAI reports that the request is authorized without requiring any interaction with the mobile device or user. If further authentication is required, EAI initiates client-side interaction – *inline* or *out of band* as described above.

- **Login** – The client provides the claimed identity of the user and the mobile device identity. Note that authentication (verification of identity) and authorization decisions are performed by *Access*.
- **Logout** – Called when the client app exits, the user is no longer in possession of the device, or an explicit call is made to terminate the current session. Logout forces a reset of any authentication and context state for the user and mobile device.

- **Access** – This is the primary interface called by the client-side authentication app or SDK to obtain context and authentication challenges, and report context / biometric data. Authorization decisions are also made by Access.
- **EnrollBiometrics** – If the user is not yet enrolled for biometric authentication, this interface generates challenges to the mobile device to collect an appropriate number of biometric samples needed to perform enrollment for each of the configured biometric engines.
- **RegisterUser / RegisterDevice** – If the user or device is not already registered, these services are called to perform self-registration of the user and device.

Each of these shim layer APIs calls a set of underlying core services as described below.

We next describe the major functions of the *shim layer* and the underlying services that they use.

8.4.1.1 EAI

This interface is called multiple times by the reverse proxy / mobile application server. When the user / app is unauthenticated, and out of band authentication is to be performed, the typical pattern is to call EAI twice:

1. The first time EAI is called, a test is performed to determine whether authentication is required. If not, a success status is returned to the caller (reverse proxy / mobile application server). If authentication is required, then the assets at risk are identified and the risk level is quantified (e.g., low, medium, high). Then it is determined as to how the mobile client is to perform authentication – inline or out of band, mobile or web – and a response to the caller initiates the authentication process. If the authentication is to be out of band, then a signal (e.g., *PushNotification*) is used to initiate the mobile authentication client.
2. The second time EAI is called, if the first request was for out of band authentication, then the thread is suspended until the authentication processing completes (client calls to Access). If the request is using inline authentication, then the authentication result should already be available and the thread is not suspended. In either case, once the authentication result is known, the authentication / authorization status is returned to the caller – reverse proxy / mobile application server. This includes the user identity and the authentication status (e.g., success, fail).

For some instances of out of band authentication, the thread requesting the authentication / authorization decision is suspended pending the results of the authentication – client calls to *Access*. When the authentication / authorization decision is completed, the thread executing *Access* will wake up / resume the suspended thread and pick up the authentication / authorization decision and return it to the caller.

The two primary sets of core services employed by *EAI* are *PushNotification* and *ResourceAccess2*. *PushNotification* services communicate via platform-specific (Apple, Google) APIs to launch services on the mobile device for out of band authentication. *ResourceAccess2* is the main class for performing mobile authentication and authorization services and will be described in further detail below.

8.4.1.2Access

This is the main interface for performing authentication and making authorization decisions. The primary purpose of this interface is to collect the data sent via HTTP multi-part MIME and pass the data to *ResourceAccess2*. If there are authentication challenges to be sent to the client, this class formats and returns the challenges to the client. If authentication has been completed or is being performed out of band, then this class handles the caching of authentication / authorization results and causes the resumption of the suspended *EAI* thread.

8.4.1.3Login

This interface obtains the claimed identity of the mobile device user, the device identifier and type of device, and will reset any authentication state if the user identity in the current session has changed – such as when a new user is using the same mobile device.

Since we are supporting self-registration, if the *userid* is not known to the system, the mobile device is redirected to *RegisterUserid* and then to *RegisterDevice*.

8.4.1.4Logout

There are two use cases for this API. The first is if a user wants to terminate the current session with the NAAS or application server. The cached state of the user / device is reset. This includes the context and authentication state. Since this interface is via HTTP(S), the HTTP session cookie is deleted, thus invalidating the current session. Subsequent calls to any of the web APIs will require that the user re-authenticate via *Login* and *Access*.

The second case is if the client device deems that user is not present, the state of the user / device is reset. The HTTP session is preserved so that the client does not need

to call *Login* again. Although subsequent calls to EAI may require calls to *Access* to collect context and authentication credentials (e.g., biometric samples) in order to be authorized for a resource access request.

8.4.1.5 RegisterUser / RegisterDevice

If the userid, userid / device pair or class of device is not registered, one or more of these APIs is called.

RegisterUser performs self-registration of the user. If *Login* were to use *LDAP* or *Active Directory*, then this API would not be called.

RegisterDevice registers the userid / device id pair. This is useful for the *Web out of band* authentication use case where we need to identify all devices owned / registered to a user and the selected devices are to be notified that authentication is required.

