Digital Signatures

Stephen Haunts

@stephenhaunts | www.stephenhaunts.com

Overview

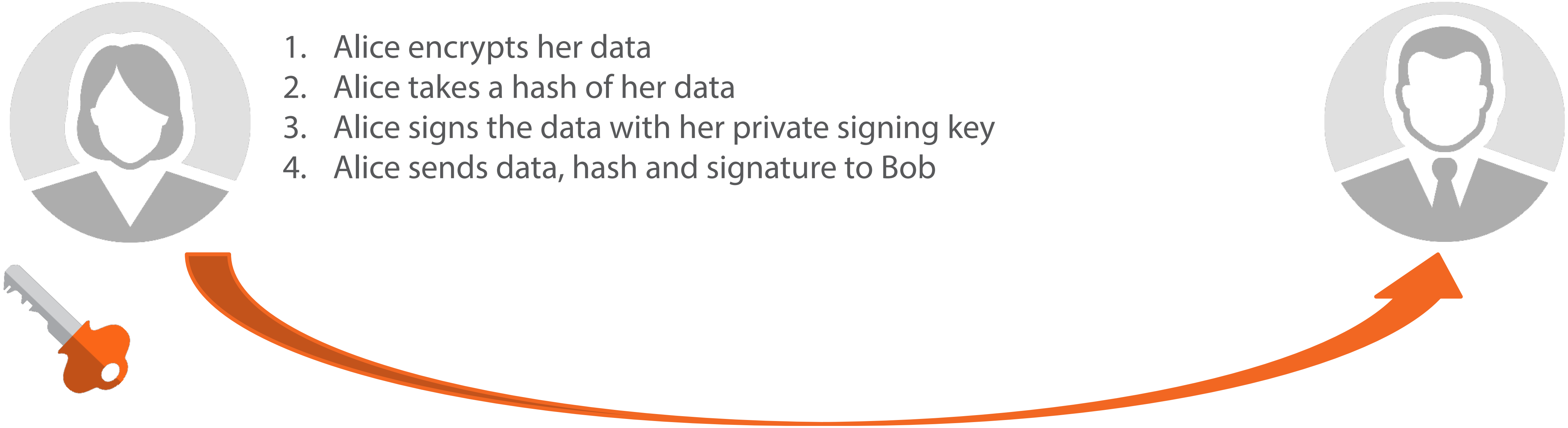
- What are digital signatures?
- Digital signatures in .NET

Digital Signatures

- Claiming authenticity of a message
- Digital signatures give both authentication and non-repudiation
- Based on asymmetric cryptography
- Digital signatures consist of
  - Public and private key generation
  - Signing algorithm using the private key
  - Verification algorithm using the public key

Digital Signatures

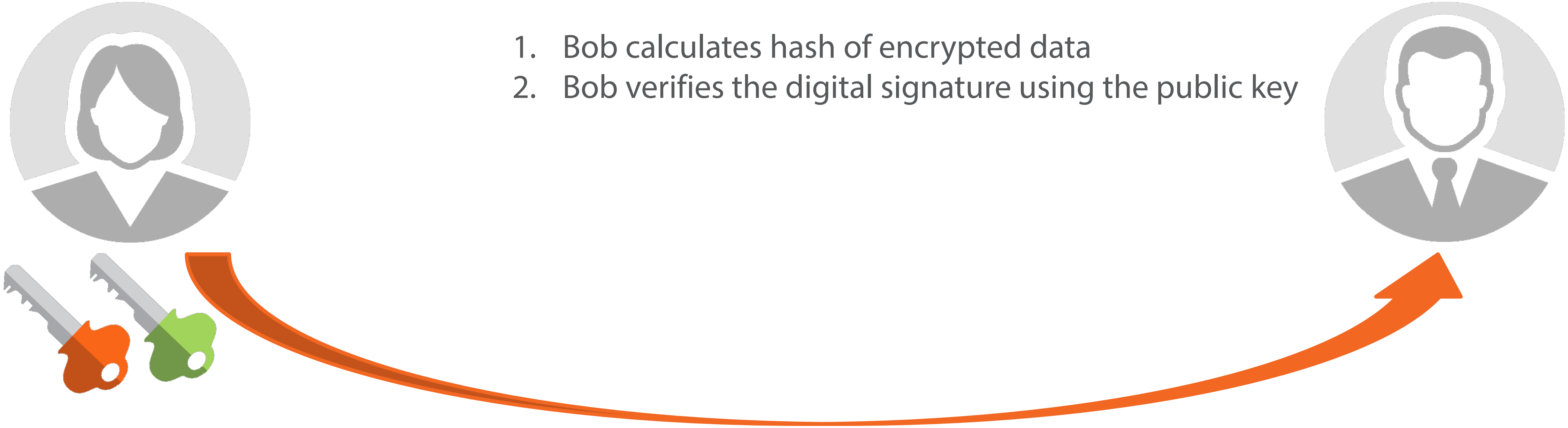
1. Alice encrypts her data
2. Alice takes a hash of her data
3. Alice signs the data with her private signing key
4. Alice sends data, hash and signature to Bob



