Understanding Java Cryptography

Frank P Moley III

Goals

* Learn Crypto Basics
* Learn how JCA and Providers work
* Learn how to consume JCA
* Discuss Best Practices
* WE WILL NOT LEARN CRYPTOGRAPHY

Agenda

* Crypto basics
* JCA & JCE + provider model
* Encryption
* Message Digests
* Digital Signatures
* Bouncy Castle

About Me

* Software Developer & Architect
* Security, Privacy, Modern Architectures, Distributed Systems, Data
* LinkedIn Learning Content Author
* User Group Co-Founder
* Constant Student

The Crypto Developers Club

What is Cryptography

* Study of secrets
* Study of methods for securing communications
* Secret in plain site
* Intense mathematical algorithms & practices
* Practical applications include data in motion and rest

Rules of Crypto Dev Club

* Never roll your own crypto
* Never merge without peer review
* Never use a compromised algorithm
* Never ignore compiler warnings
* Never reuse keys
* Never take shortcuts

Glossary

* Cipher text - data that has been hidden in plain sight
* Key - the value that determines the output of a crypto
* Plain text - the data that will be hidden or was once hidden
* Salt - value used to enhance the randomness of one-way functions
* Initialization Vector - data used to initialize a block cipher routine

Base Crypto Functions

* Digital Signature - mathematical function to verify authenticity and integrity and provide non-repudiation
* Encryption - two way mathematical function to mask data such that only the key holders can view it in the clear
* Hashing - one way mathematical function to map variable sized data into a finite sized mostly unique value

JCA & JCE

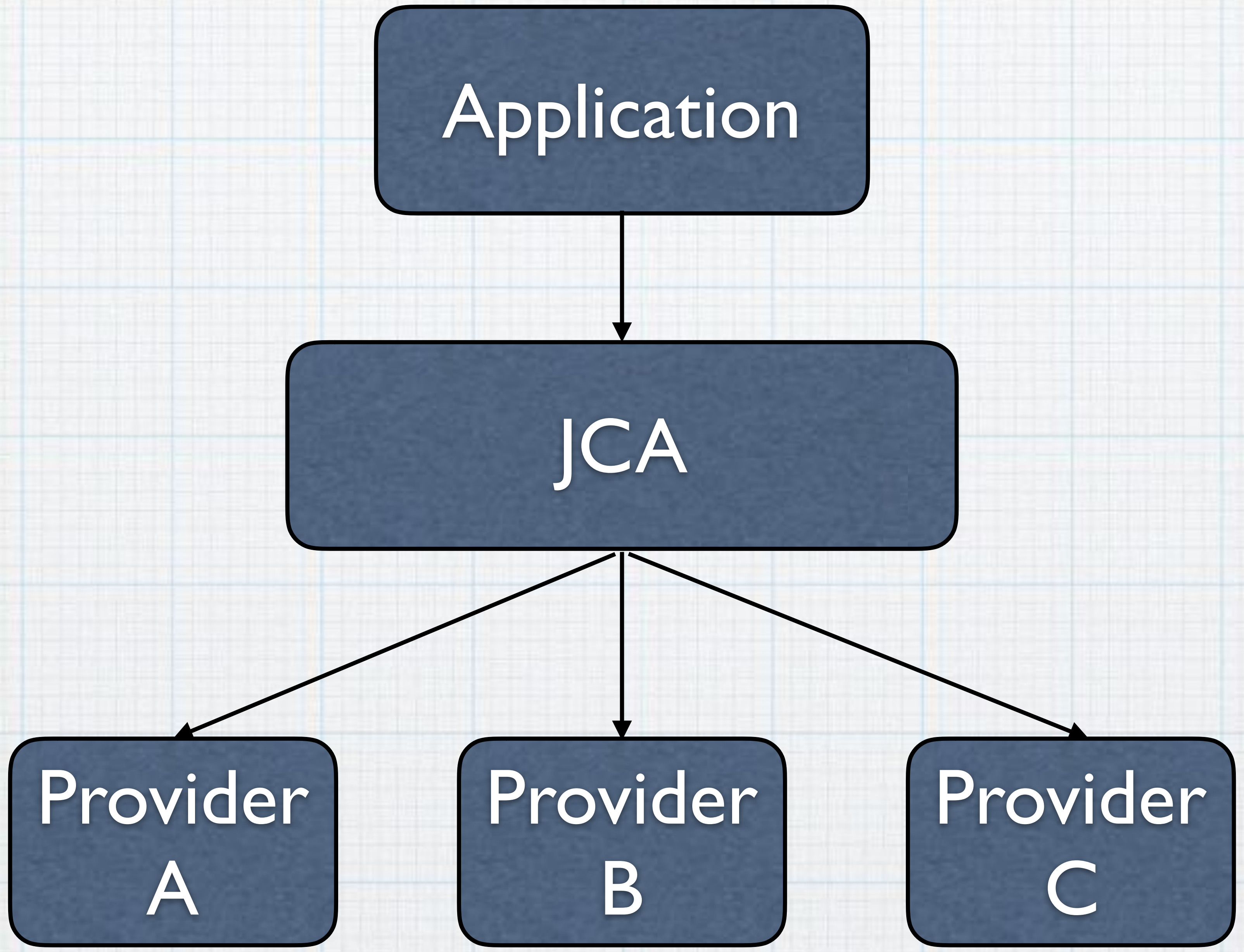
Java Cryptography Architecture

* It is the API for all cryptographic operations in Java
* Designed to allow JRE vendors to provide cryptographic implementations through a provider network
* Several standardized engines for cryptographic operations and several associated objects

