STA GROUP

Presents…

Securing your JavaEE Microservice with HMAC

Scott Kramer

A little about Presenter…

- Currently
  - STA Architect and "hands on" coder
  - Our biz is Business Transformation
- Also…
  - Tinker-er
  - JavaOne Speaker 2014 & 2015
  - Volunteer(ed) for iJUG, cJUG, Chicago Coder Conference, Police, etc…
  - Married to woman of my dreams
  - Certifications: SCJP & wrote several CBT's, coding 25 years+, and published stuff…

Scott Kramer
Hands-On Architect
STA Group Inc.

Agenda

- Micro-services
  - Concept and value proposition
  - Monolithic architecture
  - Micro-service architecture
  - Security and data architecture
- Securing Micro-services
  - Cryptology Concepts
  - HMAC
  - When, Why, and How to use
- Technical implementation
  - Examples

Source: http://www.phaidon.com/

STA GROUP

Microservices

Microservices evolution to a family's evolution…

Payment Processing

Map<Object, Object>
process (Map<Object, Object>)

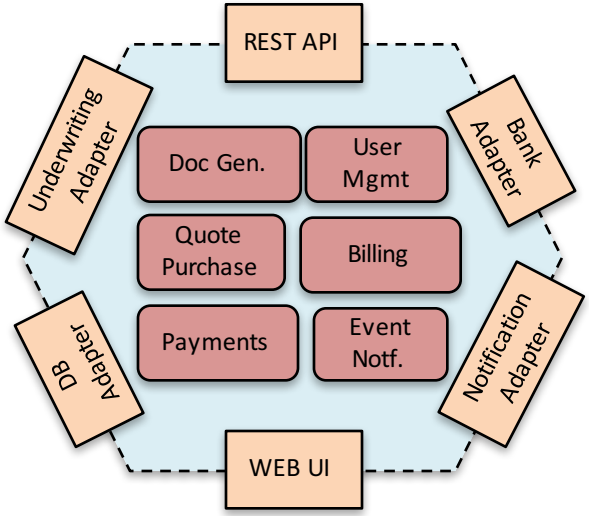
List<Payment>
processPayments (List<Integer>)

List<StudentPayment>
processStudentLoan
(List<Integer>)

Micro-services concepts and value proposition

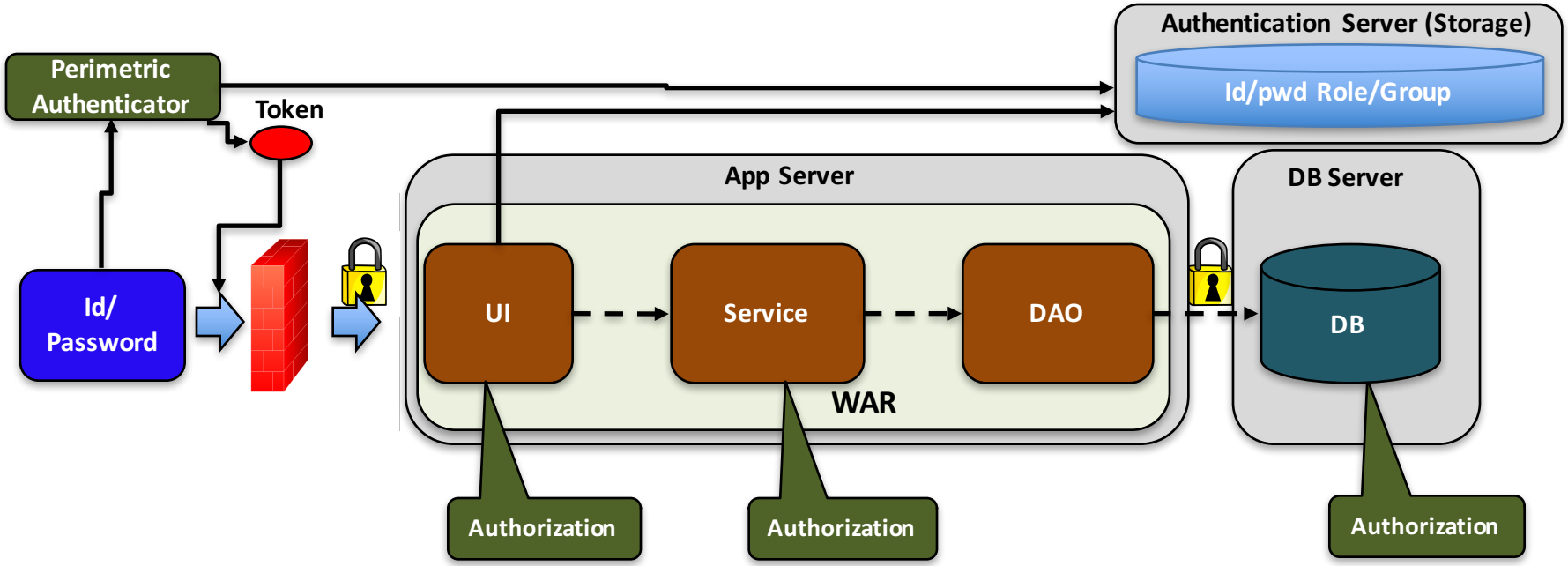
- Service with single business capability
- Scalable
- Separate process and self-contained
- Share nothing model
- Independently deployed
- Atomic and idempotent
- Performant
- Usually REST based
- Smart endpoint using dumb messaging

