# CA Rapid App Security

**Simplified, Comprehensive Authentication for Modern Applications**

Kapil Lohia, VP, Business Unit Strategy
kapil.lohia@ca.com
JUNE 2017

# Digital Transformation is Underway

>2.8B

smartphone users by 2020[2]

>38B

connected devices by 2020[3]

>1.8B

users of intelligent digital assistants by 2021[4]

>50%

of breaches exploited legitimate passwords in 2015[5]

Sources:
1.  Vanson Bourne
2.  Statista
3.  Juniper Research
4.  Statista
5.  DARKReading

**#1** Security concerns are the top obstacle in the app economy   -- Vanson Bourne Market Research

# Impact of Digital Transformation on Banking

## MOBILE BANKING NOW

Weekly mobile bankers exceeded that of weekly branch bankers in 2015 for first time ever[1]

## MOBILE BANKING FUTURE

>2B

Mobile users will have used their devices for banking purposes by the end of 2021[2]

## MOBILE PAYMENTS

~70%

Of all mobile users in the U.S. will make a mobile payment in 2017[3]

## MOBILE WALLET

249%

Increase in mobile wallet transactions on year over year basis in 2016 report[4]

Sources:
1. Javelin Strategy
2. Juniper Research
3. Tech Crunch
4. The Street

# The Challenges to Software Development in the Application Economy

How do I create:

- A native security experience that is convenient, frictionless, and makes it fast to add to new security modalities to apps

- Secure enough to meet audits & compliance requirements

- Provides "developer velocity": maximum value without high costs to develop, operate, or support?
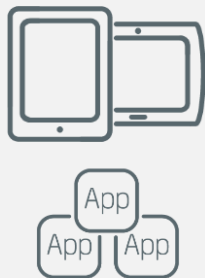
**Convenience**

**Security**

**Costs**

# What if you could…

| Authenticate Users via Wide Array of Credentials | Connecting from Mobile Devices & Mobile Apps | Securing the Device to Server Communications | Transparently Analyze Risk based on Contextual Data | Initiate Step-up Authentication when Risk is High |
|---|---|---|---|---|

## From a Single Solution?

# Introducing CA's Rapid App Security Solution

A comprehensive authentication platform for the application economy  - mobile, web, IoT

- Fraud Prevention in real-time

- Robust authentication methods inclusive of contextual authentication

- Faster & convenient means to secure Apps

- Strengthen consumer confidence and enhance brand value by creating a simplified user experience

