50.042 FCS Summer 2024 Lecture 12 – Asymmetric Cryptography

Felix LOH
Singapore University of Technology and Design

With selected materials adapted from: *Understanding Cryptography: A Textbook for Students and Practitioners, by C. Paar and J. Pelzl*

Asymmetric cryptography: motivation

- Last lecture, we started discussing the topic of key establishment over an insecure channel and how asymmetric crypto can help us achieve that goal
- In particular, we discussed DHKE and the idea of using a *public* key and a *private* key
    - Public key: known to everyone, can be safely transmitted over an insecure channel
    - Private key: must be kept secret
- Using a *one-way function*, as well as the public and private keys, one can compute a *shared* secret session key for legitimate parties

- Now that we have some notion of a functional real-world asymmetric crypto scheme (DHKE), let's <u>formally</u> discuss the basic idea of asymmetric crypto and its principles

Basic idea and principles of asymmetric cryptography

- The basic idea of asymmetric cryptosystems is that different keys are used for encryption and decryption:
  - The **public key**, which is known to everyone, is used for **encryption**
  - The **private key**, which is secret, is used for **decryption**
  - By contrast, symmetric ciphers use the same secret key for both encryption and decryption

- The encryption and decryption functions of most asymmetric cryptosystems are based on selected functions in number theory and as a result, have a compact mathematical description
  - Contrast this with symmetric ciphers, where the encryption and decryption functions cannot be succinctly described by some math equation

Basic idea and principles of asymmetric cryptography

Alice		**Bob**
	$\xleftarrow{\quad k_{pub} \quad}$	$(k_{pub}, k_{pr}) = k$
$y = e_{k_{pub}}(x)$		
	$\xrightarrow{\quad y \quad}$	
		$x = d_{k_{pr}}(y)$

- Basic protocol: Alice wishes to encrypt and send a message *x* to Bob
    1. Bob sends Alice his public key k_{pub} over the insecure channel
    2. Alice then encrypts *x* using k_{pub} to obtain a ciphertext *y*, which she then transmits over the same channel
    3. Bob then decrypts *y* using his private key k_{pr} to obtain *x*
- Even if Oscar knows the public key k_{pub}, he can't use it to decrypt *y*

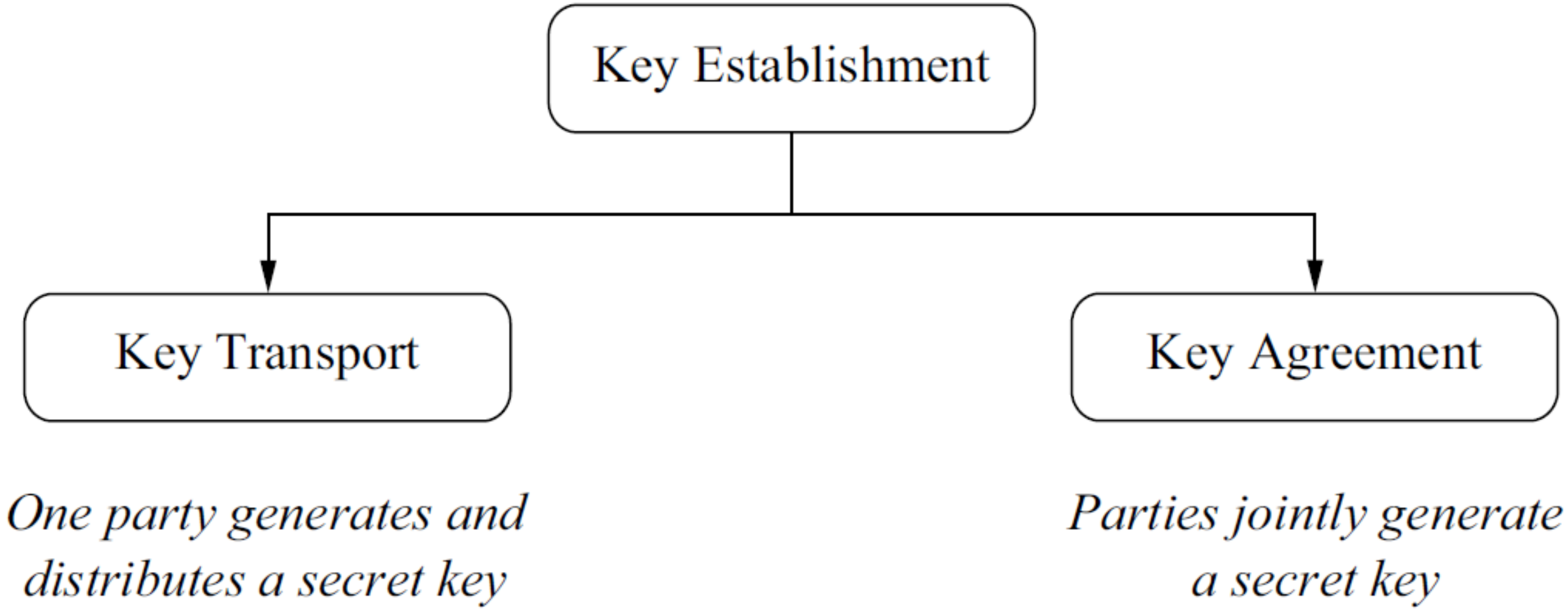
Basic idea and principles of asymmetric cryptography