Monolithic architecture – High Level

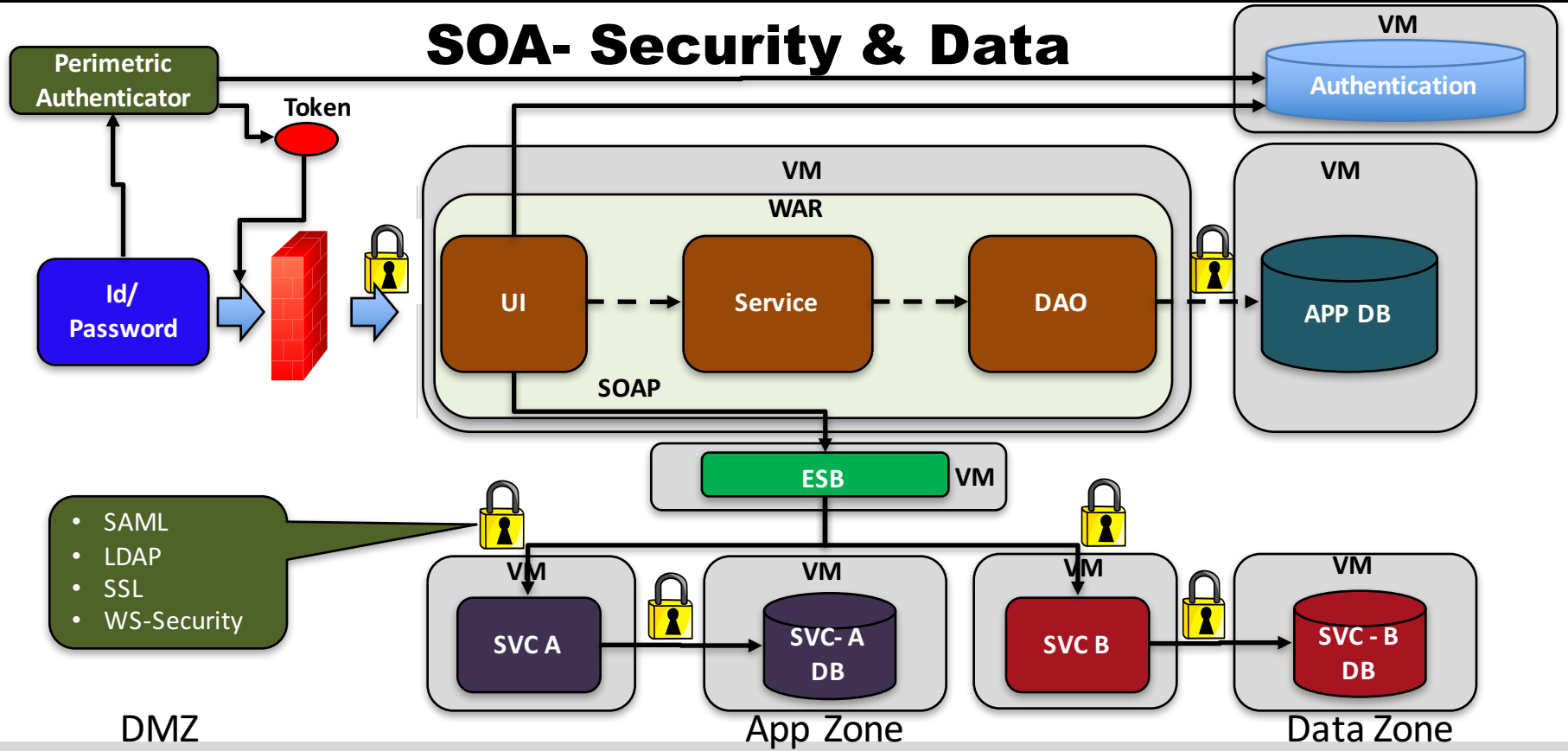


- Encompasses all functionality
- Modular architecture
- Supports multiple interfaces
- All dependencies are well defined and contained
- Security
- Data Architecture