*Built on CA Mobile API Gateway and CA Advanced Authentication*

# Rapid App Security Solution from CA



Install Device Certificate

DNA

PKI

OTP

Risk Analysis

Social Login

Mobile & Cross Device SSO

OpenID

Universal SDK

Applications

Devices

Application Services

Authentication Gateway

User Store

Data Store

Identity Processes

Auth and Risk Server

✓ Risk Engine
✓ Rules & Policies
✓ Strong Authentication
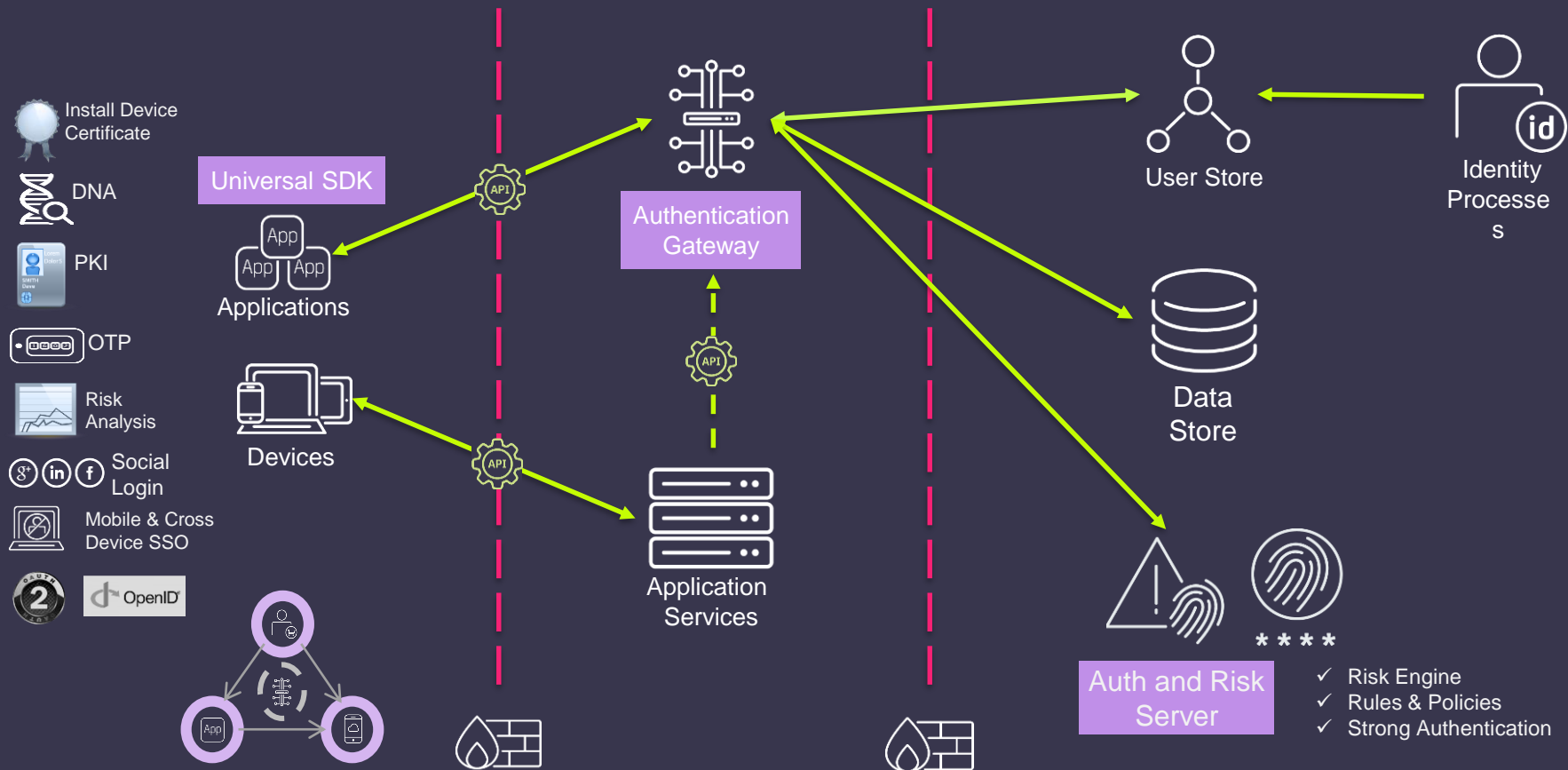
# Additionally…Authentication and API Security