8.4.1.6 PushNotification

A set of classes perform PushNotification services:

- register and unregister an association between a device and the device's proprietary (Apple, Google) PushNotification identifier.
- prepare messages to be sent to the mobile authentication app on the device. These messages indicate the authentication app to launch and parameters, such as random identifier that is used as a security token to prevent spoofing attacks.
- Interface with the proprietary PushNotification libraries to send the messages to the mobile devices. PushNotification to a single device or a broadcast to multiple devices is supported by the API.

8.4.2 Core authentication and authorization services

The core set of services coordinate the processing of contextual (risk) factors, (biometric) authentication, risk assessment, authorization decision, authentication challenge generation, and risk communication. These services are managed by what we previously referred to as ResourceAccess2. These services are designed to be independent of the environment in which they are operating. As described above, for this project, a shim layer based on JEE Servlets interface to ResourceAccess2.

8.4.2.1 ResourceAccess2

Each resource access request received by the NAAS is translated into a call to ResourceAccess2 that performs the following set of services:

- Risk factor / context evaluation
- Authentication – multi-factor biometric and non-biometric
- Risk evaluation
- Authorization decision
- Challenge generation – risk factors / context, authentication
- Risk communication
- Audit

Each of these topics are discussed below, with the exception of *risk evaluation* (Section 6.0), and *risk communication* (Section 3.0).

The inputs to the authentication and authorization process are:

- Timestamp of the request
- Claimed user identity – a userid
- Value at risk -- a numeric or quantized value
- Persistent data repository – history of transactions, etc.
- Authentication policy
- Risk / contextual values sent from the client
- Authentication credentials sent from the client – biometric and non-biometric

The outputs from the process are:

- An authorization decision: Challenges (interactive & non-interactive), authorized, denied, or an internal error indicator
- Challenges – interactive and non-interactive
- Risk messages

- Prompts for the challenges
- Any other messages to be sent to the client

It is the responsibility of the caller to ResourceAccess2, such as the shim layer (e.g., Access), to translate this output into a format acceptable to the caller and/or the client device authentication code.

The implementation of ResourceAccess2 is to be stateless, persisting any required history in a data store. This approach was taken so that this class can easily operate in a JEE Servlet environment that requires the servlet itself to be stateless. This design point also assists with scalability whereby a cluster of servers can work from a common repository of persisted authentication and audit state.

While ResourceAccess2 itself needs to be stateless, the authentication and risk / context evaluation is stateful. The design is to rely on a persistent store to maintain the state of the risk / context and authentication evaluations. While the current implementation caches the state in memory, a product will likely cache the state in a persistent store to allow for recovery from software failures / restarts and/or scale-out of the service to support large numbers of concurrent users / authentication requests.

The authentication and authorization process proceeds in the following steps:

1. Create an audit entry for the authentication attempt. This is useful for risk communication, such as an insider attack – use of your mobile device by friends / family / colleague / unknown attacker
2. If the content is not sensitive, then authorize the request. No challenges are sent to the client.
3. If new biometric samples are received from the client and it is expected to be carrying a secret for replay attack detection, ensure that the samples have the expected secret. If not, discard the sample(s), generate an audit record, and (optionally) fail the request – return that the request is not authorized and terminate the authentication session.
4. Retrieve any previous authentication results (e.g., biometric verification) that have not expired due to time-to-live constraints.
5. Process any new authentication credentials received from the client – biometric or non-biometric – and update / add to the set of authentication results
6. If biometric verification scores are available, compute a biometric fusion score

7. Retrieve any previous context / risk factor evaluation results that have not expired due to time-to-live constraints.
8. Process any new context / risk factors received from the client and update / add to the set of current context / risk factors
9. Compute an authorization decision based on the value at risk, context / risk factors and authentication results, subject to the authorization policy
10. If the authorization is granted or denied, no further processing required and the caller is informed of the decision.
11. Context / risk factor challenges are generated based on the current context / risk factor results and authentication policy. Subject to the authentication policy, each of the context / risk evaluators makes a decision as to whether a new challenge is to be sent to the mobile device.
12. Generate risk communication messages to be sent back to the client device.
13. If new challenges are generated, the caller (e.g., EAI or Access) will be informed of the decision – that there are challenges to be sent to the client. Authentication challenges are generated based on the current authentication policy. Subject to the authentication policy, each of the context / risk evaluators makes a decision as to whether a new challenge is to be sent to the mobile device.
14. The caller is informed of the decision – that there are challenges and messages to be sent to the client.
15. To improve usability and responsiveness of the system, we want to minimize user interaction.

When context / risk factor (non-interactive) challenges are generated, we choose not to generate authentication (interactive) challenges. The reason is that context / risk factor evaluation results may provide sufficient confidence such that the authentication policy will be satisfied to allow for the granting of authorization. Notably, these context / risk challenges generally do not require end user interaction. Only when context / risk factors are insufficient to satisfy the authentication policy do we generate authentication challenges since these challenges generally require some form of user interaction – such as the collection of biometric samples, PIN, password or other security credential.

