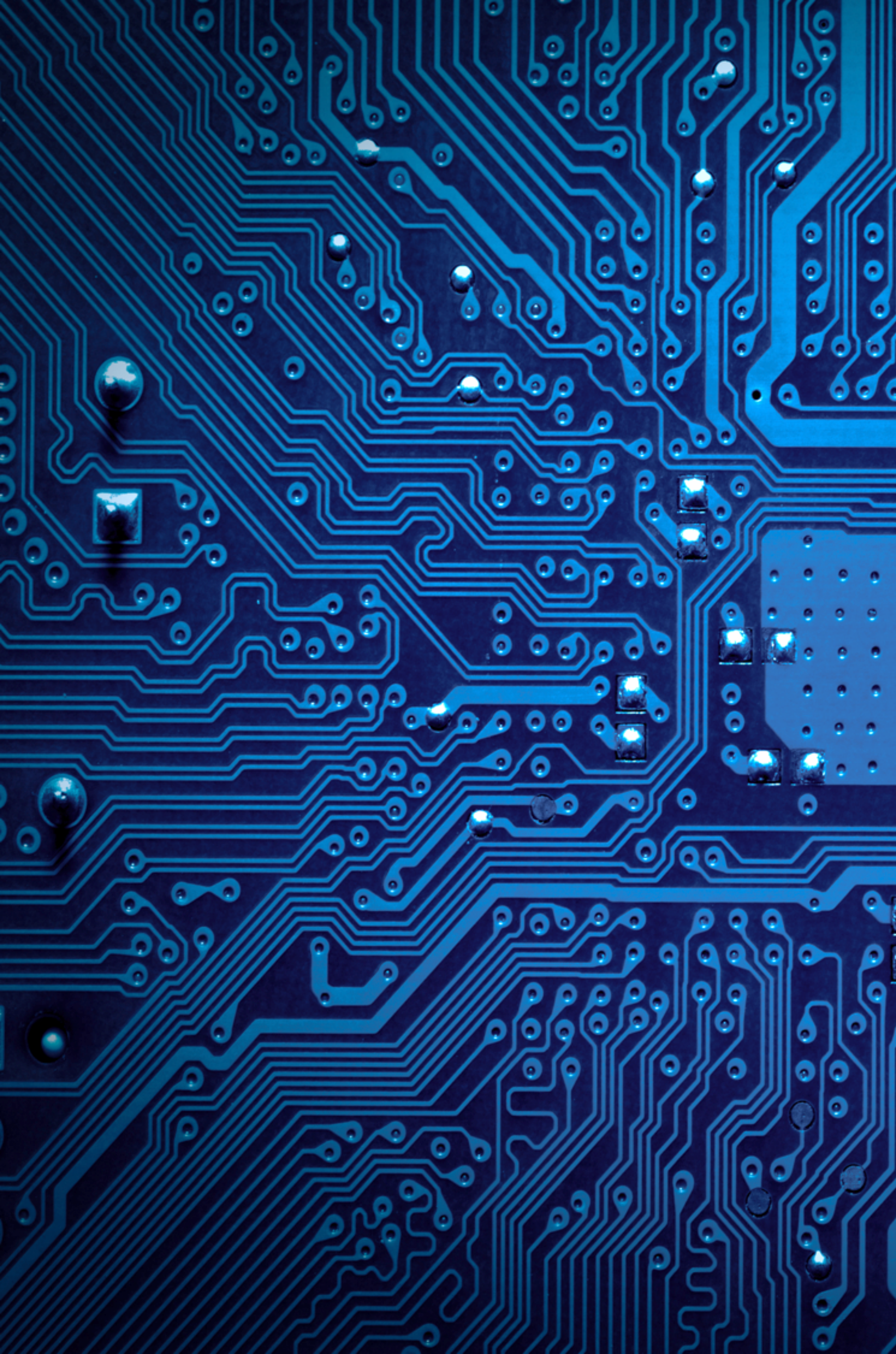


Public Key Cryptography

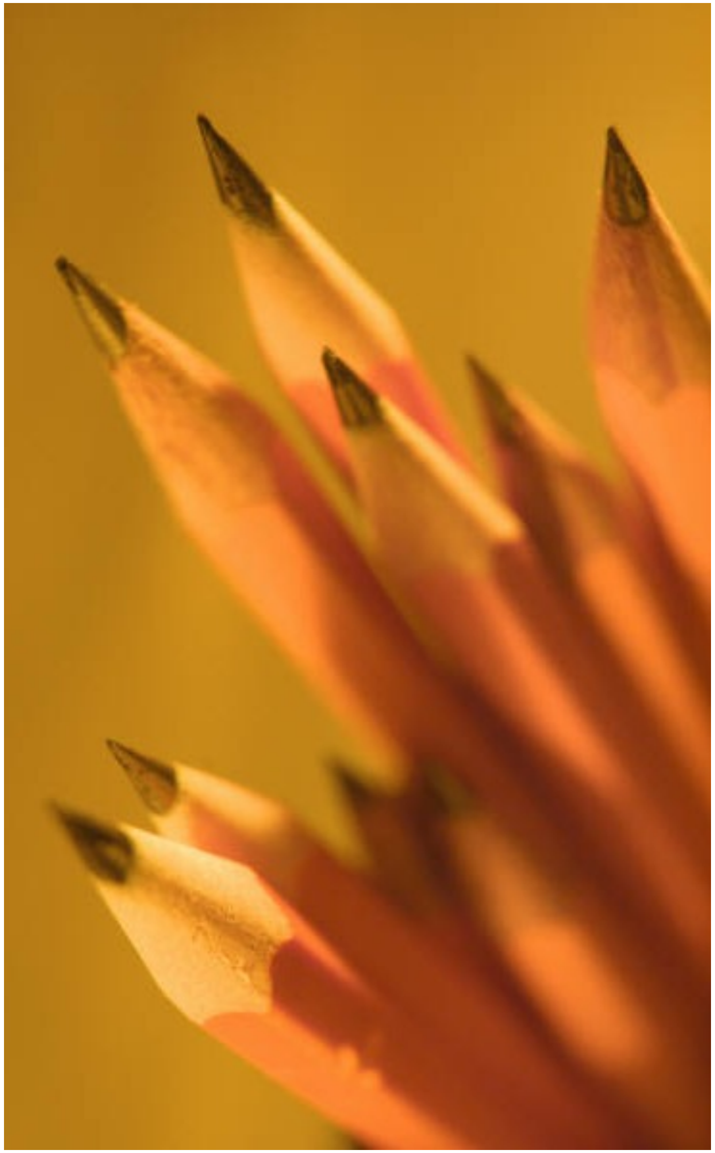
ECE568 – Lecture 11
Courtney Gibson, P.Eng.
University of Toronto ECE

Outline

Introduction
- Encryption/Decryption
- Authentication

RSA Algorithm
- Background
- Operation

Public-Key Infrastructure (PKI)
- Digital Signatures and Certificates
- PKI and PGP
- Certificate Revocation



