

CSE508 Network Security

2/29/2016 **Encrypted Communication** (Part 2)

Michalis Polychronakis

Stony Brook University

Public Key Cryptography

Many algorithms with different purposes

One common property: pair of values, one public and one secret

Encryption

Anyone can encrypt a message using a recipient's public key

Only a recipient can decrypt a message using their private key

No shared secret! Private key (secret) is stored only at one side

Digital signatures

Sign a message with a private key

Session key establishment

Exchange messages to create a shared secret key

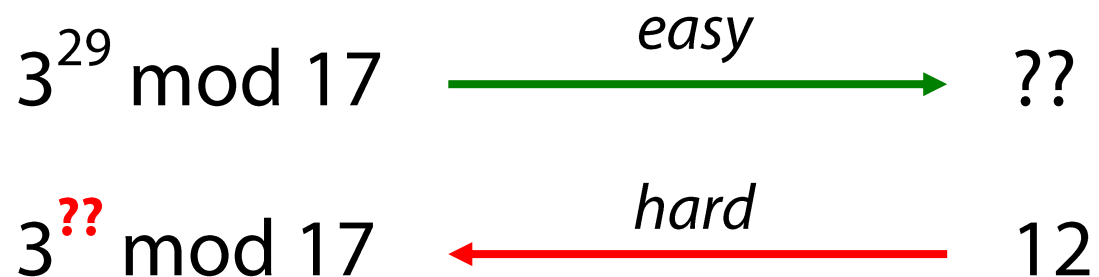
Diffie–Hellman Key Exchange

Allows two parties to jointly establish a shared secret key over an insecure communication channel

The established key can then be used to encrypt subsequent communication using a symmetric key cipher

“New Directions in Cryptography” by Whitfield Diffie and Martin Hellman, 1976

Based on the discrete logarithm problem



Diffie–Hellman Key Exchange

Alice and Bob agree on a large (at least 1024 bit) prime number p and a base g – *both public*

p is usually of the form $2q+1$ where q is also prime

g is a generator of the multiplicative group of integers modulo p
(for every x coprime to p there is a k such that $g^k \equiv x \pmod{p}$)