Typical Provider Structure

* JCA provides the API
* Various implementations, or providers, provide algorithms
* Providers often vary in the algorithms they support
* Providers vary by JRE vendor as well as third party

Application Interaction



Secure Random

* Engine to create cryptographically strong random numbers
* Different than java.lang.Random
* Requirement for good crypto operations
* Seeds for keys and other algorithms needs

Cipher

* Engine to provide encryption
* Asymmetric and Symmetric encryption support
* Stream and Block ciphers
* Padding on block ciphers

Message Digest

* Engine to produce cryptographically secure hashes
* One way operations
* Fix sized output from variable sized input

Signature

* Engine to create and validate digital signatures
* Combination of hash and PKI asymmetric encryption
* Very useful in identity cases

Java Cryptography Extension

* JCE is the default Java provider implementation
* Come out of the box with JDK
* Provides implementation for every engine
* Doesn't support every algorithm or variation

Unlimited Strength?

* Prior to Java 8u151 JCE unlimited strength was a separate download
* Security.setProperty("crypto.policy", "unlimited");
* Java 8u162 enabled by default
* Java 9 enabled by default

Encryption

What is Encryption

* Process by which plain text data becomes hidden
* Ciphered data becomes difficult, hopefully impossible, to read without the key
* Only authorized parties, therefore, can decrypt

History

* Existed in posterity, maybe as far back as Ancient Egypt
* Middle ages and Renaissance - Arab and European Mathematicians
* WWII - Enigma machine
* Modern Times

Goals

* Confidentiality - aka privacy
* Integrity - not modified by bad actor
* Authentication - you are who you say you are
* Nonrepudiation - prove a message came from you believe sent it

Stream vs Block

* Steam ciphers take each bit at a time and encrypt or decrypt them
* Can be manipulated without cracking
* Block ciphers work on blocks of bits instead of individuals

Block Cipher Modes

* Cipher Block Chaining (CBC) combines previous block cipher with current block plain text
* Uses Initialization Vector
* Cipher Feedback (CFB), Output Feedback (OFB), and Electronic Code Book (ECB) less common

Symmetric vs Asymmetric

* Symmetric Encryption - shared key
* Asymmetric Encryption - public/private key pair
* Shared key block and stream ciphers are relatively quick
* PKI is extremely slow
* Usually PKI is used solely to transfer a shared key… TLS

Typical Uses

Symmetric

* Data at Rest
* Data in motion after initial handshake
* Steam and Block in use, however most Streams are out of scope (i.e. RC4 is cracked)

Asymmetric

* TLS
* Digital Signatures
* Certificate Trusts

Key Size

* Key size within an algorithm increases security
* Key size varies by algorithm, especially in Asymmetric examples
* Sufficient Key Size is critical

Codes vs Ciphers

* Important to note the differences
* Codes are replacements or substitutions
* Ciphers are mathematical encryptions

Encryption Examples in Code

Message Digest

Cryptographic Hash

* Hash function that takes arbitrary input and produces fixed sized output
* Easy to calculate
* Difficult to find original input
* Difficult to duplicate with unique inputs

AKA

* Digital Fingerprint
* Digest
* Message Digest
* Checksum

“Cracking” a Hash

* Finding an algorithm to generate a collision between two hashes
* Finding a algorithm to identify a unique and different input that will yield a given hash
* MD5, SHA-1 considered “cracked”

Hash Uses

* Digital Signatures
* Digital Fingerprints
* Logging sensitive data
* Saving Passwords

Message Digest examples in code

Passwords

Baseline

* Never store encrypted passwords
* Never use broken hash algorithms
* Never store in plain text

Good Password Hashing Algorithms

* Not cracked
* Not susceptible to brute force
* Inherent slowness

Broken

* MD5 - Cracked and fast
* SHA1 - Cracked and fast
* SHA2 - fast
* SHA3 - not a standard, fast

Possibilities

* PBKDF2
* bCrypt
* sCrypt

A Note on Entropy

* Humans struggle with random passwords
* Complexity comes from entropy
* Entropy is more impacted by length than characters
* Longer phrases are easier for humans to remember, no more patterns

BCrypt in code

Digital Signatures

What is a Digital Signature

* Asymmetrically encrypted hash of a digital payload
* A value that can provide guarantee of authenticity
* A value that can provide a mechanism of non-repudiation
* A value that can provide a guarantee of integrity

How is a Digital Signature Calculated

* Asymmetric KeyPair created
* Hash created
* Hash encrypted with private key
* Verification includes rehashing and matching to decryption

Digital Signature Flow

* Alice calculates hash of message
* Alice encrypts using private key and sends signature and message to Bob
* Bob hashes message
* Bob decrypts signature with Alice's Public Key and compares

Real Life Uses

* Binary program signing for authenticity and integrity
* Message signing for non-repudiation, integrity, and authenticity
* Watermarking

Digital Signature in Code