Alice		**Bob**
	$\xleftarrow{\quad k_{pub} \quad}$	k_{pub}, k_{pr}
choose random k $y = e_{k_{pub}}(k)$		
	$\xrightarrow{\quad y \quad}$	
		$k = d_{k_{pr}}(y)$
encrypt message x: $z = AES_k(x)$		
	$\xrightarrow{\quad z \quad}$	
		$x = AES_k^{-1}(z)$

- In practice, the asymmetric cryptosystem, e.g. RSA, is not used to encrypt the plaintext message itself; rather, it is used to <u>encrypt a shared secret key</u> *k* that will be used in a symmetric cipher, like AES
  - Here, RSA is utilized as a key transport method

Key establishment (recap)

- The techniques for key establishment can be classified into two main groups:
  - Key transport methods (e.g. RSA)
  - Key agreement methods (e.g. DHKE)



Yet more modular arithmetic…

- Before we discuss the Elgamal and RSA algorithms in more detail, there’s a little bit more modular arithmetic concepts to cover first
- Mostly recap from previous lectures, but some additional new theorems

Group: definition (recap)

- A group $G = (S, \circ)$ is a set of elements, S, together with an operation $\circ$ which combines two elements of S. A group **must** have the following four properties:

- The group is **closed**, i.e. $a \circ b = c \in S$ for all $a, b \in S$
- The group operation is **associative**, i.e. $a \circ (b \circ c) = (a \circ b) \circ c$ for all $a, b, c \in S$
- There is an **identity** element $i \in S$ with respect to the operation $\circ$, such that $i \circ a = a \circ i = a$ for all $a \in S$
- For each $a \in S$, there exists an **inverse** element $a^{-1} \in S$, such that $a \circ a^{-1} = a^{-1} \circ a = i$

Order of a finite group (recap)

- We have seen sets with an infinite number of elements, such as $\mathbb{Z}$ and $\mathbb{R}$
- In cryptography, we are generally more interested in sets with a finite number of elements (i.e. finite sets) such as the set $\mathbb{Z}_m$, which has m elements
- The order $|G|$ of a finite group G is the number of elements in G

- E.g. the order of the group $G = (\mathbb{Z}_m, +)$ is $|G| = |\mathbb{Z}_m| = m$

Order of an element in a finite group (recap)

- The order $ord(a)$ of an element $a \in S$ in a group $G = (S, \circ)$ is the smallest positive integer k, such that

  $$a^k = \underbrace{a \circ a \circ a \ldots a \circ a}_{k \text{ times}} = i,$$

  where $i \in S$ is the identity element with respect to the operation $\circ$

- E.g.
  - The order of the element 1 in $G = (\mathbb{Z}_6, +) = (\{0, 1, 2, 3, 4, 5\}, +)$ is 6, since $1^6 = 1 + 1 + 1 + 1 + 1 + 1 \equiv 0 \; mod \; 6$, with element 0 being the identity element
  - The order of the identity element 0 in $G = (\mathbb{Z}_6, +)$ is 1, since $0^1 \equiv 0 \; mod \; 6$
  - The order of the element 2 in $G = (\mathbb{Z}_6, +)$ is 3, since $2^3 = 2 + 2 + 2 \equiv 0 \; mod \; 6$
  - The order of the element 1 in $G = (\mathbb{Z}_m, +)$ is m