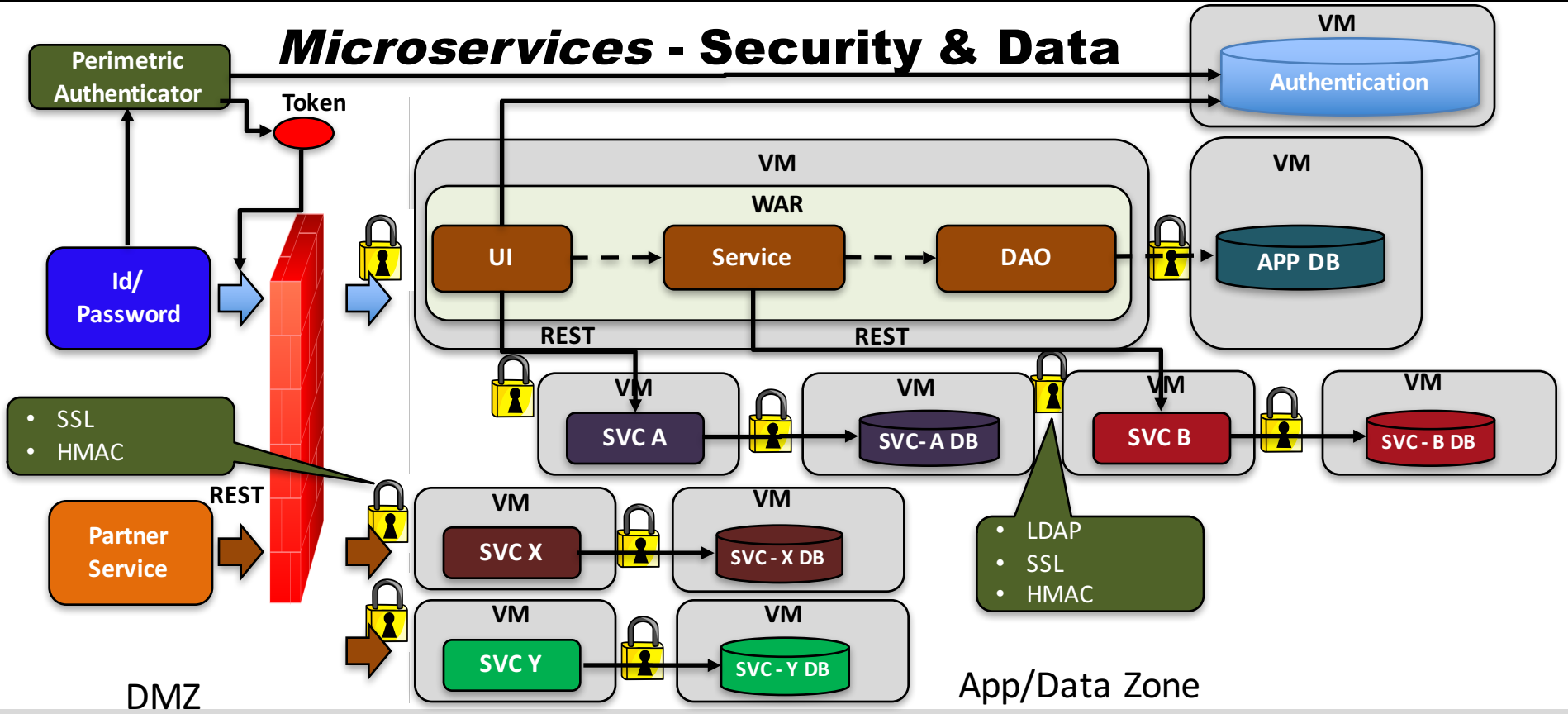
Monolithic Architecture - Security & Data



SOA- Security & Data



Microservices - Security & Data



STA GROUP

Cryptology Concepts

Basic Cryptology Concepts, Techniques, and Algorithms

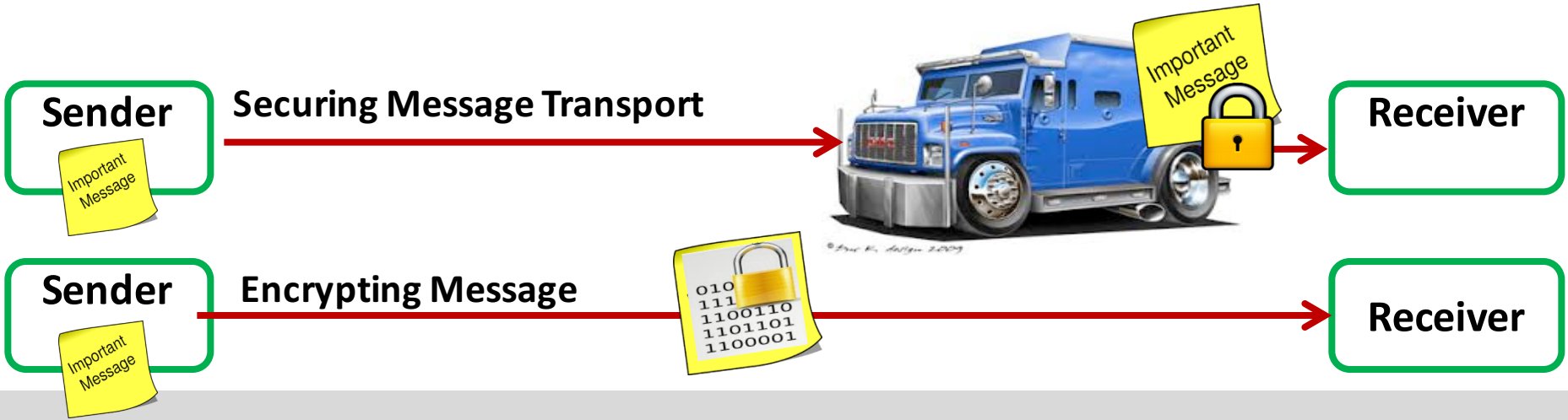
Locks n' Keys

- Locking with a combination number ("key") is susceptible to someone the guessing answer.
  - Automated computer guessing is called "brute force".
  - Cryptology locking = encryption, unlocking = decryption.