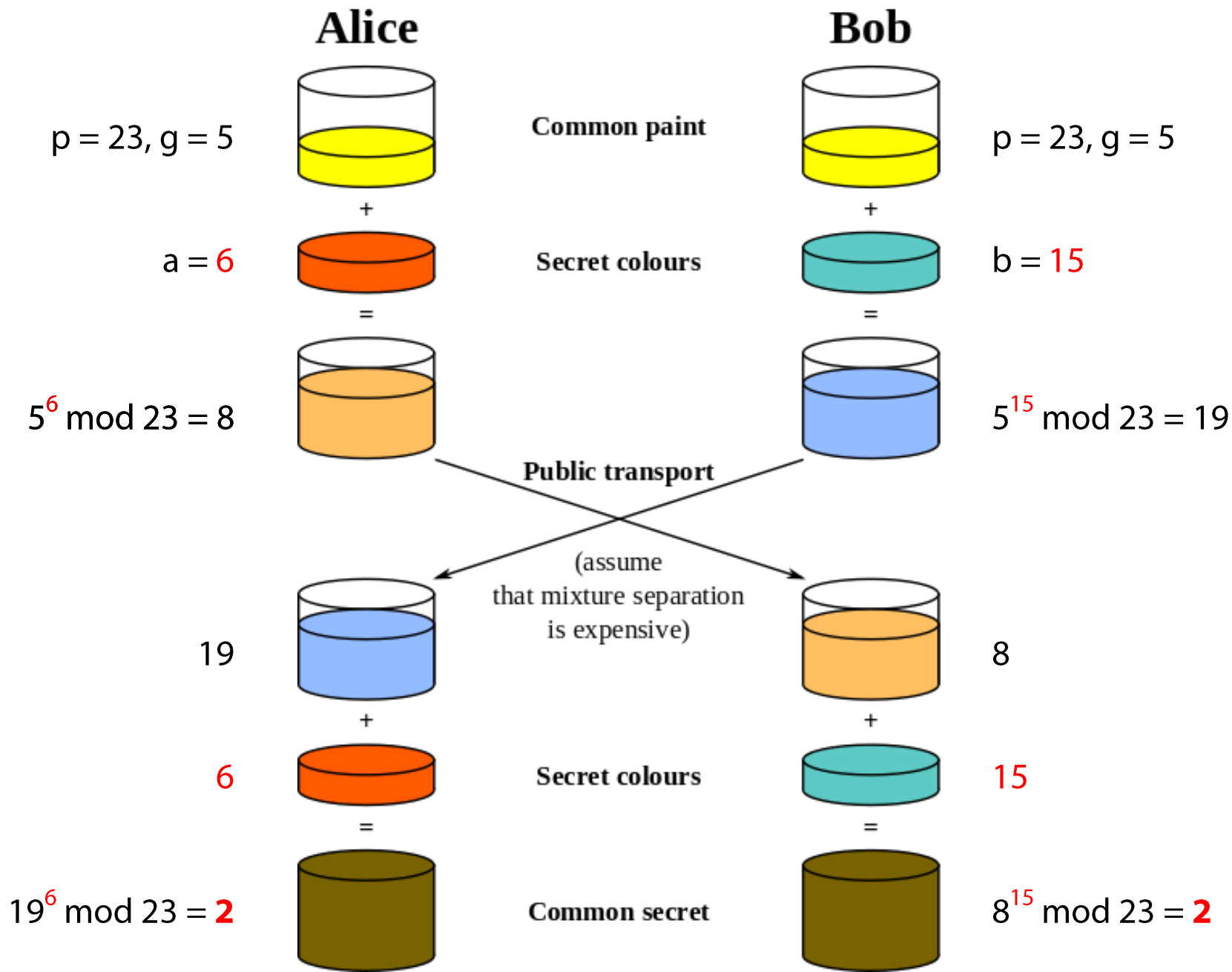
Alice picks a secret (private) large random number a and sends to Bob $g^a \pmod{p}$

Bob picks a secret large random number b and sends to Alice $g^b \pmod{p}$

Alice calculates $s = (g^b \pmod{p})^a = g^{ba} \pmod{p}$

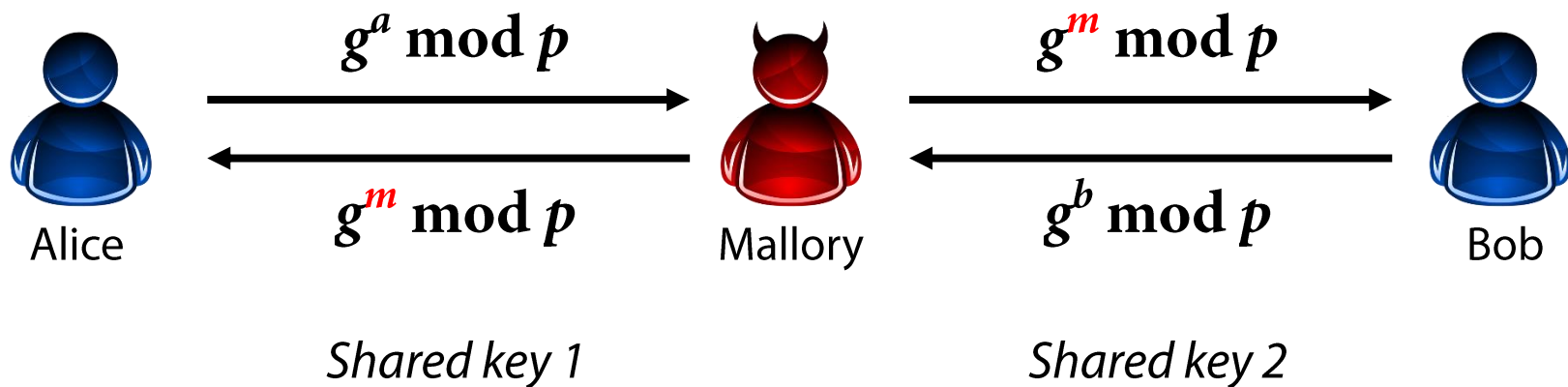
Bob calculates $s = (g^a \pmod{p})^b = g^{ab} \pmod{p}$