Ring: definition (recap)

- A ring is a group (S, ○, *) that has a second operation *, with the following requirements on *:

- The operation * must be **closed**, i.e. $a * b = c \in S$ for all $a, b \in S$
- The operation * must be **associative**, i.e. $a * (b * c) = (a * b) * c$ for all $a, b, c \in S$
- There must be an **identity** element $i' \in S$ with respect to the operation *, such that $i' * a = a * i' = a$ for all $a \in S$
- The operation * must be **distributive** over the operation ○
- *Note: there is no invertibility requirement on the operation **

Integer rings (recap)

- The integer ring is an important example of a ring

- The integer ring $(\mathbb{Z}_m, +, \cdot)$ consists of:

1. The finite set $\mathbb{Z}_m = \{0, 1, 2, \ldots, m\text{-}1\}$
2. Two operations '+' and '·' for all $a, b \in \mathbb{Z}_m$ such that:

   $a + b \equiv c \bmod m, (c \in \mathbb{Z}_m)$

   $a \cdot b \equiv d \bmod m, (d \in \mathbb{Z}_m)$

Field: definition (recap)

- A field $F = (S, +, \cdot)$ is a ring with the following properties:

- All elements of S form an additive group with the group operation ‘+’ and the identity element 0
- All elements of S, except the element 0, form a multiplicative group with the group operation ‘·’ and the identity element 1
  - Particularly, each non-zero element has a multiplicative inverse
- When the two group operations are mixed, the operation ‘·’ is distributive over the operation ‘+’, i.e. $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ for all $a, b, c \in S$

Finite fields (recap)

- In cryptography, we are usually interested in fields with a finite number of elements
  - These fields are known as finite fields or Galois fields
- The number of elements in the field is called the *order* of the field (similar to the order of a finite group)

- The following theorem regarding finite fields is fundamental:

  A field with order m only exists if m is a prime power, i.e. $m = p^n$ for some positive integer n and prime integer p. p is called the characteristic of the finite field.

Prime fields (recap)

- A prime field $GF(p)$ is a Galois (finite) field of prime order (i.e. $n = 1$)
- The two operations of the field are integer addition *modulo p* and integer multiplication *modulo p*

- The following important theorem defines a prime field:

  Let p be a prime integer. The integer ring $\mathbb{Z}_p$ is denoted as $GF(p)$ and is referred to as a prime field, or as a Galois field with a prime number of elements. All non-zero elements of $GF(p)$ have an inverse. Arithmetic in $GF(p)$ is done *modulo p*.

Euler's phi function (recap)

- The number of integers in the set $\mathbb{Z}_m$ that are *relatively prime* to *m* is denoted by $\Phi(m)$
  - An integer *j* is relatively prime to *m* if $gcd(j, m) = 1$

- E.g.
  - When $m = 6$, we have $\mathbb{Z}_6 = \{0, 1, 2, 3, 4, 5\}$. Then, we have $gcd(0, 6) = 6$; $gcd(1, 6) = 1$; $gcd(2, 6) = 2$; $gcd(3, 6) = 3$; $gcd(4, 6) = 2$ and $gcd(5, 6) = 1$. So, there are two integers that are relatively prime to 6 and thus, $\Phi(6) = 2$

  - When $m = 5$, we have $\mathbb{Z}_5 = \{0, 1, 2, 3, 4\}$. Then, we have $gcd(0, 5) = 5$; $gcd(1, 5) = 1$; $gcd(2, 5) = 1$; $gcd(3, 5) = 1$ and $gcd(4, 5) = 1$. So, there are four integers that are relatively prime to 5 and thus, $\Phi(5) = 4$