The authentication and context / risk factors are generally associated with userid / device pairs. All of this state is typically maintained in a persistent store. One of the closely tied services is the management of this persistent store such that if the user logs

out, or the client communicates to the NAAS that the user is no longer present with the device, then this state is discarded. This forces the user to re-authenticate to this service before resource access will be authorized.

8.4.2.1.1 Predictive scheduling of challenges

One of the practical problems with contextual factor and authentication data collection is the time required for the mobile device to collect the data. For example, the collection of Bluetooth data (device name, MAC address) from nearby devices polling cycle time is many seconds, sometimes tens of seconds. Waiting for the polling to complete will greatly slow down a challenge to the mobile device to return the list of nearby Bluetooth devices. There are a couple of strategies that can be applied to address this issue. The first is for the mobile device to cache the Bluetooth data when it is received so it can be immediately sent to the NAAS. While a simple strategy, it may be expensive in terms of constant polling for Bluetooth data. It may also become stale if not regularly collected. Alternatively, the NAAS can use a predictive model to decide when authentication is likely to happen and start polling the client device for this data. As the results become available, the mobile device can report it to the security services, or cache the results until the data is requested. While we describe this approach in the terms of context / risk factors such as Bluetooth data, the same approach can be used for collection of biometric data that requires time, such as gait or heart beat analysis. This approach to reducing the overall time to perform authentication has been described in a short paper [KovedWay] and patent filing.

8.4.3 Authenticators and Context / Risk Evaluators

Since the set of authenticators and context / risk evaluators needs to be extensible and configurable, we have defined three types of classes that can be configured into the NAAS:

1. **Authentication verifiers** – verify (biometric) credentials / samples. For biometrics, this is typically verification against an enrollment.
2. **Authentication enrollers** – enroll a user for a (biometric) verifier.
3. **Context evaluators** – processes the context / risk data sent by a client.

8.4.3.1 Authentication Verifier

There are two broad classes of authentication verifiers. The simplest kind verifies a security credential. This may be as simple as QRCode verification. As an example, we implemented Apple Touch ID for Apple devices. This verifier triggers verification of identity – fingerprint verification -- on the mobile device. From the network security

service perspective, we simply need to ask the Apple device whether the user can successfully perform identity verification on the device.

The second kind of verifier requires enrollment, which we will describe in the next subsection.

To make these verifiers configurable in the runtime, we have defined an interface that all of the authentication verifiers are required to implement, and contain the following methods:

- `getName` – typically the name of the class
- `getAuthCategory` – the kind of verifier category – e.g., face, voice, fingerprint
- `verify` – performs the verification process based on the user identity (userid), and values received from the client. The data required from the client (discrete values, files, etc.) is specified when an authentication verifier is configured into the system.

The verification result contains:

- an authentication confidence score
- the minimum and maximum values that may be returned by this scoring algorithm
- verifier-specific response values. These may include noise levels or other values that may be of use to the fusion algorithm, responses to the client including error messages – e.g., user's face was not found in the image
- The authentication category – e.g., face, voice, fingerprint
- The Java class that generated this result
- `addChallenges` – given a confidence score needed by the fusion algorithm, and any prior authentication result generated by this authenticator, generate authentication challenges to send to the client. Challenges may request one or more credentials (e.g., biometric samples). The structure accommodates for the request of more than one credential.
- `addChallengesCount` – if authentication challenges are not required, return the number of authentication credentials that would be required. This is typically used by the client to indicate that an authentication challenge was satisfied. As a result, the client may provide visual feedback that further authentication with this

modality is not needed. However, the client can optionally allow the user to re-enter this security credential.

- `isEnrolled` – provides an indication as to whether the specified user has enrolled for this modality. If enrollment is required, then either the client is instructed to perform enrollment for this modality, or authentication challenges should not be generated for this authentication modality.

8.4.3.2 Authentication Enrollers

Some authentication verifiers require enrollment. This is true for biometrics where the verification is performed by the NAAS rather than on the device (e.g., Apple Touch ID). After several iterations on design, we have settled on a design that has only one operation: *enrollUser*. This operation requires the user identity, the device type, and the input from the client. The data required from the client (discrete values, files, etc.) is specified when an authentication verifier is configured into the system. If the user is not enrolled, and no or insufficient data is provided by the client to perform enrollment, then challenges are returned to be sent to the client.

This interface could, in principle, be merged with the *authentication verifier* interface. In practice, it was useful to have them separated for the purposes of testing during development. There are a limited number of cases where one enroller class may be used by two or more verifiers. In practice, it is possible that implementations could be merged into a single class to simplify deployment.

8.4.3.3 Context Evaluators

To make these configurable in the runtime, we have defined an interface that all of the context evaluators are required to implement, and all contain the following methods:

- `score` – scores risk / context data based on the user identity (`userid`), and values received from the client. The data required from the client (discrete values, files, etc.) is specified when a context evaluator is configured into the system.