Applications

Devices

Universal SDK

Authentication & API Gateway

User Store

Identity Processes

Data Store

Auth and Risk Server

* * * *

Application Services
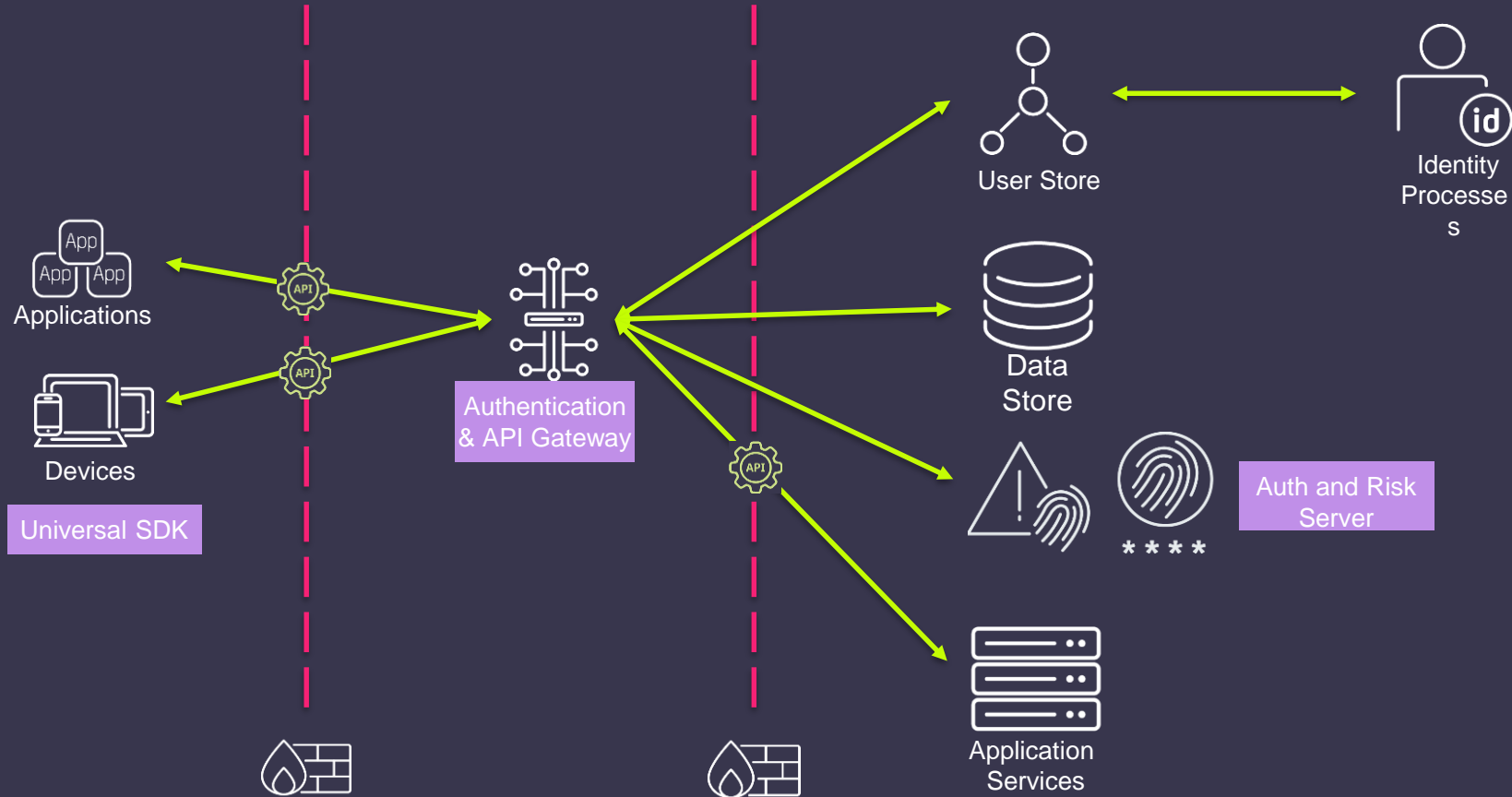
# CA's Rapid Security Authentication Architecture & Components