Euler's phi function: comments (recap)

- For large m, calculating Euler's phi function $\Phi(m)$ by going through all elements of the set $\mathbb{Z}_m$ and computing the greatest common divisor is extremely slow
- However, if we know the *factorization* of m, then there is a much faster method to compute $\Phi(m)$
- This property is critical to the RSA algorithm

Fast method to compute Euler's phi function (recap)

- Let *m* have the following *factorized* form:

  $m = p_1^{e_1} \cdot p_2^{e_2} \cdot \ \ldots \cdot p_n^{e_n}$, where the p_j are distinct prime numbers and the e_j are positive integers

- Then we have $\Phi(m) = \prod_{j=1}^{n} (p_j^{e_j} - p_j^{e_j - 1})$

- E.g. when $m = 240 = 2^4 \cdot 3 \cdot 5$, we have

  $\Phi(240) = (2^4 - 2^3) \cdot (3^1 - 3^0) \cdot (5^1 - 5^0) = 8 \cdot 2 \cdot 4 = 64$

- Note that computing $\Phi(240)$ by computing the gcd 240 times would have been much slower than using the formula above; but the formula requires that we know the factorization of *m*

Fermat’s Little Theorem

Let *a* be an integer and *p* be a prime number. Then we have

$\boldsymbol{a^p \equiv a \bmod p}$

- Note that arithmetic in a prime field *GF*(*p*) is performed modulo *p*, so the above theorem holds for all elements of a prime field (except 0), and the theorem can be rewritten as $a \cdot a^{p-2} \equiv 1 \bmod p$
- This has a useful implication:
  - a^{p-2} is the multiplicative inverse of *a*, thus we can invert any integer *a*, modulo some prime *p*, by using the formula $a^{-1} \equiv a^{p-2} \bmod p$

Euler's Theorem

Let a and m be integers that are relatively prime, i.e. $gcd(a, m) = 1$. Then we have $a^{\Phi(m)} \equiv 1 \bmod m$

- Arithmetic in an integer ring $(\mathbb{Z}_m, +, \cdot)$ is performed modulo m, so the above theorem holds for all elements of an integer ring
- Fermat's Little Theorem is special case of this theorem, i.e. when m is a prime number

- This theorem is critical to the RSA algorithm

One-way functions (recap)

- A function $f()$ is a one-way function if:
  1) $y = f(x)$ is computationally easy, and
  2) $x = f^{-1}(y)$ is computationally infeasible.
- Important note:
  - The ‘one-way function’ defined here for asymmetric cryptosystems is **not the same** as a ‘one-way function’ defined in the context of *hash functions*
  - In the case of a hash function, the ‘one-way function’ refers to a non-invertible function $f(x)$ where the inverse $f^{-1}(y)$ does **not** mathematically exist
  - In the case of asymmetric cryptosystems, the inverse of a one-way function does exist, but is extremely difficult to compute

- Most asymmetric crypto schemes of practical use, including RSA and DHKE, are based on a one-way function

DHKE (recap)

a. Diffie-Hellman setup phase (picking of domain parameters):
   1. Choose a large prime p
   2. Choose a generator $\alpha \in \{2, 3, \ldots, p-2\}$
   3. Publish the domain parameters p and α (for Alice and Bob to use in the next phase – key exchange)

DHKE (recap)

Alice		**Bob**
choose $a = k_{pr,A} \in \{2, \ldots, p-2\}$		choose $b = k_{pr,B} \in \{2, \ldots, p-2\}$
compute $A = k_{pub,A} \equiv \alpha^a \bmod p$		compute $B = k_{pub,B} \equiv \alpha^b \bmod p$
	$\xrightarrow{k_{pub,A}=A}$	
	$\xleftarrow{k_{pub,B}=B}$	
$k_{AB} = k_{pub,B}^{k_{pr,A}} \equiv B^a \bmod p$		$k_{AB} = k_{pub,A}^{k_{pr,B}} \equiv A^b \bmod p$