} *shared key*



Man-in-the-Middle Attack

Alice and Bob share no secrets

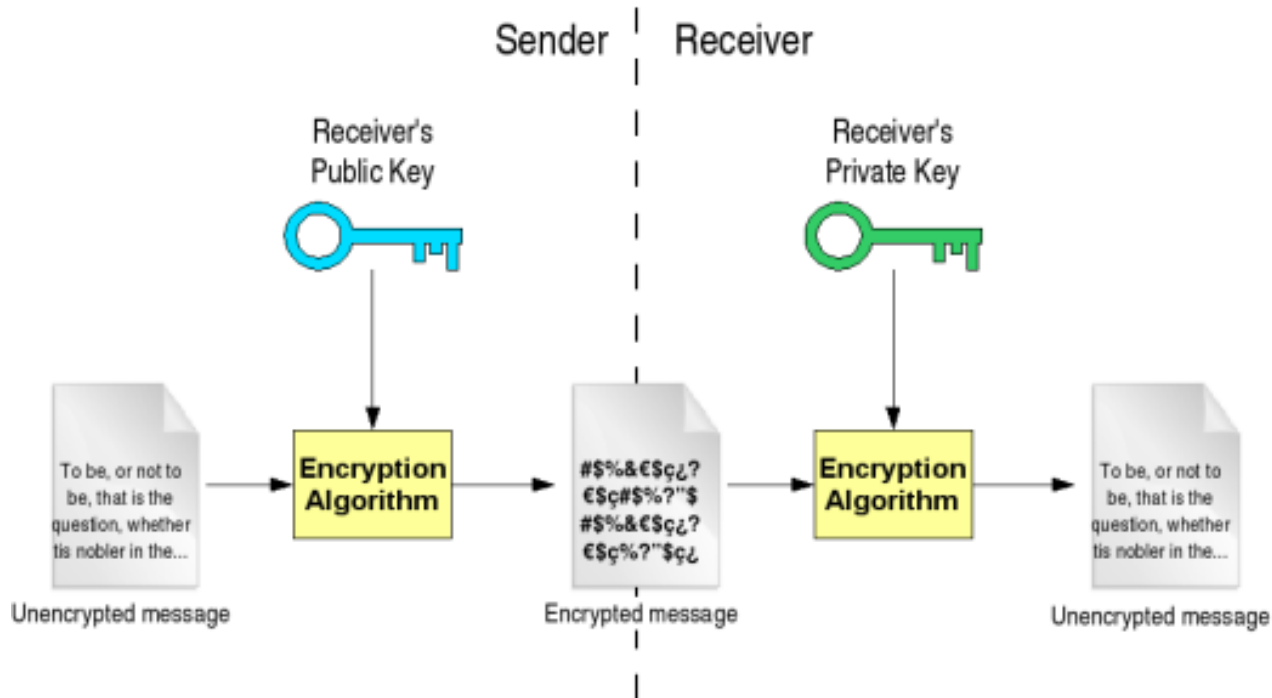


Mallory actively decrypts and re-encrypts all traffic

No authentication: Alice and Bob assume that they communicate directly

General problem: *need for a root of trust*

Public Key Cryptography



Pros:

- No shared secrets
- Easier key management
- Provides secrecy and authenticity

Cons:

- Slow
- Large keys
- Key generation is more difficult

RSA

Named after its inventors Rivest, Shamir, Adleman

Based on the assumption that factoring large numbers is hard

Variable key length

Largest (publicly known) factored RSA modulus is 768-bit long

That was in 2009: took 2 years and many hundreds of machines

It is believed that 1024-bit keys may already (or in the near future) be breakable by a sufficiently powerful attacker

2048-bit keys should be the absolute minimum

For now (next decade or two)...



Commercial National Security Algorithm Suite and Quantum Computing FAQ



Q: What is the Commercial National Security Algorithm Suite?