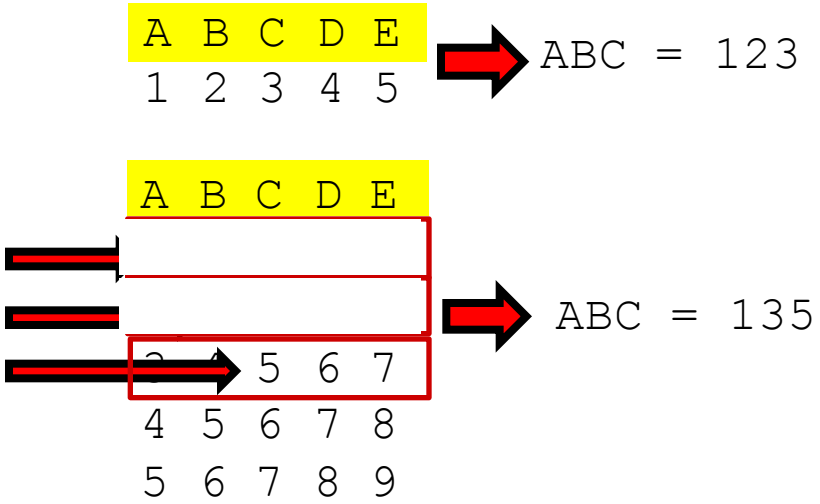
- Quick - What is the strongest lock?
  - Computer locks rely on crazy difficult math problems.
  - Todays strong lock is tomorrows brute force victim
  - Longer keys = more processing time to encode/decode

Ongoing Security Challenge…

- Message authentication and encryption is simple, fast and stateless.
- Easy to change access control based on shared secret keys.
- Security is not dependent on the application communication topology.

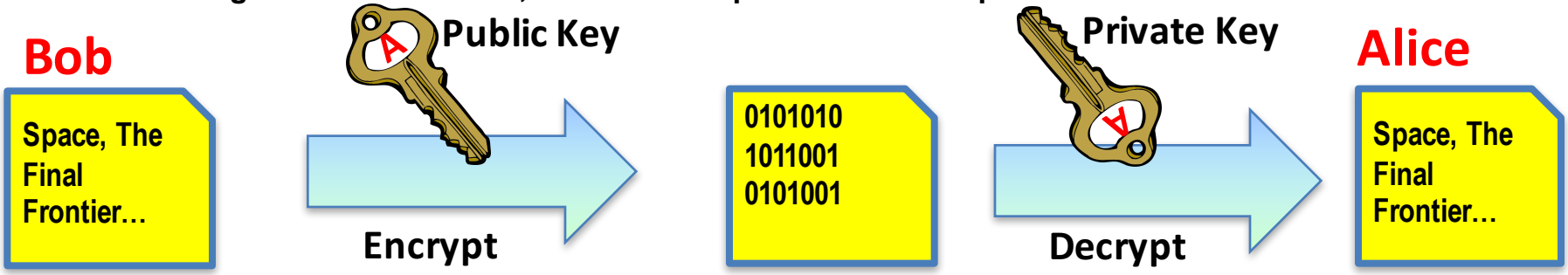


Basic Cryptology Concepts



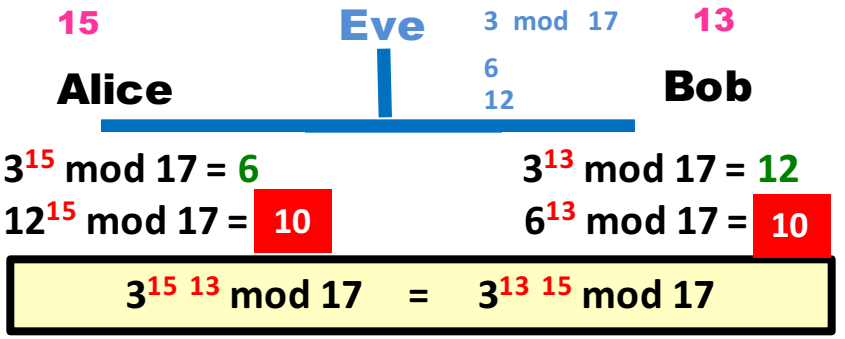
Solution: How two people securely exchange a message ?

- Clifford Cox solved this via complex mathematics
  - RSA, basis for Public key and Private key and more….
  - Multiplying two numbers is easy, determining the original prime numbers from result is hard.
  - ***Factoring*** large integers that are product of multiplying two ***large*** prime numbers is complex.
  - Can be slow; needs setup (Software Install; Key Locations; Certificate Authority, etc.)
  - When to use?
    - Certificate Authority setup;
    - Configuration not a issue; Protocol independence not required



Solution:

2 People *(who never met)*
create a secret key *publicly*.



- Diffie-Hellman used “Discreet Logarithm” math to solve.