Introduction

Encryption, decryption, authentication

Public Key Cryptosystems

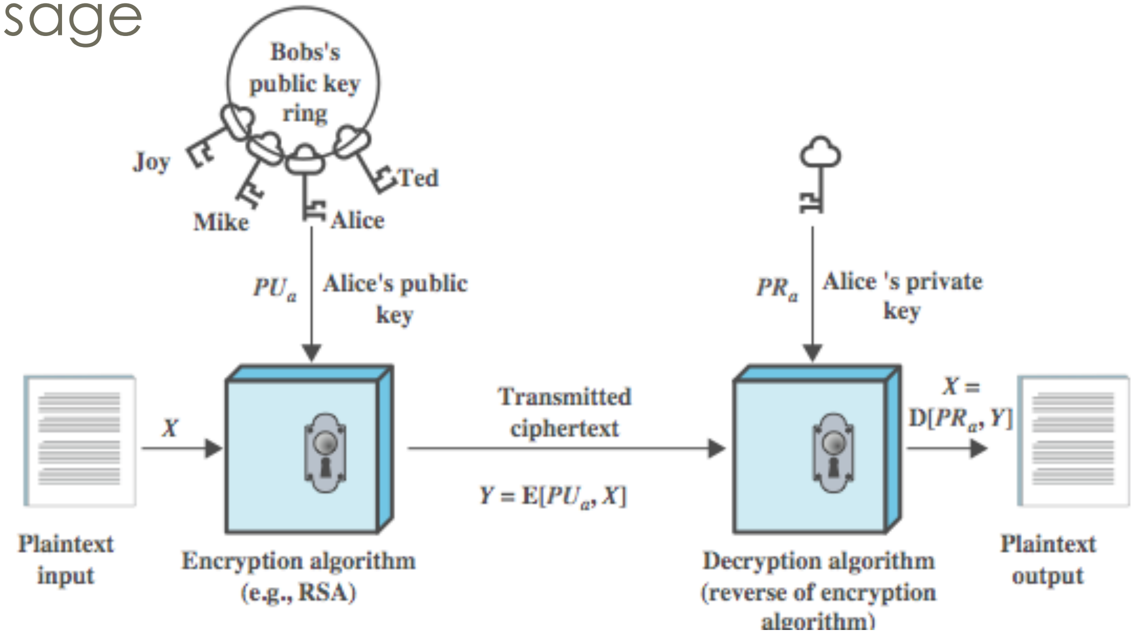
Public Key cryptosystems use a **pair** of keys:

- Every user has a public/private key pair
- The private and public key reveal nothing about each other
- Users distribute the public key, while keeping the private key in a safe place
- Messages encrypted with one key can only be decrypted with the other key

Public Key Encryption

Encryption: the sender encrypts the message with the intended recipient's public key

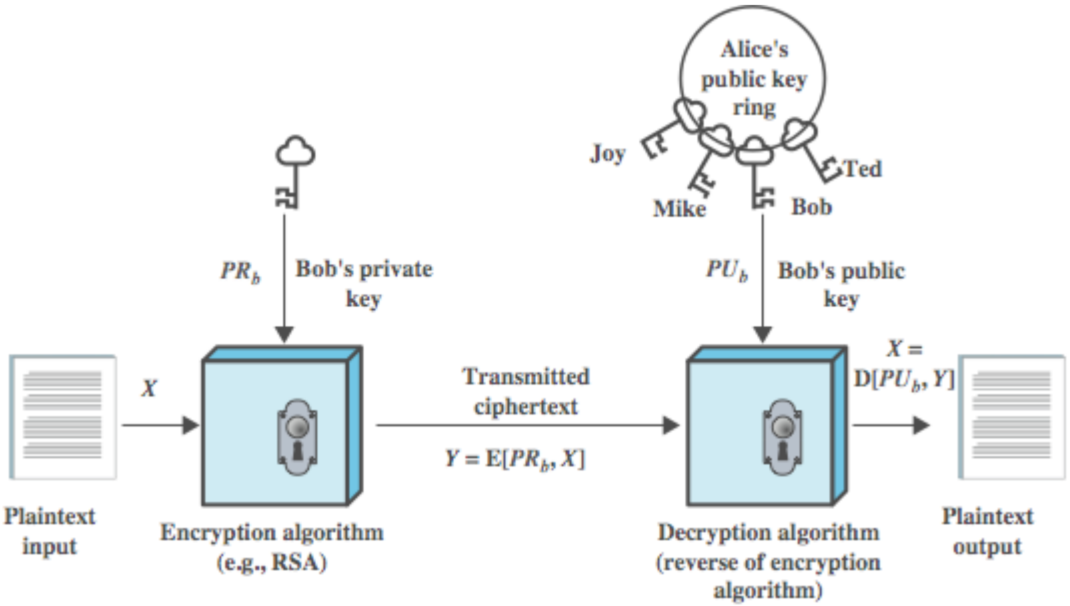
- Only the recipient should have the private key, so only the recipient can decrypt the message



Public Key Authentication

- For authentication, the message is encrypted with the sender's private key (also called **signing**)
  - Any recipient can decrypt using sender's public key
  - Only sender could have encrypted the text we received, thus providing **authentication** and **non-repudiation**
  - Example application: e-mail