The scoring result contains:

- the context type, typically the class name
- an anomaly score
- the scorer-specific response values.
- the class that generated this result

- `addChallenges` – given a confidence score needed by the authentication policy, and any prior authentication result generated by this context evaluator, generate authentication challenges to send to the client. Challenges may require one or more values. The structure accommodates for the request of more than one context value.

8.4.4 Biometric fusion

The results from the biometric verifications are cached for the purpose of being passed to the biometric fusion engine. Based on policy, a subset of biometric authentication evaluation results are to be included in fusion scoring. Since there is a mix of biometric and non-biometric authentication, with a subset to be used for fusion scoring, a filter is applied to the authentication results, passing to the fusion engine only those authentication results to be included in the fusion scoring. The fusion result is then cached for evaluation by the authentication policy.

8.4.5 Authorization policy

There are multiple ways that authorization policy can be specified. Section 6.0 discusses an approach that we explored in this project. Another common approach is to use conditional logic (propositional logic) to make a determination as to whether a user request is to be granted.

In its simplest form, a statement of the form can be used:

*if (2 biometric authentications succeed) **then** {authorize. deny} request*

The reverse proxy server with which we demonstrated integration (ISAM) has such a language for specifying conditions under which authorization will be granted.

To support a system that uses propositional style rules, we created a simple scheme for specifying both authentication and contextual factors that need to be satisfied. The intent is to provide an interface to the set of authentication and context evaluators rather than creating a complete language for specifying authentication conditions. You can consider the set of conditions to be a predicate in the propositional language

Each of the named predicates can be defined as a set of evaluations of one or more authentication evaluators (biometric, non-biometric) or context / risk evaluators. Each evaluator in the set has the properties:

- *Ignored, required, optional*
- *Binary, score, fusion (for biometric authentication only)*

If the evaluator is labeled as *ignored*, then the evaluator is not included in determining the predicate value.

If the evaluator is *required*, then the client must supply data to this evaluator to be included in deciding the predicate value.

If the evaluator is *optional*, then any data supplied by the client will be evaluated and considered in determining the predicate value.

The result of an evaluator may result in a numeric value, as we have seen with the biometric verifiers. Such evaluators may have success threshold scores defined that can be used to decide whether the evaluator has “succeeded”. For example a fingerprint verifier may generate a score between 0.0 and 1.0. Based on testing, it may be determined that a value over 0.3 is considered a successful match. Thus we can set a policy such that any biometric verification score over 0.3 will be considered a success. These success scores can be set on a per-evaluator basis.

Other evaluators generate a binary result – e.g., pass / fail. For example, in the case of iOS Touch ID, the NAAS receives a binary decision (*pass*, *fail*) from the client.

In the current system, we use fusion scoring only for the biometrics. The concept of fusion could be extended to non-biometric factors as well. If a biometric evaluator is labeled as *fusion*, then the results of the evaluators labeled as *fusion* will have their scores sent to the biometric fusion engine (see Section 5.0) to obtain a fusion score. It is the fusion score, not the results from the individual biometric evaluators, that is used as input to determining the predicate value. A threshold value is also defined for the fusion score as the criterion for the pass / fail value for the authorization policy.

For the purposes of demonstration, we chose a simple scheme for determining the predicate, using the criteria just described as input into the decision process. For each named predicate p , we define a set of evaluators to be included in the predicate evaluation. We allow for a mix & match approach, where binary, score and fusion may be selected for the different evaluators. Specifically,

- *fusion* – the biometric fusion score must reach the threshold score in order to succeed. This is required. There is a separate parameter that allows for multiple biometric authentication retry attempts if the threshold is not met. We implemented this in our prototype.

If multiple biometrics are included in the fusion score, then a policy could be defined to make recommendations as to which biometrics are most likely to generate scores that will result in a successful fusion score. Authentication

challenges can then be sent to the client to collect new biometric samples for evaluation and fusion scoring. Such an algorithm is outside the scope of this project.

- *score* – if a score-based evaluator returns a value that meets or exceeds the specified threshold, then the evaluator is deemed to have succeeded. As with fusion, if multiple scoring evaluators do not succeed, then a policy could be defined to allow for retries.
- *binary* – similar to *score*, the evaluator either succeeded or failed. As with fusion, if multiple binary evaluators do not succeed, then a policy could be defined to allow for retries.

In our prototype, the predicate value

$$p = (fusion \geq rfs) \ \&\& \ (count(score(i)) + count(binary(j)) > rsc)$$

Where *rfs* is the required fusion score for the predicate to succeed.

Where *rsc* is the required success count for the predicate to succeed.

And the function *count()* counts the number of *score* and *binary* evaluators in the predicate evaluator set that return a success value.