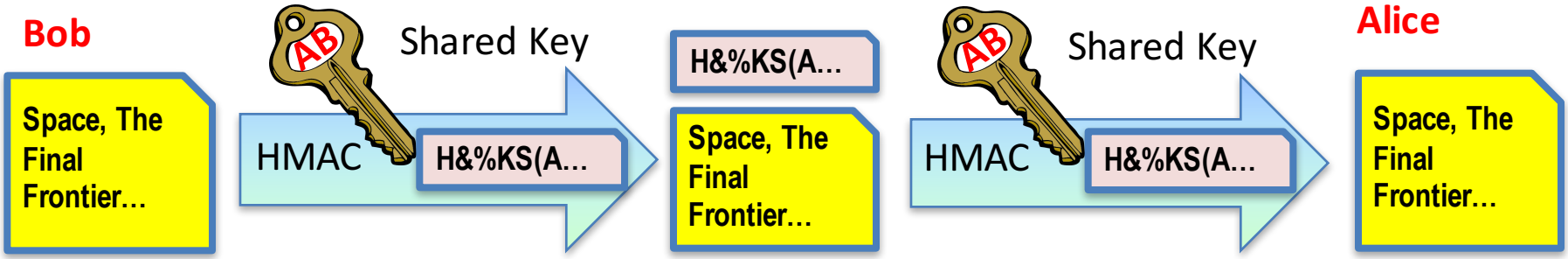
  <primitive-root>$^{X * Y}$ mod <base-number> ≡ <result>

- One-way encoding is fast even with large numbers
- Reversing the encoding without keys is very, very difficult

Designing a better encryption technique…

- Faster than Public/Private; protection from known attacks; simple to implement.
- Hashed Method Authentication Code (**RFC-2104**)– basic idea is to concatenate key and message then hash them together.

  HMAC = Hash(key // Hash(*key* // *message*)).
- Unlike SSL it can be transmitted via any protocol with headers, is machine/router independent, and stateless.



Cryptology Algorithm Summary…

- **RSA** – Asymmetric algorithm; Public/Private Key. Basis for SSL, SSH, PGP, TLS.
- **DIFFIE-HELLMAN –** Public secure key exchange between 2 parties who have never met.
- **HMAC –** Hashed Message Authentication Code; Very fast symmetric key algorithm *(10 to 100000x).*
- **AES** - Advanced Encryption Standard, fast symmetric key algorithm supersedes DES is based on substitution permutation network.
- **SHA-2** - Secure Hash algorithm, replaced SHA-1(vulnerable) and considered a better choice than *MD5*. One way computation of hash from message.

HMAC: Making it better…

- While HMAC is a fast algorithm, the ability to send additional data with the HMAC allows for more benefits (Timestamps, Keys, UserID's, etc).
  - When decoding data, quickly discard inauthentic messages
  - Limits denial of service attacks
  - It reduces your "attack surface" *(areas where attack can occur.)*

- Encrypt-then-MAC is provably secure (similar to IPSEC).
- Only possible attack against HMACs is brute force to get the secret key….
- HMAC-MD5 does not suffer from weaknesses that have been found in MD5.

Key Problems *(pun intended)…*

- Distribution & Exchange?
  - Variety of solutions for distribution

    LDAP (MD5), Diffie-Hellman, SSL (PGP, AWS, Azure), etc.
- Management & Authentication?
  - Design and Implementation Decision

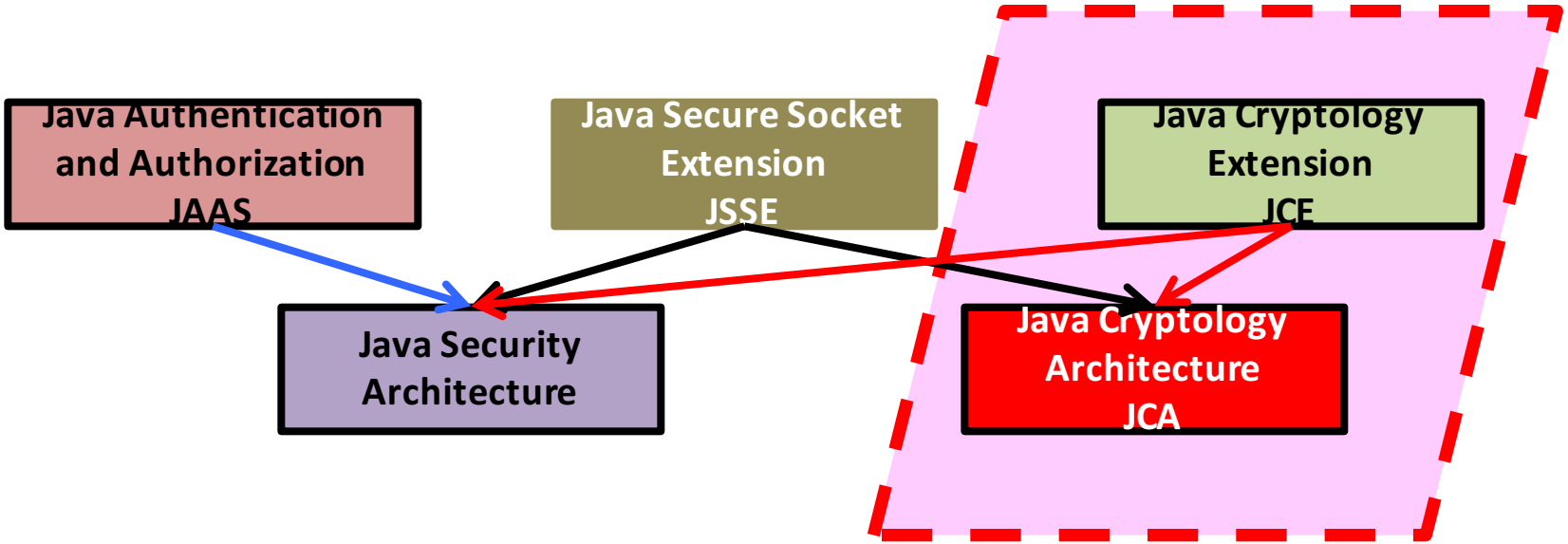
    LDAP; OWASP2; SAML; Certificate Authority, etc.
- Usage?
  - Based on design, its an implementation issue.
  - Factors: Length; Distribution; Replacement; Time Constraints, etc.