Public Key Authentication



RSA

Algorithm, Extended Euclidian Method, limits of RSA

The RSA Algorithm

- RSA is a popular public key algorithm
- RSA was first published by Ron Rivest, Adi Shamir and Len Adleman at MIT in 1977
- They subsequently founded RSA Corporation, which licenses cryptographic protocols and provides security products
- RSA was patented in the US, but the patent expired in 2000
- There is evidence that the GCHQ in London (British equivalent of the American NSA) had invented a similar algorithm as early as 1973

The RSA Algorithm

RSA is also based on modular arithmetic:

- A modulus **n** is produced by multiplying together two large prime numbers **p** and **q** (*i.e.*, **n** = **p•q**)
  - The size of **n** defines the key size
  - 1024-bit RSA uses 1024 bits to represent **n**
- Also, define *phi*: $\varphi = (p-1)(q-1)$

- A public key **e** is selected that is **coprime** to φ
- A private key **d** is then found so that:

  $$e \bullet d = 1 \pmod{\varphi}$$

  - This key can be efficiently computed using the **Extended Euclidean Method**

Extended Euclidean Method

If (**e•d** = 1, mod φ) then (**e•d** + **k** φ = 1) for some constant, **k**:

- Given **e** and φ, we wish to efficiently compute **d**
- The first equation has a unique solution in the case where **e** and φ are coprime

Extended Euclidean Method

The Euclidean table method works as follows:

k	d	r (remainder)	q (quotient)
1	0	φ	
0	1	e	$\varphi \div e$
$k_i = k_{i-2} - q_{i-1} \cdot k_{i-1}$	$d_i = d_{i-2} - q_{i-1} \cdot d_{i-1}$	$r_i = r_{i-2} \bmod r_{i-1}$	$q_i = r_{i-1} \div r_i$

- When $r_i == 1$ then $d = d_i$
- Note that $(k_i \varphi + d_i e) = r_i$ in each step

Example

Say we have $p = 13$ and $q = 17$:

- $n = 221$, $\varphi = 192$, pick $e = 17$, $e \bullet d + k\,\varphi = 1$

k	d	r (remainder)	q (quotient)
1	0	192	
0	1	17	11
1	-11	5	3
-3	34	2	2
7	-79	1	2

$d = (-79 \bmod \varphi) = (-79 \bmod 192) = (192 - 79) = 113$

RSA Operations

- RSA uses modular exponentiation (A^b mod n)
- Encryption uses public key
  - $C = M^e \bmod n$ (note that the modulus is n, not φ)
  - $M < n$ (or else RSA does not work)
- Decryption uses private key
  - $M = C^d \bmod n$
- What values are public and what is kept secret?
  - **n** and **e** can be public
  - Adversary should not be able to get φ, or else **d** can be recovered, so **p** and **q** should be kept secret
- RSA is based on the difficulty of factoring
  - How difficult is it?

Why Does RSA Work?

- We need some math to understand why RSA works
- **Fermat's little theorem**
  - If **n** is prime, and **a** is an integer coprime to **n**, then
  - $a^{n-1} = 1 \bmod n$ (proof omitted, although not difficult)
- **Euler's theorem**
  - If **n** has prime factors **p**, **q**, …
  - Define totient function $\varphi(n) = (p-1)(q-1)\ldots$, then
  - $a^{\varphi(n)} = 1 \bmod$ **n** (**a** must be coprime to **n**)
- Can be used to reduce large powers modulo **n**
  - Say we wish to calculate $7^{222} \bmod 10$
  - Then $\varphi(10) = 4$, and by Euler's theorem, $7^4 \bmod 10 = 1$
  - Then $7^{222} \bmod 10 = 7^{55*4+2} \bmod 10 = 7^2 \bmod 10 = 9$

Why Does RSA Work?