A: The Commercial National Security Algorithm Suite is the suite of algorithms identified in CNSS Advisory Memorandum 02-15 for protecting NSS up to and including TOP SECRET classification. This suite of algorithms will be incorporated in a new version of the National Information Assurance Policy on the Use of Public Standards for the Secure Sharing of Information Among National Security Systems (CNSSP-15 dated October 2012). The Advisory

Algorithm	Usage
RSA 3072-bit or larger	Key Establishment, Digital Signature
Diffie-Hellman (DH) 3072-bit or larger	Key Establishment
ECDH with NIST P-384	Key Establishment
ECDSA with NIST P-384	Digital Signature
SHA-384	Integrity
AES-256	Confidentiality

RSA

Choose two distinct large prime numbers p and q

Let $n = pq$ (modulus)

Select e as a relative prime to $(p - 1)(q - 1)$

Calculate d such that $de \equiv 1 \pmod{(p - 1)(q - 1)}$

Public key = (e, n)

Private key = (d, n)

To encrypt m , calculate $c \equiv m^e \pmod n$

Plaintext block must be smaller than the key length

To decrypt c , calculate $m \equiv c^d \pmod n$

Ciphertext block will be as long as the key

RSA in Practice

RSA calculations are computationally expensive

Use RSA in combination with a symmetric key

Send an encrypted message:

Encrypt message with a random symmetric key

Encrypt symmetric key with recipient's public key

Transmit both the encrypted message and the encrypted key

Set up an encrypted communication channel:

Negotiate a symmetric key using RSA

Use the symmetric key for subsequent communication

PKCS: Public-Key Cryptography Standards (#1–#15)

Make different implementations interoperable

Avoid various known pitfalls in commonly used schemes

Forward Secrecy

Threat: capture encrypted traffic now, use in the future

- Private keys may be compromised (e.g., infiltrate system)

- Cryptanalytic breakthrough

FS: Ensure that even if current keys are compromised, past encrypted traffic cannot be compromised

- Cannot read old messages

- Cannot forge a message and claim that it was sent in the past

Support

- IPsec, SSH, Off-the-Record messaging (OTR)

- TLS (Diffie–Hellman instead of RSA key exchange)

Note a panacea

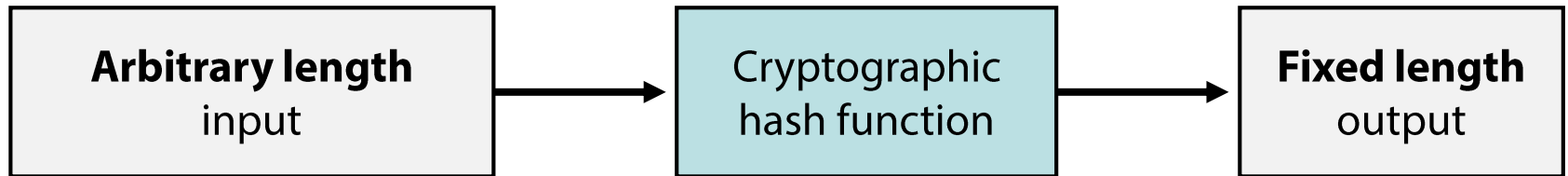
- Session keys might be kept in memory for hours

- Server could be forced to record all session keys

- TLS session resumption need careful treatment

Cryptographic Hash Functions

Hash functions that are considered practically impossible to invert



Properties of an ideal cryptographic hash function

Easy to compute the hash value for any given message

Infeasible to generate a message that has a given hash

Infeasible to modify a message without changing the hash

Infeasible to find two different messages with the same hash

Many-to-one function: *collisions can happen*

Cryptographic Hash Function Properties

Pre-image resistance

Given a hash value h it should be computationally infeasible to find any input m such that $h = \text{hash}(m)$

Example: break a hashed password

Second pre-image resistance

Given m_1 it should be computationally infeasible to find m_2 such that $m_1 \neq m_2$ and $\text{hash}(m_1) = \text{hash}(m_2)$

Example: forge an existing certificate

Collision Resistance

It should be computationally infeasible to find two different inputs m_1 and m_2 such that $\text{hash}(m_1) = \text{hash}(m_2)$ (collision)