*Typical Flow for Consumers to access Online Applications via Mobile App*



Consumers → Mobile Devices → Mobile App → Web Services → Applications

*New Components introduced by the CA's Rapid App Security solution*



Consumers → Mobile Devices → Mobile App → Gateway → Applications

Universal SDK

Auth & Risk Server
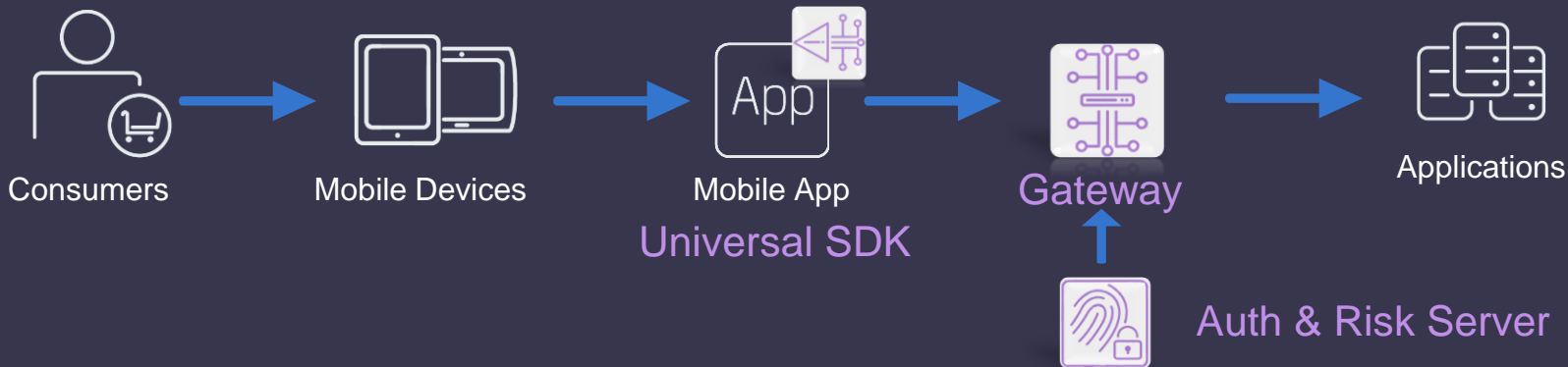
# Security that Elevates the Experience

## Customer/Employee

Better experience

Security based on the level activity being performed

Improved trust without impact to user experience

## Developer

Security that is faster to implement

Security that is cheaper to implement

Security that is easier to implement (especially for non-security experts)