Alice sends encrypted data, the hash and the digital signature to Bob

Digital Signatures

1. Bob calculates hash of encrypted data
2. Bob verifies the digital signature using the public key



Alice sends encrypted data, the hash and the digital signature to Bob

Digital Signatures

	Public Key	Private Key
Encryption (RSA)	Encrypt	Decrypt
Digital Signatures	Verify Signature	Sign Message

Digital Signatures in the .NET Framework

- Digital signatures use 3 main classes
  - RSACryptoServiceProvider
  - RSAPKCS1SignatureFormatter
  - RSAPKCS1SignatureDeformatter

Digital Signatures in the .NET Framework

```
public byte[] SignData(byte[] hashOfDataToSign)
{
    using (var rsa = new RSACryptoServiceProvider(2048)){
        rsa.PersistKeyInCsp = false;
        rsa.ImportParameters(_privateKey);

        var rsaFormatter = new RSAPKCS1SignatureFormatter(rsa);
        rsaFormatter.SetHashAlgorithm("SHA256");

        return rsaFormatter.CreateSignature(hashOfDataToSign);
    }
}
```

Digital Signatures in the .NET Framework

```
public byte[] SignData(byte[] hashOfDataToSign)
{
  using (var rsa = new RSACryptoServiceProvider(2048)){
    rsa.PersistKeyInCsp = false;
    rsa.ImportParameters(_privateKey);

    var rsaFormatter = new RSAPKCS1SignatureFormatter(rsa);
    rsaFormatter.SetHashAlgorithm("SHA256");

    return rsaFormatter.CreateSignature(hashOfDataToSign);
  }
}
```

Digital Signatures in the .NET Framework

```
public byte[] SignData(byte[] hashOfDataToSign)
{
    using (var rsa = new RSACryptoServiceProvider(2048)){
        rsa.PersistKeyInCsp = false;
        rsa.ImportParameters(_privateKey);

        var rsaFormatter = new RSAPKCS1SignatureFormatter(rsa);
        rsaFormatter.SetHashAlgorithm("SHA256");

        return rsaFormatter.CreateSignature(hashOfDataToSign);
    }
}
```

Digital Signatures in the .NET Framework

```
public byte[] SignData(byte[] hashOfDataToSign)
{
    using (var rsa = new RSACryptoServiceProvider(2048)){
        rsa.PersistKeyInCsp = false;
        rsa.ImportParameters(_privateKey);

        var rsaFormatter = new RSAPKCS1SignatureFormatter(rsa);
        rsaFormatter.SetHashAlgorithm("SHA256");

        return rsaFormatter.CreateSignature(hashOfDataToSign);
    }
}
```

Digital Signatures in the .NET Framework

```
public bool VerifySignature(byte[] hashOfDataToSign, byte[] signature)
{
    using (var rsa = new RSACryptoServiceProvider(2048)){
        rsa.ImportParameters(_publicKey);

        var rsaDeformatter = new RSAPKCS1SignatureDeformatter(rsa);
        rsaDeformatter.SetHashAlgorithm("SHA256");

        return rsaDeformatter.VerifySignature(hashOfDataToSign, signature);
    }
}
```