Code

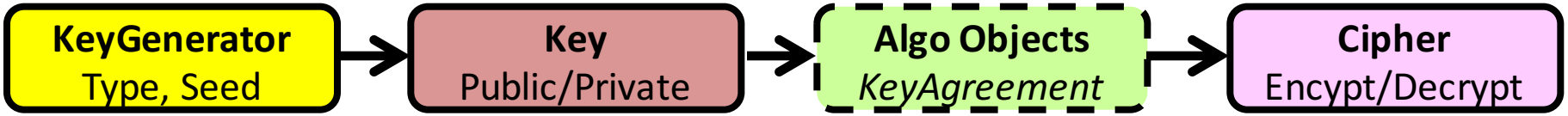
Quick Trip Down *"Java Crypto Lane"* & Some Code Examples

Java Security

- Java Security Architecture Overview



Java: Important Objects.



- **KeyGenerator** (KeyPairGenerator)– Generates Algo's Keys
  - .getInstance() ⇨ .initialize(Algo, Rnd) ⇨ .generate…()
- **Key** (KeyPair) – Top level interface for all keys
  - .getPrivate()/.getPrivate() – Gets Key based on Algorithm
- **Cipher** – provides encryption and decryption
  - .getInstance(algorithm) – Gets an instance of algorithm's cipher
  - .init(mode, key) – initialize cipher with mode and key info
  - .doFinal(*data*) – encrypt or decrypt data

STA GROUP

DH Code: Shared Key Calculation

```
BigInteger base = BigInteger.valueOf(2);
BigInteger modulous = new BigInteger(s, 16);
javax.crypto.spec.DHParameterSpec sharedModulusAndBase = new DHParameterSpec(modulous, base);

KeyPairGenerator aliceKeyPairGenerator = KeyPairGenerator.getInstance("DH");
aliceKeyPairGenerator.initialize(sharedModulusAndBase);
KeyPair aliceKeyPair = aliceKeyPairGenerator.generateKeyPair();
KeyAgreement aliceKeyAgreement = KeyAgreement.getInstance("DH");
DHPrivateKey alicePrivateKey = (DHPrivateKey) aliceKeyPair.getPrivate();
DHPublicKey alicePublicKey = (DHPublicKey) aliceKeyPair.getPublic();
aliceKeyAgreement.init(alicePrivateKey);
System.out.println("Alices Public Key: " + alicePublicKey.getY());

aliceKeyAgreement.doPhase(bobPublicKey, true);
SecretKey alicesDHSharedKey = aliceKeyAgreement.generateSecret("DES");
```