## Organization

Reduced risk, with stronger security and superior authentication options

Improved and less costly security enables higher value services via mobile

Manage security with security experts, rather than in each app

# Rapid App Security with CA Security Solutions



Universal SDK

Applications

Devices

Authentication
Gateway

Application
Services

Access Gateway
Secure Proxy

Policy Server

User Store
CA Directory

CA Identity Manager

CA Single Sign-On

Data
Store

Auth and Risk
Server

✓ Risk Engine
✓ Rules & Policies
✓ Strong Authentication
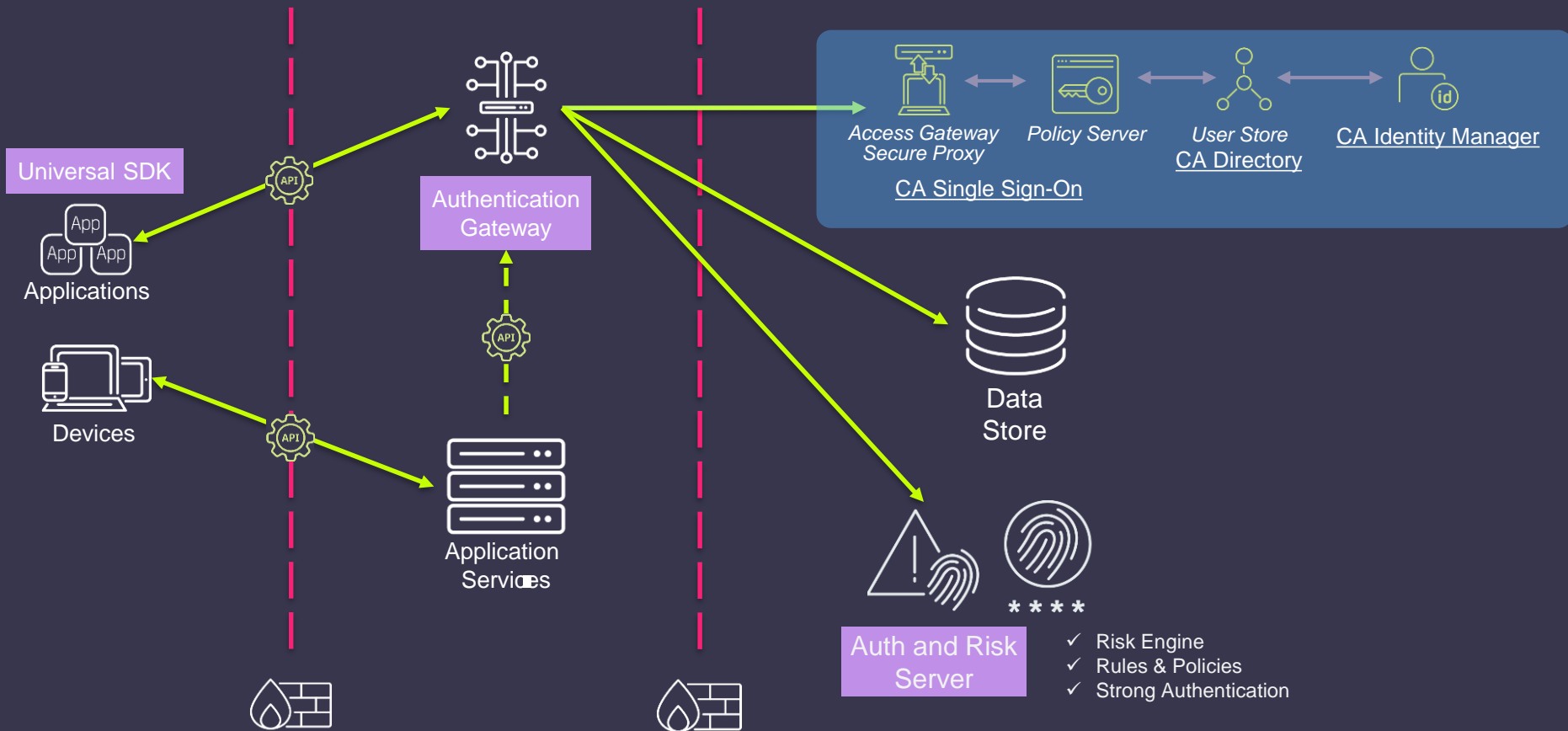
# "Three-way Trust" Enables Greater Security



USER

Uses App

Uses device, has session

APP

Runs on
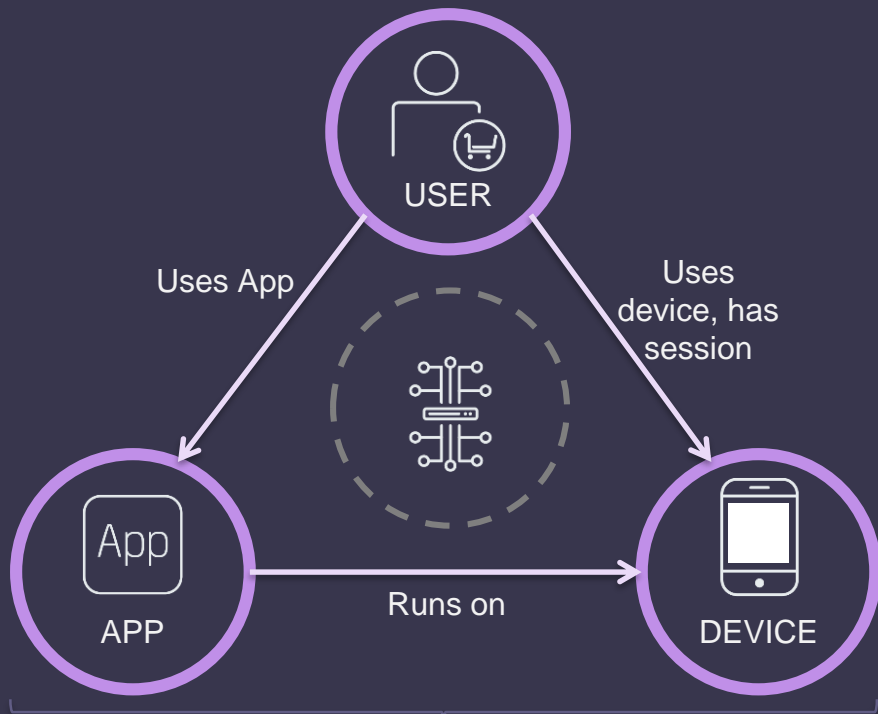
DEVICE

## Rapid App Security Solution:

- Identifies user, app, and device

- Learns and tracks relationship between the three entities

- Three-way trust relationship is registered & authenticated to enterprise gateway