$C^d \bmod n$

$= (M^e \bmod n)^d \bmod n$

$= M^{e \bullet d} \bmod n$

- Recall that $(e \bullet d = 1) \bmod \varphi$, so $(e \bullet d) = (k\varphi + 1)$ for some integer k

$= M^{k\varphi(n) + 1} \bmod n$

$= M^{k\varphi(n)} \bullet M \bmod n$

$= (M^{k\varphi(n)} \bmod n) \bullet (M \bmod n)$

$= (M \bmod n)$ [*applying Euler's theorem*]

$= M$ (assuming $M < n$)

Note there is no formal proof that factoring is hard, and that no easy algorithm exists for it

- However, RSA has been in use for over 20 years and the factoring problem has been known for much longer . . . yet, no solution has come to light

Improper use of RSA

- Recall that a message is **signed** by encrypting the message with the sender's private key
- RSA has very poor resistance to spoofing because encryption uses exponentiation

```
encrypt(K • M)  =  (K • M)^d    // d is private key
                =  K^d • M^d
                =  encrypt(K) • encrypt(M)
```

If someone will sign messages the adversary gives them, then she can trick them into signing messages they have never seen:

- Suppose a victim will not sign message **M**, but the adversary can pick a **K** and get the victim to sign **K•M** and **K**, then a signature on **M** can be recovered

Public-Key Infrastructure (PKI)

Key-signing authorities, key revocation, PGP “web of trust”

Public Key Infrastructure

Does public key cryptography prevent man-in-the-middle attacks?

- If Alice wants to share a key with Bob, encrypting it with Bob's public key prevents Mallory from getting the key
- What if Mallory arranges for Alice to get Mallory's public key, but makes her think that it's Bob's key?
  - Then Alice will encrypt whatever message she wants to send to Bob with Mallory's key
  - Bob won't be able to decrypt and might complain to Alice
  - However, damage is already done, since Mallory can decrypt Alice's message to Bob
- Public Key Infrastructure (PKI) solves this problem

Public Key Infrastructure

PKI is a system where a **trusted third party** (a principal that everyone trusts) vouches for the identity of a key (*i.e.*, that the key belongs to a principle)

- Example:
  - **Assume** that Alice and Bob trust Trent
  - **Assume** that everyone knows Trent's public key
  - Bob creates a public key and goes to Trent; Trent sees both Bob and his public key, creates a **certificate** that says "This public key xxx belongs to Bob" and **signs** it with his (Trent's) private key
  - Bob sends Alice his own public key along with the certificate that bears Trent's signature
  - Alice uses Trent's public key and the certificate to verify Bob's public key

Public Key Infrastructure

- Mallory cannot pretend her key is Bob's key
  - Mallory cannot ask Trent to give her a certificate claiming her key is Bob's key (Trent will only give her a certificate that says the key belongs to Mallory)
  - Mallory cannot forge (or fake) Trent's signature and thus cannot fake a certificate that says her key belongs to Bob

Public Key Infrastructure

Common standard format for certificates is X509

- This format is used in SSL (Lab 3)
- PKI allows using a **chain of certificates** issued by a hierarchy of CAs

Are we back to the **trusted central server** for key exchange? What's the difference in this case?

- Trust level
- Availability, integrity

Certificate Authorities

In the real world, a **Certificate Authority** (CA) plays the role of "Trent"

- Several major CAs (Verisign, Entrust, Equifax, etc.)
- PKI is used within many large organizations

When a browser connects to a secure website, the website sends the browser a certificate that you can verify by viewing the certificate.

PGP: An Alternative to PKI

Instead of having a central trusted party, Pretty Good Privacy (PGP) uses a **web of trust**:

- Every user has a public/private key pair and is capable of signing certificates
- If user Alice is able to verify that a certain public key really belongs to Bob, she can sign a certificate saying so with her private key
- Similarly, if Charlie can verify Alice's public key then he can sign it with his private key
- Trust is transitive: if you trust Charlie, then you can trust Alice, and Bob. If you only trust Alice, then you trust Bob, but not Charlie.

$$\text{Charlie} \xrightarrow{\text{Signed}} \text{Alice} \xrightarrow{\text{Signed}} \text{Bob}$$