b. Diffie-Hellman key exchange phase (generate a joint secret key k_{AB}):
   1. Alice picks a private key, $k_{pr,\,A} = a \in \{2, 3, \ldots, p\text{-}2\}$ and computes her public key $k_{pub,\,A} = A \equiv \alpha^a \; mod \; p$, then sends the public key over to Bob
   2. Likewise, Bob picks a private key, $k_{pr,\,B} = b \in \{2, 3, \ldots, p\text{-}2\}$ and computes his public key $k_{pub,\,B} = B \equiv \alpha^b \; mod \; p$, then sends the public key over to Alice
   3. Upon receiving each other's public key, Alice computes the joint secret key $k_{AB} = (k_{pub,\,B})^a \equiv \alpha^{ba} \; mod \; p$ and Bob computes $k_{AB} = (k_{pub,\,A})^b \equiv \alpha^{ab} \; mod \; p$

Elgamal: an extension to DHKE

- Recall that the DHKE protocol generates a joint secret key k_{AB} for use in a symmetric cipher
  - It doesn't really perform any encryption and decryption
- We can extend the functionality of DHKE, such that it can function as a cipher, which is known as Elgamal:
  1. Suppose Alice wishes to send Bob a message $x \in \mathbb{Z}^*_p$
  2. After Alice and Bob have both computed the shared secret key k_{AB} via DHKE, Alice can encrypt the plaintext x by multiplying it with k_{AB}, modulo p, to obtain the ciphertext y, i.e. $\boldsymbol{y \equiv x \cdot k_{AB} \bmod p}$
     - k_{AB} is used as a multiplicative mask here
  3. Upon receiving the ciphertext y, Bob can decrypt it by multiplying y with the inverse of k_{AB}, modulo p, i.e. $\boldsymbol{x \equiv y \cdot k_{AB}^{-1} \bmod p}$
     - The inverse of k_{AB} can be found by using Fermat's Little Theorem

RSA: introduction

- An asymmetric cryptosystem proposed by Ron Rivest, Adi Shamir and Leonard Adleman in 1977
- It is commonly used in many protocols relevant to internet security:
  - Most public key infrastructure (PKI) products
  - Protocols for certificates and key exchange, e.g. SSL and TLS
  - Secure email, e.g. Outlook, Pretty Good Privacy (PGP)

- It is capable of encrypting and decrypting messages, but is typically used to encrypt/decrypt keys for use in *symmetric* ciphers
  - That’s mainly because of performance issues – we’ll talk about those issues later in this lecture

RSA: introduction

- It uses the integer factorization problem as its one-way function
  - More on this later in the lecture
- Encryption and decryption is performed in the integer ring $\mathbb{Z}_n$
  - Because $\mathbb{Z}_n$ contains only the elements {0, 1, …, n-1}, the binary value of both the plaintext x and the ciphertext y must be less than n
- Modular exponentiation plays an important role in RSA

- RSA consists of two phases, similar to DHKE:
  a. Key generation (setup)
  b. Encryption/decryption

RSA

a. Key generation phase (setup):
   - The output of this phase is $k_{pub} = (n, e)$ and $k_{pr} = d$
   1. Choose two large prime numbers p and q
   2. Compute $n = p \cdot q$
   3. Calculate $\Phi(n) = (p - 1) \cdot (q - 1)$
   4. Select the public exponent $e \in \{1, 2, \ldots, \Phi(n)-1\}$, such that $gcd(e, \Phi(n)) = 1$
   5. Compute the private key d, such that $d \cdot e \equiv 1 \; mod \; \Phi(n)$, i.e. d is the inverse of e modulo $\Phi(n)$

- For security, n should be 1024 bits long at a minimum

RSA

b. Encryption/decryption phase:

- **Encryption**: Given the plaintext x and the public key $k_{pub} = (n, e)$, the encryption function is $\boldsymbol{y = e_{k_{pub}}(x) \equiv x^e \bmod n}$, where $x, y \in \mathbb{Z}_n$

- **Decryption**: Given the ciphertext y and the private key $k_{pr} = d$, the decryption function is $\boldsymbol{x = d_{k_{pr}}(y) \equiv y^d \bmod n}$, where $x, y \in \mathbb{Z}_n$