Digital Signatures in the .NET Framework

```
public bool VerifySignature(byte[] hashOfDataToSign, byte[] signature)
{
    using (var rsa = new RSACryptoServiceProvider(2048)){
        rsa.ImportParameters(_publicKey);

        var rsaDeformatter = new RSAPKCS1SignatureDeformatter(rsa);
        rsaDeformatter.SetHashAlgorithm("SHA256");

        return rsaDeformatter.VerifySignature(hashOfDataToSign, signature);
    }
}
```

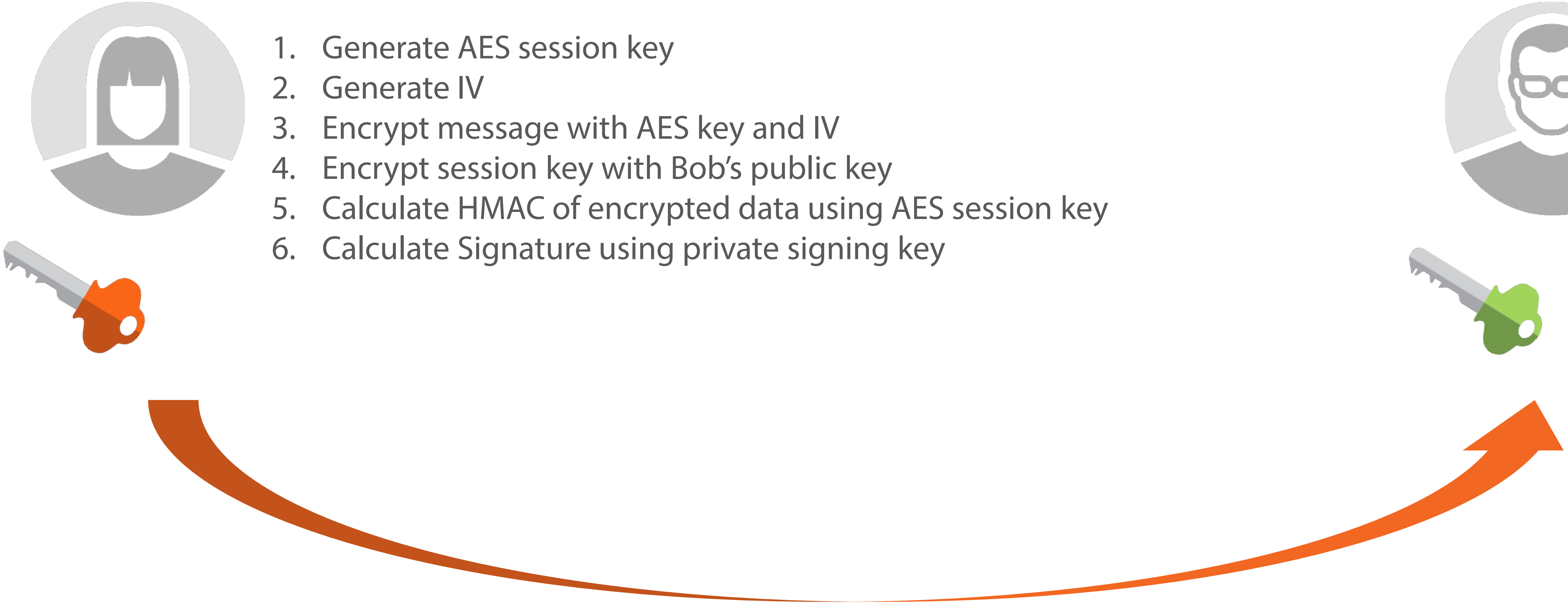
Digital Signatures in the .NET Framework

```
public bool VerifySignature(byte[] hashOfDataToSign, byte[] signature)
{
    using (var rsa = new RSACryptoServiceProvider(2048)){
        rsa.ImportParameters(_publicKey);

        var rsaDeformatter = new RSAPKCS1SignatureDeformatter(rsa);
        rsaDeformatter.SetHashAlgorithm("SHA256");

        return rsaDeformatter.VerifySignature(hashOfDataToSign, signature);
    }
}
```