8.4.6 Administrative functions

We found the following administrative functions to be useful for managing the system.

8.4.6.1 Runtime configurable parameters

A number of configurable parameters have been defined:

- Temporary files – where to store temporary files uploaded from the mobile device. These files are typically the biometric data to be evaluated.
- How long to wait for an authentication client to respond before failing an authorization request. In the absence of a timeout, it is possible that the server would run out of threads, exposing the system to a denial of service attack.
- The name of the authentication app to be started by a PushNotification request. The names are platform specific since Apple and Google use different mechanisms to launch applications from a background task.

8.4.6.2 Registries

We defined four different registries: *user*, *device*, *device type*, *PushNotification*.

1. In a typical deployment, users will be defined in a user registry such as LDAP or Active Directory. For small standalone deployments of the technology, a small user registry is required. Viewing, adding and removing users are the key functions.
2. Originally we created a device registry that was independent of the user who was using the device. However, as we added the functions to support the Web use case where we needed to associate devices with users, we bound userids to the registered devices. The device registration includes the unique device identifier, the userid(s) associated with the device, manufacturer, model and whether the device is to be used when broadcasting authentication requests for the user. The manufacturer often implies the operating platform (iOS, Android, etc.). This is needed when sending PushNotification messages to the device since the PushNotification services are platform (operating system) specific. The device model may be useful for generating platform / device specific authentication or context challenges since sensors vary by platform or device type.
3. A device type registry is useful for registering device capabilities (camera, fingerprint, Bluetooth, NFC, etc.). We did not make use of this registry in the current implementation.
4. The PushNotification registry contains the same basic information as the device registry, but also the most recently registered IP address for the device. The IP address is needed for the use case where an unmodified app is making a resource request and needs to be authenticated. By looking up the IP address, a PushNotification request can be made to the device to initiate authentication. In principle, this registry could be merged with the device registry.

8.4.6.3 Authentication and context / risk evaluators

As previously noted, the NAAS supports an extensible set of context / risk evaluations. We provide a GUI to select the authenticators and context/risk evaluators to be loaded and be available for use by an authorization policy during runtime. Some of these evaluators run in separate processes, so additional parameters are required to specify the location of these remote servers.

8.5 Conclusions

We successfully created a runtime system that can integrate all of the key elements of a multi-factor authentication and authorization service – secure communication with a mobile device, multiple modes of interacting with the mobile device to perform authentication (inline and out of band), multi-factor biometric authentication, risk based authorization and risk communication. We successfully combined these services with an existing network based authorization service (reverse proxy) and a web based mobile application service.

In addition, we created runtime services to address the needs of three classes of users of the system: end users requiring authentication and authorization, developers, and administrators. We have done so in a way that is extensible and meets a wide range of use cases – from traditional integrated apps, to single sign-on, to web and IoT authentication using mobile devices. When we started, we had not fully appreciated the need to address the needs of both the programmers and administrators. Security almost always has a need for an administrative interface for, at least, configuring policy. From our discussions with customers we discovered that ease of deployment is a critical success factor. We need to create a flexible set of services that would enable these administrators to define a broad range of security policies from the existing policy definition interfaces. More important is the need to provide ease of integration into existing and new apps. Programmers, in general, lack the sophistication and expertise needed to do complex security programming. As a result, we created multiple ways to integrate our technology into new and existing apps. The response from customers about our design has been very positive.

9.0 REFERENCES

9.1 Risk Perception and Communication

- [Acquisti and Grossklags, 2005] Acquisti, A., Grossklags, J. "Privacy and rationality in individual decision making". *IEEE Security and Privacy* 2 (2005). doi:10.1109/MSP.2005.22
- [Braunstein, Granka and Staddon, 2011] Braunstein, A., Granka, L. and Staddon, J. "Indirect content privacy surveys: measuring privacy without asking about it" In Proc. 7th Symposium on Usable Privacy and Security (SOUPS '11), p1
- [Bravo-Lillo, Crabor, Downs, and Komanduri, 2011] Bravo-Lillo, C., Cranor, L. F., Downs, J., Komanduri, S. "Improving Computer Security Dialogs". *Human-Computer Interaction – INTERACT 2011*. Lecture Notes in Computer Science, Springer. 6949 (2011). doi:10.1007/978-3-642-23768-3_2
- [Egelman, Cranor, Hong, Egelman and Hong, 2008] Egelman, S., Cranor, L. F., Hong, J., Egelman, S., Hong, J. "You've Been Warned: An Empirical Study of the Effectiveness of Web Browser Phishing Warnings". In *Proc. CHI '08*. doi:10.1145/1357054.1357219
- [Evans 2003] Evans, J. "In two minds: dual-process accounts of reasoning". *Trends in Cognitive Sciences* 7 (2003). doi:10.1016/j.tics.2003.08.012.
- [Fahl, Harbach, Acar and Smith, 2013] Fahl, S., Harbach, M., Acar, Y., Smith, M. "On the ecological validity of a password study". In *Proc. 9th Symposium on Usable Privacy and Security (SOUPS '13)*, pp1-15.
- [Felt, Reeder, Almuhiemedi and Consolvo, 2014] Felt, A.P., Reeder, R., Almuhiemedi, H., Consolvo, C. "Experimenting at scale with google chrome's SSL warning". In *Proc. CHI '14*. doi: 10.1145/2556288.2557292
- [Herley, 2009] Herley, C. "So long, and no thanks for the externalities: the rational rejection of security advice by users". In *Proc. NSPW '09*. doi: 10.1145/1719030.1719050
- [Keith, Thompson, Hale, Lowry & Greer, 2013] Keith, M., Thompson, S., Hale, J., Lowrey, P. and Greer, C. 2013. "Information Disclosure on Mobile Devices: Re-Examining Privacy Calculus With Actual User Behavior". *Int. Journal Of Human-Computer Studies* 71 (2013), 1163-1173.