Example: prepare two contradicting versions of a contract

Birthday Paradox

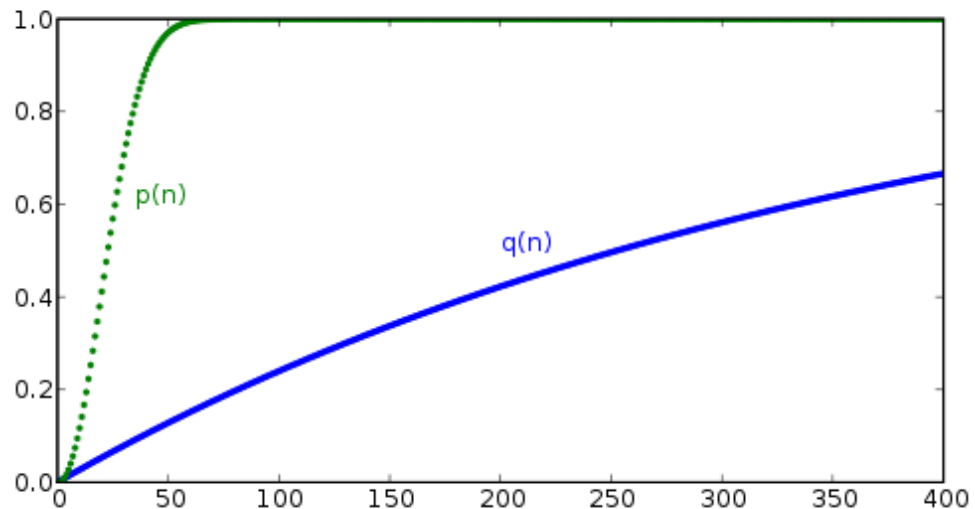
How many people does it take before the odds are 50% or better of having...

...another person with the same birthday as *you*? **253**

Second pre-image resistance

...two people with the same birthday? **23**

Collision resistance



Uses of Cryptographic Hash Functions

Data integrity

Digital signatures

Message authentication

User authentication

Timestamping

Certificate revocation management

Common Hash Functions

MD5: 128-bit output

1993: Boer and Bosselaers, “pseudo-collision” of the MD5 compression function: 2 different IVs which produce an identical digest

1996: Dobbertin, collision of the MD5 compression function

2004: Wang, Feng, Lai, and Yu, collisions for the full MD5

2005: Lenstra, Wang, and de Weger, construction of two X.509 certificates with different public keys but same hash

2008: Sotirov, Stevens, Appelbaum, Lenstra, Molnar, Osvik, de Wege, creating rogue CA certificates

Use it? NO, it's unsafe

SHA-1: 160-bit output

2005: Rijmen and Oswald, attack on a reduced version of SHA1 (53 out of 80 rounds)

2005: Wang, Yao, and Yao, an improvement, lowering the complexity for finding a collision to 2^{63}

2006: Rechberger, attack with 2^{35} compression function evaluations

2015: Stevens, Karpman, and Thomas, freestart collision attack

Use it? Use SHA-256 or better instead



[Sign in](#)

Security Research and Defense Blog

Search TechNet with Bing

 Search this blog Search all blogs[Home](#) [About](#) [View More Blogs](#)

TechNet Blogs » Security Research & Defense » Flame malware collision attack explained

Flame malware collision attack explained



swiat

6 Jun 2012 9:57 AM

0

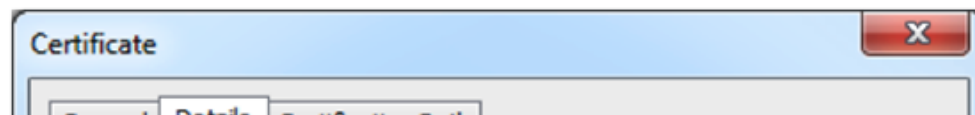
Since our last MSRC blog post, we've received questions on the nature of the cryptographic attack we saw in the complex, targeted malware known as Flame. This blog summarizes what our research revealed and why we made the decision to release [Security Advisory 2718704](#) on Sunday night PDT. In short, by default the attacker's certificate would not work on Windows Vista or more recent versions of Windows. They had to perform a collision attack to forge a certificate that would be valid for code signing on Windows Vista or more recent versions of Windows. On systems that pre-date Windows Vista, an attack is possible without an MD5 hash collision. This certificate and all certificates from the involved certificate authorities were invalidated in [Security Advisory 2718704](#). We continue to encourage all customers who are not installing updates automatically to do so immediately.