- Elevates authentication or prevents access when pattern deviates from norm

OAUTH 2 OAUTH

OpenID®
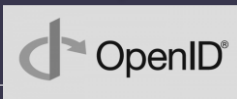
PKI

# Rapid Integration with Native Mobile Apps



CA MAG AA - Multi Step Authentication :: Credential type AuthID
SDK Method loginAuthID()

**Representation of the number of API calls that need to be written to enable MFA for mobile app without CA solution**

**All Reduced to one API Call for the App Developer with CA solution**

**loginWithAuthID (User Id, AuthID Pin)**

www.websequencediagrams.com
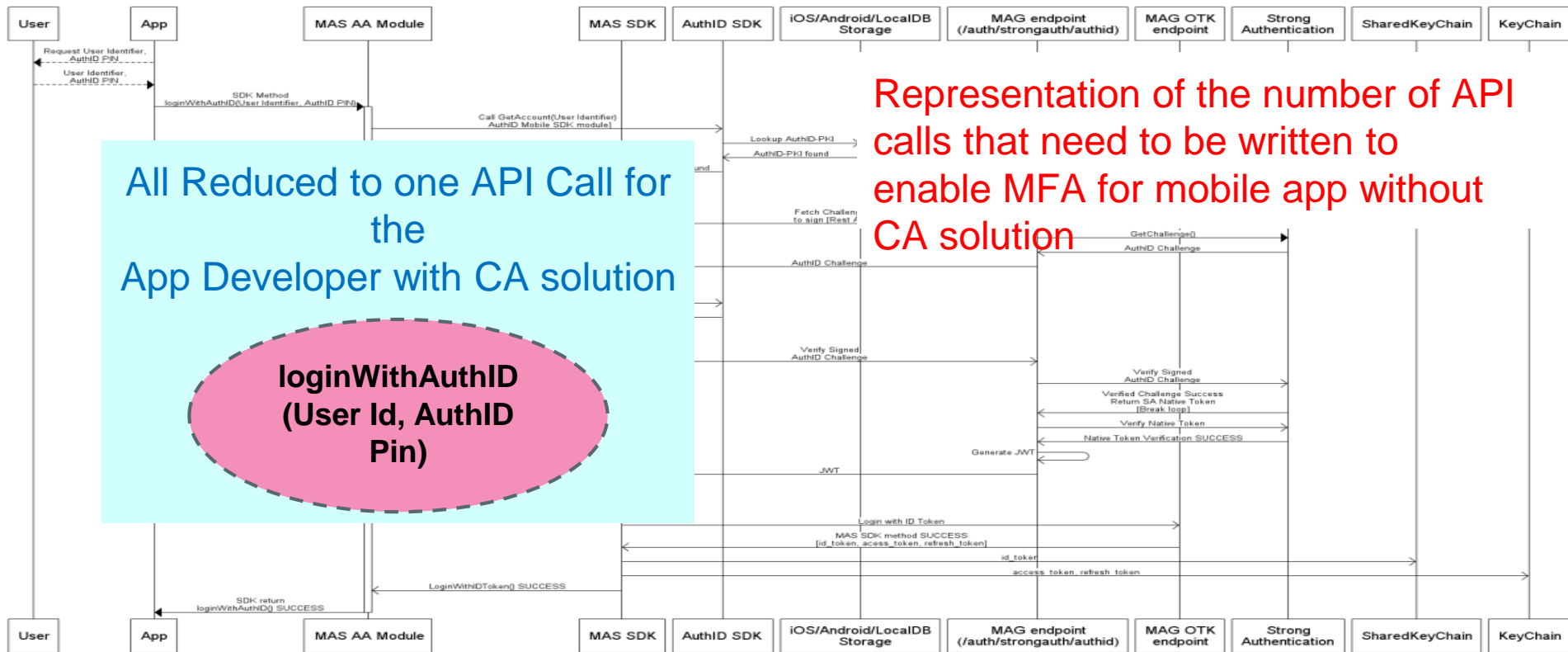
# Rapid App Security Use Case