- [Peer, Vosgerau and Acquisti, 2013] Peer, E., Vosgerau, J., Acquisti, A. "Reputation as a sufficient condition for data quality on Amazon Mechanical Turk". *Behavior Research Methods* (2013). doi:10.3758/s13428-013-0434-y
- [Trewin, Swart, Koved, and Singh, 2016] Trewin, S., Swart, C., Koved, L., and Singh, K. "Perceptions of Risk in Mobile Transactions". Proceedings of the Mobile Security Technologies 2016 workshop. IEEE.
- [Tversky and Kahneman, 1981] Tversky, A., & Kahneman, D. "The framing of decisions and the psychology of choice". *Science* 211 (1981).

9.2 Human-Computer Interaction

- [Bangor, Kortum and Miller, 2008] Bangor, A., Kortum, P. T. and Miller, J. T. "An empirical evaluation of the system usability scale". *International Journal of Human-Computer Interaction*, 2008.
- [Brooke, 1996] Brooke, J. SUS: A quick and dirty usability scale, pages 189–194. Taylor and Francis, 1996.
- [Molich and Nielsen, 1990] Molich, R., and Nielsen, J. (1990). "Improving a human-computer dialogue", *Communications of the ACM* 33, 3 (March), 338-348
- [Nielsen and Molich, 1990] Nielsen, J., and Molich, R. (1990). "Heuristic evaluation of user interfaces", *Proc. ACM CHI'90 Conf.* (Seattle, WA, 1–5 April), 249-256
- [Nielsen, 1994] Nielsen, J. (1994). "Heuristic evaluation". In Nielsen, J., and Mack, R.L. (Eds.), Usability Inspection Methods, John Wiley & Sons, New York, NY
- [Sauro 2011] Sauro, J. "Measuring usability with the System Usability Scale (SUS)", 2011. <http://www.measuringusability.com/sus.php>. Accessed March 6, 2015.

9.3 Biometrics

- [Poh and Kittler, 2012] Poh, N. and Kittler, J. "A Unified Frame work for Biometrics Expert Fusion Incorporating Quality Measures", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 34(1):3-18, 2012.
- [Daas] Dass, S. C., Nandakumar, K. and Jain, A. K. "A principled approach

- Nandakumar and Jain, 2005] to score level fusion in multimodal biometrics systems”, *Proc. Audio- and Video-based Biometric Person Authentication (AVBPA) 2005*, pp. 1049-1058, Rye Brook, NY, July 2005.
- [Poh and Bengio, 2005] Poh, N. and Bengio, S. “Improving Fusion with margin-derived confidence in biometric authentication tasks”, *5th Int’l. Conf. Audio- and Video-Based Biometric Person Authentication (AVBPA)*, LNCS 3546, pages 474-483, 2005.
- [Nandakumar, Chen, Jain and Daas, 2006] Nandakumar, K., Chen, Y., Jain, A. K. and Dass, S. C. “Quality-based score level fusion in multibiometric systems”, *ICPR 2006*.
- [Jain, Nandakumar and Ross, 2005] Jain, A. K., Nandakumar K., and Ross, A. “Score normalization in multimodal biometric systems”, *Pattern Recognition*, 2005.
- [Poh, Bourlai and Kittler, 2009] Poh, N., Bourlai, T. and Kittler, J. “A multimodal Biometric Test Bed for Quality-dependent, cost-sensitive and client-specific score-level fusion algorithms”, *Pattern Recognition Journal*, vol. 43, no. 3, pp. 1094–1105, 2009.
- [VeriLook] <http://neurotechnology.com/verilook.html>. Accessed March 6, 2015.
- [VeriFinger] http://neurotechnology.com/vf_sdk.html. Accessed March 6, 2015.
- [VeriSpeak] <http://neurotechnology.com/verispeak.html>. Accessed March 6, 2015.