Code Demonstration

Digital Signatures in the .NET Framework

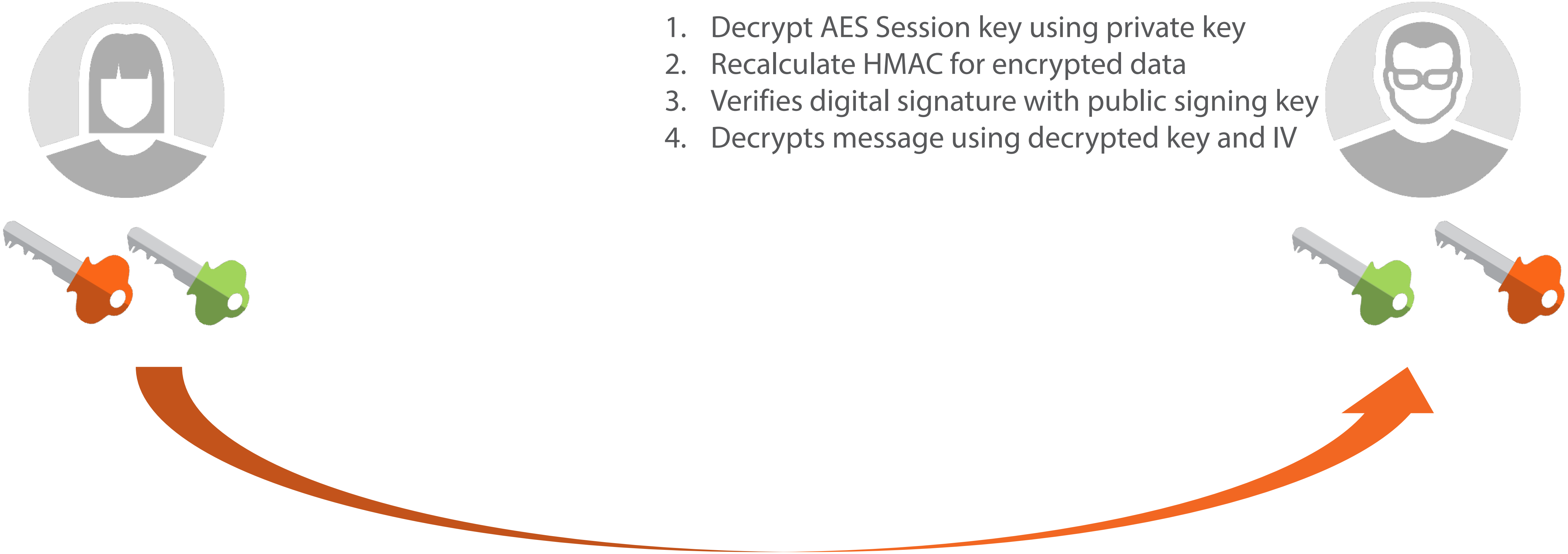
Extending the Hybrid Encryption Example



1. Generate AES session key
2. Generate IV
3. Encrypt message with AES key and IV
4. Encrypt session key with Bob’s public key
5. Calculate HMAC of encrypted data using AES session key
6. Calculate Signature using private signing key

Encrypted data, encrypted session key, IV, HMAC and signature are sent to Bob

Extending the Hybrid Encryption Example



1. Decrypt AES Session key using private key
2. Recalculate HMAC for encrypted data
3. Verifies digital signature with public signing key
4. Decrypts message using decrypted key and IV

Encrypted data, encrypted session key, IV, HMAC and signature are sent to Bob

Code Demonstration

Hybrid Encryption Including Digital Signatures

Module Summary

- Authenticity of a message
- Created by a known sender with no deniability
- Authentication and non-repudiation

Module Summary

- Uses the following classes in .NET
  - RSACryptoServiceProvider
  - RSAPKCS1SignatureFormatter
  - RSAPKCS1SignatureDeformatter

Module Summary

- Hybrid encryption example supports
  - Confidentiality
  - Integrity
  - Non-repudiation
  - Authentication