Mysterious Missing Extensions

When we first examined the Flame malware, we saw a file that had a valid digital signature that chained up to a Microsoft Root authority. As we reviewed this certificate, we noticed several irregularities. First, it had no X.509 extension fields, which was not consistent with the certificates we issued from the Terminal Server licensing infrastructure. We expected to find a Certificate Revocation List (CRL) Distribution Point (CDP) extension, an Authority Information Access (AIA) extension, and a "Microsoft Hydra" critical extension. All of these were absent.

Share Article

Follow Us

[RSS for Posts](#) [@msftsecresponse](#) [Security Newsletter](#)[Report a Vulnerability](#)

NOVEMBER 4, 2015 10:00 AM

SHA-1 Deprecation Update

By [Kyle Pflug](#) / Program Manager, Microsoft Edge

In a previous update on TechNet, we announced that [Windows will block SHA-1 signed TLS certificates starting on January 1, 2017](#). In light of recent advances in [attacks on the SHA-1 algorithm](#), we are now considering an accelerated timeline to deprecate SHA-1 signed TLS certificates as early as June 2016.

Mozilla recently [announced a similar intent on the Mozilla Security Blog](#). We will continue to coordinate with other browser vendors to evaluate the impact of this timeline based on telemetry and current projections for feasibility of SHA-1 collisions.

For more details on our schedule, please see [Windows Enforcement of Authenticode Code Signing and Timestamping](#) on Technet, or

RELATED POSTS

[HTTP Strict Transport Security comes to Internet Explorer 11 on Windows 8.1 and Windows 7](#)

[Read more](#)

[Ending support for the RC4 cipher in Microsoft Edge and Internet Explorer 11](#)

[Read more](#)

[How Microsoft Edge and Internet Explorer 11 on Windows 10 work better together in the Enterprise](#)

[Read more](#)

[Protecting Microsoft Edge against binary injection](#)

[Read more](#)

Message Authentication Codes (MACs)

Verify both message integrity and authenticity

Keyed-hash message authentication code (HMAC)

For a cryptographic hash function H :

$$\text{HMAC}(K, m) = H((K \oplus \text{opad}) \parallel H(K \oplus \text{ipad} \parallel m))$$

opad/ipad: outer/inner padding

\parallel denotes concatenation

Double nesting prevents length-extension attacks

Impossible to generate the HMAC of a message without knowing the secret key

Order of Encryption and MACing

Encrypted data usually must be protected with a MAC

Encryption alone protects only against passive adversaries

Different options:

Encrypt-and-MAC $E(P) || M(P)$

No integrity of the ciphertext

MAC-then-Encrypt $E(P || M(P))$

No integrity of the ciphertext (have to decrypt it first)

Encrypt-then-MAC $E(P) || M(E(P))$

Preferable option – *always MAC the ciphertext*

Digital Signatures

Use RSA backwards:

Sign (encrypt) with the private key

Verify (decrypt) with the public key

Ownership of a private key turns it into a *digital signature*

Anyone can verify that a message was signed by its owner

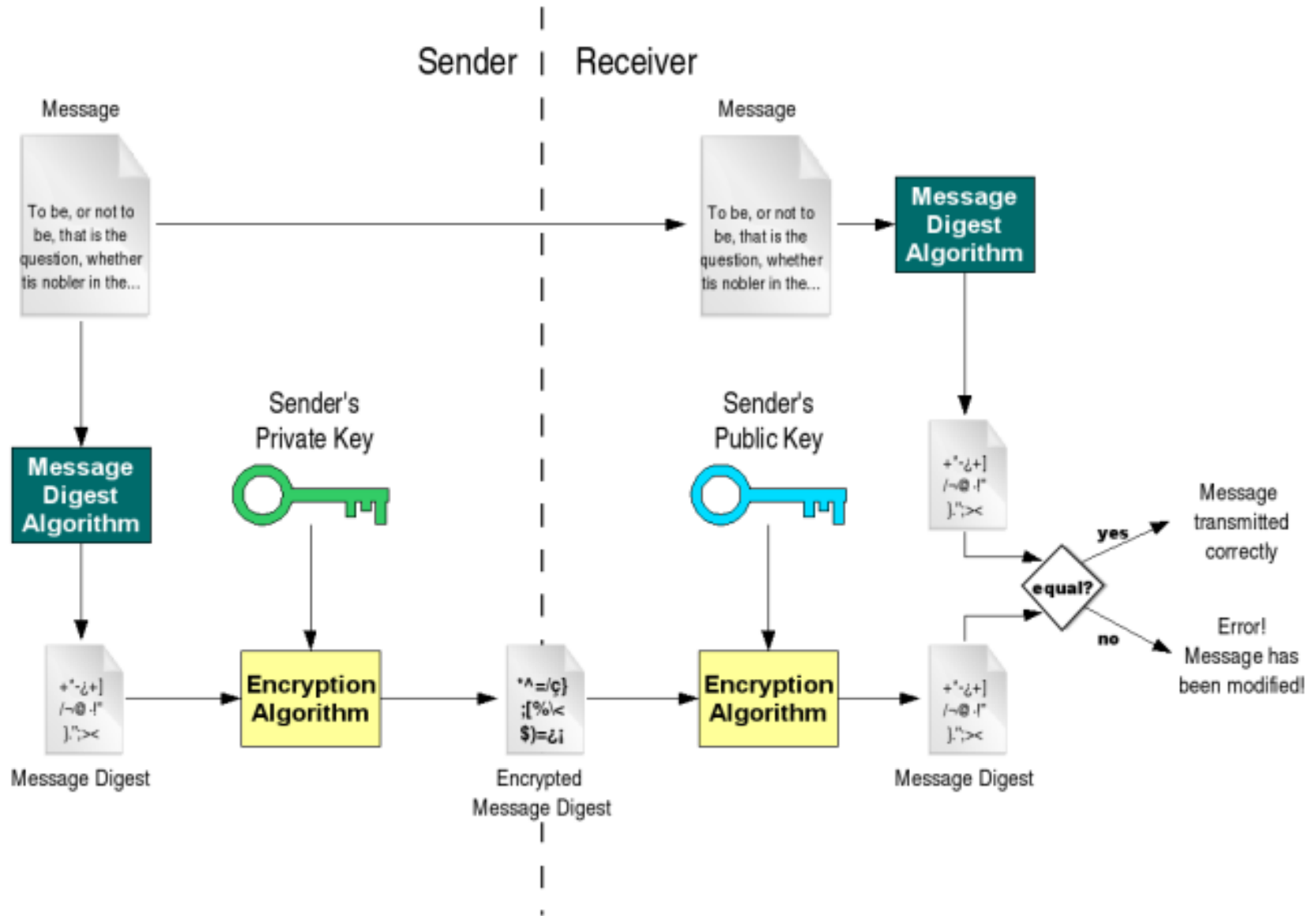
What if a private key was stolen or deliberately leaked?

Non-repudiation

Again, too expensive to sign the whole message

Calculate a cryptographic hash of the message and sign the hash

Digital Signatures



Hashes vs. MACs vs. Digital Signatures

	Hash	MAC	Signature
Integrity	✓	✓	✓
Authentication		✓	✓
Non-repudiation			✓
Keys	<i>None</i>	<i>Symmetric</i>	<i>Asymmetric</i>

Public Key Authenticity

Authentication without confidence in the keys used is pointless

Need to gain confidence or proof that a particular public key is authentic

- It is correct and belongs to the person or entity claimed
- Has not been tampered with or replaced by an attacker

Different ways to establish trust

TOFU: trust on first use (e.g., SSH)

Web of trust – decentralized (P2P) trust model (e.g., PGP)

PKI: public key infrastructure (e.g., SSL)

(subject of future lecture)

Adi Shamir: *Crypto is typically bypassed, not penetrated*