9.4 Risk-Based Access Control

- [Bol07] Bolstad, William M. Introduction to Bayesian Statistics, Wiley-Interscience, 2007
- [Bol11] Bolstad, William M. Understanding Computational Bayesian Statistics, Wiley Series in Computational Statistics, September 21, 2011
- [Chen07] Cheng, P., Rohatgi, P., Keser, C., Karger, P.A., Wagner, G.M., Schuett, A., Reninger, A.S. “Fuzzy Multi-Level Security : An Experiment on Quantified Risk-Adaptive Access Control”, 2007 *IEEE Symposium on Security and Privacy*, May, 2007

- [CJB11] Christensen, Ronald. Bayesian Ideas and Data Analysis, Wesley Johnson, Adam Branscum, Timothy E. Hanson, CRC press, 2011
- [Dar09] Darwiche, Adnan. Modeling and Reasoning with Bayesian Networks, Cambridge University Press, April 6, 2009
- [GL06] Gamerman, D., and Lopes, H. F. Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference, 2nd Ed., Chapman & Hall/CRC Texts in Statistical Science, May 10, 2006
- [GP S13] Pattinson, M. "GLOBAL POSITIONING SYSTEM (GPS) PERFORMANCE, JANUARY TO MARCH 2013, QUARTERLY REPORT", UK NATS Services Ltd (NSL), April 15, 2013.
http://www.caa.co.uk/docs/2004/201305GNSS2013Q1_SPS_Performance.pdf
 Accessed March 6, 2015.
- [GPS14] "Global Positioning System (GPS) Standard Positioning Service (SPS) Performance Analysis Report, Report #86, Reporting Period: 1, April 30, June 2014", William J. Hughes Technical Center NSTB/WAAS T&E Team, Atlantic City International Airport, NJ 08405, July 31, 2014
http://www.nstb.tc.faa.gov/reports/PAN86_0714.pdf
 Accessed March 6, 2015.
- [GPSa] "US Government Web Page on GPS Accuracy"
<http://www.gps.gov/systems/gps/performance/accuracy/>
 Accessed March 6, 2015.
- [JSM97] Jang, J.R., Sun, C., Mizutani, E. Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence, Prentice Hall, September 26, 1997
- [LL96] Lin, C., Lee, C.S.G. Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems, Prentice Hall, May, 1996

[LLC10] Liang, F., Liu, C. Carroll, R. Advanced Markov Chain Monte Carlo Methods: Learning from Past Samples, Wiley Series in Computational Statistics, August 23, 2010

[NIBE] "Beta Distribution in NIST Engineering Statistics Handbook".
<http://www.itl.nist.gov/div898/handbook/eda/section3/eda366h.htm>
Accessed March 6, 2015.

9.5 Mobile Client and Security Services

[PhoneGap] Adobe PhoneGap. <http://phonegap.com/> Accessed March 6, 2015.

[Sencha] Sencha. <http://www.sencha.com> Accessed March 6, 2015.

[Worklight] IBM Worklight.
<http://www-03.ibm.com/software/products/en/mobilefirstfoundation>
Accessed March 6, 2015.

[FriendCaster] FriendCaster for Facebook. <http://friendcasterapp.com/>
Accessed March 6, 2015.

[RFC2388] "Returning Values from Forms: multipart/form-data". IETF.
<https://www.ietf.org/rfc/rfc2388.txt>
Accessed March 6, 2015.

[Apple] "didReceiveRemoteNotification".
https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIApplicationDelegate_Protocol/#!/apple_ref/occ/intfm/UIApplicationDelegate/application:didReceiveRemoteNotification
Accessed March 6, 2015.

[GCM] "Google Cloud Messaging for Android". Google.
<https://developer.android.com/google/gcm/index.html>
Accessed March 6, 2015.

[ISAM] IBM Security Access Manager for Web.
<http://www.ibm.com/software/products/en/access-mgr-web>
Accessed March 6, 2015.

[ISAM4Mobile] IBM Security Access Manager for Mobile.
www.ibm.com/software/products/en/access-mgr-mobile
Accessed March 6, 2015.

9.6 Network Authentication and Authorization Services

[Felt] <https://twitter.com/apf/status/551083956326920192>
Accessed March 6, 2015.

[HOTP] “HOTP: An HMAC-Based One-Time Password Algorithm, Request for Comment 4226”, The Internet Society, 2005.
<http://www.ietf.org/rfc/rfc4226.txt> Accessed March 6, 2015.

[ISAM] “IBM Security Access Manager for Mobile”. <http://www-03.ibm.com/software/products/en/access-mgr-mobile>
Accessed March 6, 2015.