Mutual

Key Generator

Random Key

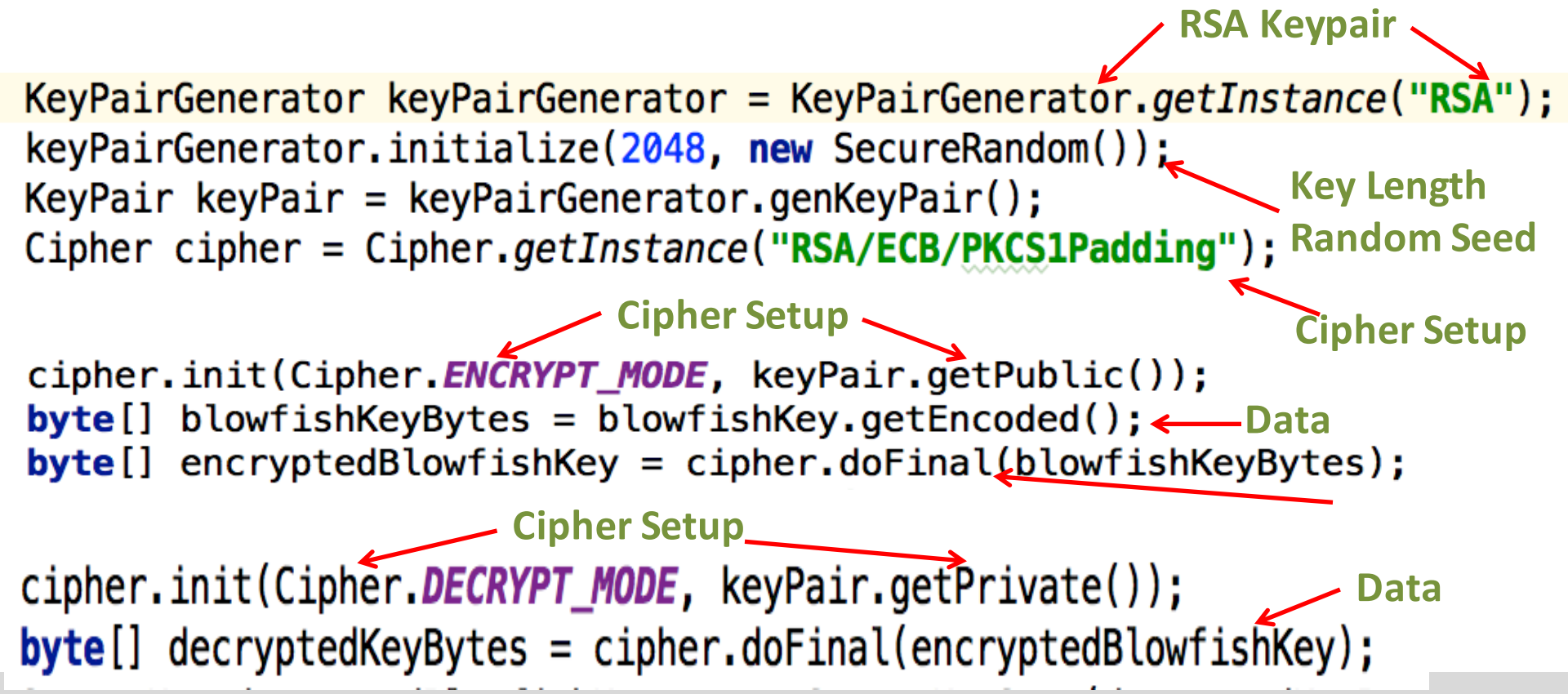
Key Pairs

DH Type

Result of Prime/Modulus using *Alices* Private Key

Using *Bob's* Init result of Secret/Prime/Modulus yields shared key.

RSA Code: Public Private Key



STA GROUP

HMAC Code: Creating a Hash

- The sender generates a HMAC code that is sent with the message, the receiver calculates the HMAC for the message to authenticate it against senders HMAC.

```
String secret = "secretKey";
String message = "Message of Space the final frontier...";

Mac sha256_HMAC = Mac.getInstance("HmacSHA256");
SecretKeySpec secret_key = new SecretKeySpec(secret.getBytes(), "HmacSHA256");
sha256_HMAC.init(secret_key);

String hash = Base64.encodeBase64String(sha256_HMAC.doFinal(message.getBytes()));
```