## HealthCare

Customer experience

Security & Privacy

Integration

## Finance

Contextual Authentication

Risk Mitigation

Open Banking

## Retail

Customer experience

Risk Mitigation

Payment security

## Manufacturing

Internet of Things(IoT)

Geo Fencing

Customer experience

## Automotive

Customer experience

Internet of Things(IoT)

Payment Security

Better End User experience with higher security

# Finance Customer Stated Objectives

- Provide more and better services with mobile (digital)
  - Higher value services => higher risk

- Make the authentication experience easier
  - Device + PIN is more secure than user/pw

- Stronger authentication -- across channels

- Personal security management center
  - Set your limits, notifications, verifications

- Reduce call center volume with self-service
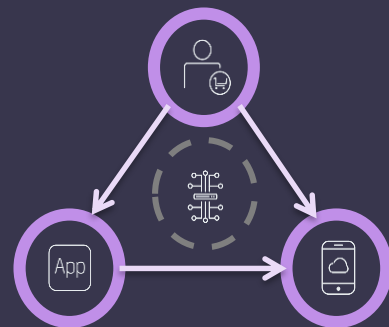
❑ Tighter "cryptographic binding"
❑ Improved security of the app and the device
❑ Tokenizing credentials
❑ Stronger and risk based identity checks

# Telco and Utility Example

## Mobile bill payment – Improve Customer Experience

| Push notification reminder for bill-pay | → | Frictionless login with fingerprint | → | End user login via Rapid App Security, DDNA, Location, Risk Analysis | → | Bill Payment successful |
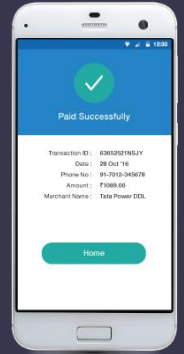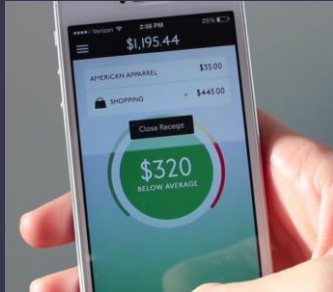


End user Authentication, Device DNA, Risk score analysis, Compliance

**Presenter**

Kapil Lohia
Kapil.lohia@ca.com
VP, Business Unit Strategy

@cainc

slideshare.net/CAinc

linkedin.com/company/ca-technologies

**ca.com/API**

# Rapid App Security SDK Architecture

MASConnecta
- MASDevice
- MASGroup
- MASUser

MASIdentityManagement
- MASGroup
- MASUser

MASStorage

MASUI

· · ·

Auth ID

Device DNA

MASFoundation
- MASUser
- MASApplication
- MASGroup
- MASDevice
- MASService
- MAS
- MASFile
- MASObject

**Legend:**
- Framework
- Model Object
- Model Object Extension

# CA MAG AA - Multi Step Authentication :: Credential type AuthID
## SDK Method loginAuthID()



| User | App | MAS AA Module | MAS SDK | AuthID SDK | iOS/Android/LocalDB Storage | MAG endpoint (/auth/strongauth/authid) | MAG OTK endpoint | Strong Authentication | SharedKeyChain | KeyChain |

Request User Identifier, AuthID PIN

User Identifier, AuthID PIN

SDK Method loginWithAuthID(User Identifier, AuthID PIN)