Certificate Revocation

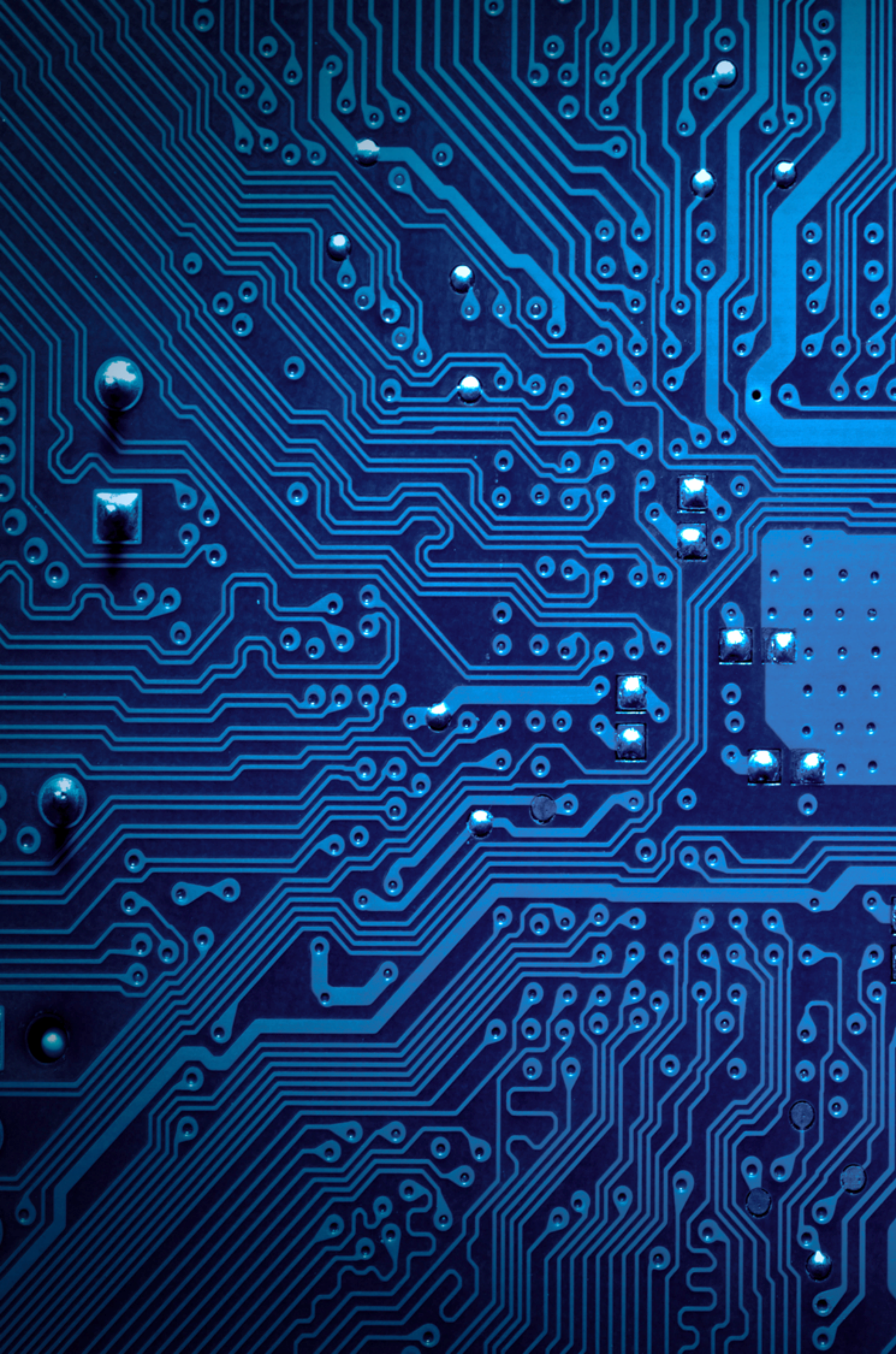
An important aspect of an certificate scheme is the ability to **revoke** certificates:

- Say Microsoft gets a public key certificate from Verisign
- Some hacker is able to steal Microsoft's private key
- The hacker can now create software releases, signed with Microsoft's private key, and the software will appear authentic to the end users' systems
- Microsoft must tell everyone to stop using Microsoft's Public Key to verify signed Microsoft products

- Microsoft uses a **revocation certificate**
  - Certificate should be signed by Verisign (why?)

The revocation certificate is usually created at the time the public key is signed by Verisign

- Certificate should be stored safely (why?)



Questions?