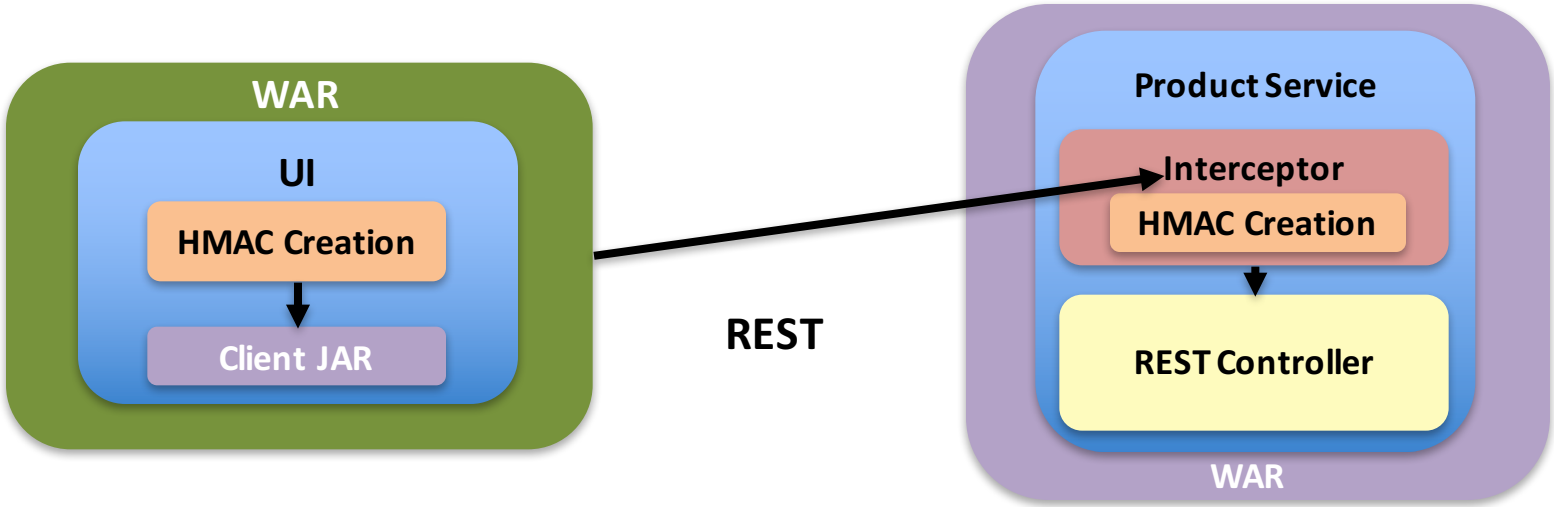
Create HMAC Object

Create a Key

Initialize HMAC with that key

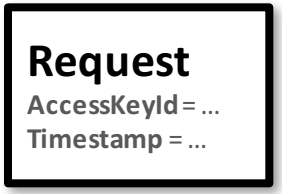
Create HMAC Code

Sample HMAC Implementation



Client

1 Create a request:



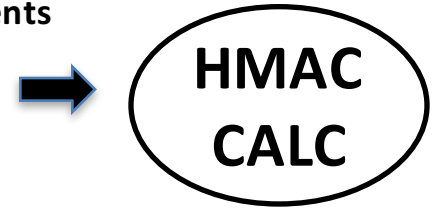
2 Create HMAC-SHA signature:

String based on request contents

+

Secret Access Key

wJalrXUtnFEMI/K7MDENG/
bPxRfiCYzEXAMPLEKEY

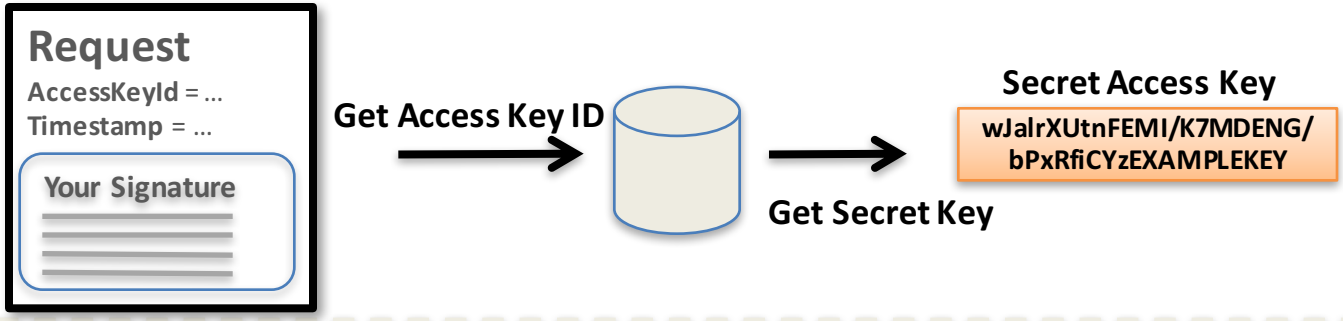


3 Send request and signature to Service:

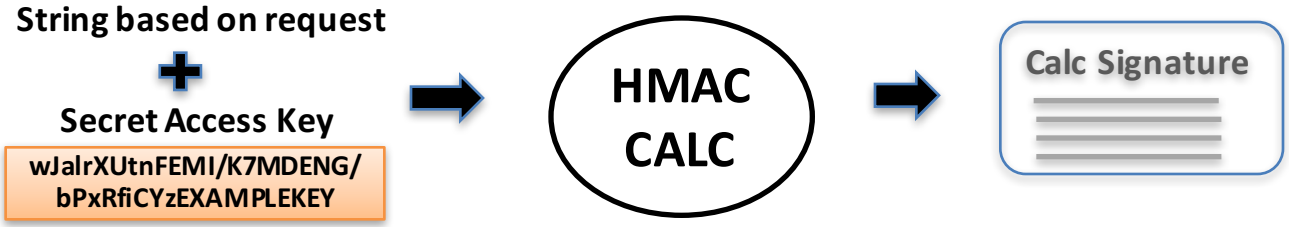
Service

Service

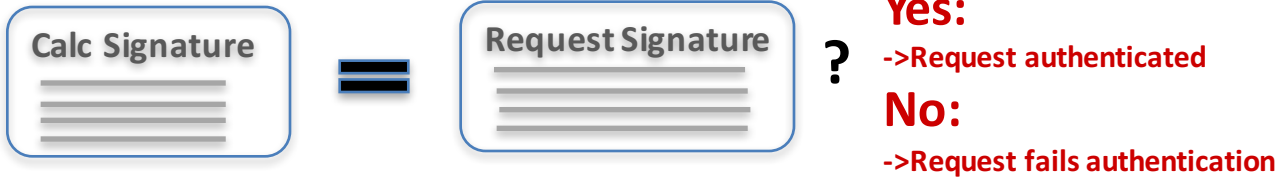
4 **Retrieve Secret Access Key:**



5 **Create HMAC-SHA signature:**



6 **Compare the two signatures:**



Client

STA GROUP

STRATEGIES FOR BUSINESS **TRANSFORMATION**

STA Group offers end-to-end business, technology and digital strategy for today's leading businesses. We deliver sustainable, reusable frameworks and processes that improve the alignment of corporate objectives, drive cost reduction and deliver compelling. customer experiences.

- Mobile – Develop wide range of mobile apps for Fortune 500
- Cloud – Expert cloud knowledge in host of technologies
- Big Data – Certified in implementing many Big Data technologies
- Security – Specialists in Application and Service Security
- Digital Strategy – Handles many companies digital brands and UX
- I.O.T .Technology – Significant IOT Developer for Fortune 500