Call GetAccount(User Identifier) AuthID Mobile SDK module]

Lookup AuthID-PKI

AuthID-PKI found

AuthID-PKI found

Calls end-point "/auth/authIDPKI" (GET Method) Method MAS.invokeUnprotected() to be implemented

Fetch Challenge to sign [Rest API]

GetChallenge()

AuthID Challenge

AuthID Challenge

AuthID Challenge

Call SignWithAccount() AuthID Mobile SDK module]

Return Signed Challenge

Calls end point "/auth/authIDPKI") (POST Method) Method MAS.invokeUnprotected() to be implemented

Verify Signed AuthID Challenge

Verify Signed AuthID Challenge

Verified Challenge Success Return SA Native Token [Break loop]

Verify Native Token

Native Token Verification SUCCESS

Generate JWT

JWT

JWT

Call MAS SDK method Mas.LoginWithIDToken()]

Login with ID Token

MAS SDK method SUCCESS [id_token, acess_token, refresh_token]

id_token

access token, refresh token

LoginWithIDToken() SUCCESS

SDK return loginWithAuthID() SUCCESS

# CA MAS AA - Multi Step Authentication :: TouchID based loginAuthID()



**Participants:** User, App, MAS AA Module, MAS SDK, AuthID SDK, iOS/Android/LocalDB Storage, MAG endpoint (/auth/strongauth/authid), MAG OTK endpoint, Strong Authentication, SharedKeyChain, KeyChain

- User → App: Request User Identifier, AuthID PIN
- App → User: User's fingerprint
- App: Derives AuthID PIN (from fingerprint) [User Identifier known]
- App → MAS AA Module: SDK Method loginWithAuthID(User Identifier, AuthID PIN)
- MAS AA Module → AuthID SDK: Call GetAccount(User Identifier) [AuthID Mobile SDK module]
- AuthID SDK → iOS/Android/LocalDB Storage: Lookup AuthID-PKI
- iOS/Android/LocalDB Storage → AuthID SDK: AuthID-PKI found
- AuthID SDK → MAS AA Module: AuthID-PKI found
- MAS AA Module → MAS SDK: Calls end-point "/auth/authIDPKI" (GET Method) Method MAS.invokeUnprotected() to be implemented
- MAS SDK → MAG endpoint: Fetch Challenge to sign [Rest API]
- MAG endpoint → Strong Authentication: GetChallenge()
- Strong Authentication → MAG endpoint: AuthID Challenge
- MAG endpoint → MAS SDK: AuthID Challenge
- MAS SDK → MAS AA Module: AuthID Challenge
- MAS AA Module → AuthID SDK: Call SignWithAccount() [AuthID Mobile SDK module]
- AuthID SDK → MAS AA Module: Return Signed Challenge
- MAS AA Module → MAS SDK: Calls end point "/auth/authIDPKI") (POST Method) Method MAS.invokeUnprotected() to be implemented
- MAS SDK → MAG endpoint: Verify Signed AuthID Challenge
- MAG endpoint → Strong Authentication: Verify Signed AuthID Challenge
- Strong Authentication → MAG endpoint: Verified Challenge Success Return SA Native Token [Break loop]
- MAG endpoint → MAG OTK endpoint: Verify Native Token
- MAG OTK endpoint → MAG endpoint: Native Token Verification SUCCESS
- MAG endpoint: Generate JWT
- MAG endpoint → MAS SDK: JWT
- MAS SDK → MAS AA Module: JWT
- MAS AA Module → MAS SDK: Call MAS SDK method Mas.LoginWithIDToken()
- MAS SDK → MAG endpoint: Login with ID Token
- MAG endpoint → MAS SDK: MAS SDK method SUCCESS [id token, access token, refresh token]
- MAS SDK → KeyChain: id_token
- MAS SDK → KeyChain: access token, refresh token
- MAS SDK → MAS AA Module: LoginWithIDToken() SUCCESS
- MAS AA Module → App: SDK return loginWithAuthID() SUCCESS