[KovedWay] Koved, L. and Zhang, B. “Improving Usability of Complex Authentication Schemes Via Queue Management and Load Shedding”. *Symposium on Usable Privacy and Security 2014*. Menlo Park, CA, July 9, 2014.
http://cups.cs.cmu.edu/soups/2014/workshops/papers/queue_koved_18.pdf Accessed March 6, 2015.

[RADIUS] “Remote Authentication Dial In User Service (RADIUS)”
<http://en.wikipedia.org/wiki/RADIUS> Accessed March 6, 2015.

[SinghKoved] Singh, K. and Koved, L. “Practical out-of-band authentication for mobile applications”. *Proceedings of the 13th ACM/IFIP/USENIX International Middleware Conference (Industrial Track)*, 2013

[Trewin2012] Trewin, S., Swart, C., Koved, L., Martino, J., Singh, K., Ben-David, S. “Biometric Authentication on a Mobile Device: A Study of User Effort, Error and Task Disruption”. *Annual Computer Security Applications Conference (ACSAC) 2012*. Orlando, Florida. December 6, 2012.

10.0 SYMBOLS, ABBREVIATIONS AND ACRONYMS

BN – Bayesian Network

CPT – Conditional Probability Table

d.f. - degrees of freedom

ERR – Equal Error Rate

F - F-test value

FAR – False Accept Rate

FRR – False Reject Rate

GUI – Graphical User Interface

HIT - Human Intelligence Task

HTML - Hypertext Markup Language

ISAM – IBM Security Access Manager

IT - Information Technology

IP address – Internet Protocol address

M - Mean

MAC address – media access control address.

NAAS – network authentication and authorization service

NAT – Network Address Translation

p - probability

pdf – Probability Distribution Function

PB -apps – mobile applications

PIN – Personal Identification Number

pmf – Probability Mass Function

PushNotification – a short message sent to a mobile device

RBA – Risk Based Authorization

ROC – Receiver Operating Characteristics

SD - Standard Deviation

SDK – software development kit.

SSL – Secure Sockets Layer

SUS - System Usability Score

Tomcat – an open source JEE Servlet runtime environment.

TVR – Trust-Value-Risk

VPN – Virtual Private Network

11.0 GLOSSARY OF TERMINOLOGY

HTML - Hypertext Markup Language

Intrinsic Trust - trust places on a user independent of any risk/context

ISAM – IBM Security Access Manager

IP address – Internet Protocol address

JEE Servlets – Java Enterprise Edition Servlets is a web programming model that utilizes software written in the Java programming language and runs in an environment, also known as a container, that specifically handles web HTTP / HTTPS requests and responses. Servlets must be stateless; all state information must be stored in a data repository.

LDAP - Lightweight Directory Access Protocol is a directory service protocol often used by an authentication service

MAC address – media access control address. This is the unique identifier used by a network communication device to provide network device addressability.

Microsoft Active Directory – a user identity directory service

Multi-part MIME – allows for the exchange of multi-media content over the internet, including over HTTP / HTTPS. For example to transmit / receive multiple media files (e.g., audio, video) in a single request.

NAT – Network Address Translation maps an IP address from one network to a different IP address on another network. This often is performed at network boundaries, such as between a corporate network and the internet, or from a home network to the internet.

Out of band – use of a secondary communication channel, often for sending signaling or control messages, that is independent of the content being transmitted over a primary communication channel.

PIN – Personal Identification Number that is typically a few digits long

pmf – Probability Mass Function

PushNotification – a short message sent to a mobile device that wakes up one or more apps. These apps may take specific actions, such as presenting an alert or perform a computation.

QRCode – a popular machine readable array of black and white squares that can be used to encode text, URLs or other content. These squares are readable by many devices, including mobile phones.

RBA – Risk Based Authorization

Reverse proxy – a proxy server that retrieves resources on behalf of a client from one or more servers. In the context of this paper, the reverse proxy performs security functions – specifically authentication and authorization.

SDK – software development kit. Library code, and perhaps tools, to enable the integration of a set of functionality into an application or service.

Shim layer – a thin layer of software to provide mapping API parameters and functions to the underlying logic

SSL – Secure Sockets Layer performs confidentiality and integrity service over the internet

Stepped up authentication – requiring additional security credentials in order to successfully authenticate. Often used when a resource access request is deemed to be high(er) risk.

Tomcat – an open source JEE Servlet runtime environment.

TR1-4 - Technical Report 1-4PushNotification – allows a server / service to send a short message to a mobile app without needing to open / maintain a network connection. This message typically wakes up the mobile app to perform an action or service.

TVR – Trust-Value-Risk – an authorization policy model that aims to have separate out *trust, value and risk* as the primary inputs to the model for evaluation

VPN – Virtual Private Network extends a private network over a public network. This enables a device on a public network to securely access resources from a private network.