RSA: example

Alice		**Bob**
message $x = 4$		1. choose $p = 3$ and $q = 11$ 2. $n = p \cdot q = 33$ 3. $\Phi(n) = (3-1)(11-1) = 20$ 4. choose $e = 3$ 5. $d \equiv e^{-1} \equiv 7 \bmod 20$
	$\xleftarrow{k_{pub}=(33,3)}$	
$y = x^e \equiv 4^3 \equiv 31 \bmod 33$		
	$\xrightarrow{y=31}$	
		$y^d = 31^7 \equiv 4 = x \bmod 33$

Proof of the correctness of RSA encryption

- Let's show that RSA decryption is the inverse of RSA encryption, i.e.

  $d_{k_{pr}}(y) = d_{k_{pr}}[e_{k_{pub}}(x)] \equiv (x^e)^d \equiv x^{d \cdot e} \equiv x \; mod \; n$

- Starting with the expression $d \cdot e \equiv 1 \; mod \; \Phi(n)$, we can use the definition of the modulo operation to re-express that equation as:

  $d \cdot e \equiv 1 + t \cdot \Phi(n)$, for some integer t

- Then we have $d_{k_{pr}}(y) = x^{d \cdot e} \equiv x^{1 + t \cdot \Phi(n)} \equiv x \cdot x^{t \cdot \Phi(n)} \equiv (x^{\Phi(n)})^t \cdot x \; mod \; n$

Proof of the correctness of RSA encryption

- This simplifies our problem to proving that: $x \equiv (x^{\Phi(n)})^t \cdot x \bmod n$

- <u>Case 1:</u> x and n are relatively prime, i.e. $gcd(x, n) = 1$:
  - When x and n are relatively prime, Euler's Theorem holds, i.e. $x^{\Phi(n)} \equiv 1 \bmod n$
  - As a result, $(x^{\Phi(n)})^t \cdot x \equiv 1^t \cdot x \equiv x \bmod n$

Proof of the correctness of RSA encryption

- Case 2: x and n are **not** relatively prime, i.e. $gcd(x, n) \neq 1$:
  - Since $n = p \cdot q$, we have $gcd(x, n) = gcd(x, p \cdot q) \neq 1$
  - Because p and q are both prime integers, and $x < n$, this means that either p is a factor of x, or q is a factor of x (but not both p and q)
  - So, we have either $x = r \cdot p$ for some integer r, or $x = s \cdot q$ for some integer s
  - Now let's assume that $x = s \cdot q$; this implies that $gcd(x, p) = 1$
  - Thus, we can use Euler's Theorem again, i.e. $x^{\Phi(p)} \equiv 1 \ mod\ p$, to show that:
    $(x^{\Phi(n)})^t \cdot x \equiv (x^{(p-1) \cdot (q-1)})^t \cdot x \equiv (x^{p-1})^{t \cdot (q-1)} \cdot x$
    $\equiv (x^{\Phi(p)})^{t \cdot (q-1)} \cdot x \equiv 1^{t \cdot (q-1)} \cdot x \equiv x \ mod\ n$
  - We will obtain a similar result if we assume that $x = r \cdot p$ instead

The integer factorization problem

- The integer factorization problem is the one-way function behind RSA
  - This problem forms the foundation of the security of RSA

- Multiplying two large prime numbers is computationally easy
- By contrast, given the product of two large prime numbers, it is very difficult to factorize this product to obtain the two prime numbers
  - So, in the case of RSA, it is difficult to factorize n into its primes p and q and as a result, it is difficult to compute $\Phi(n)$

RSA: comments

- There are a couple of non-trivial tasks during the key generation phase
  - Choosing the two large prime numbers *p* and *q* is not straightforward, but one way to do that is to pick two large numbers *p* and *q* randomly, then use a primality test, such as the Fermat test or the Miller-Rabin test, to check that the two numbers chosen are prime – if *p* and/or *q* are not prime, then pick the number(s) again
  - Computing the private key $k_{pr} = d$ is also not trivial, but this can be obtained using the extended Euclidean algorithm (EEA)
- The encryption and decryption functions, which involve modular exponentiation, can be computationally intensive – even though those operations can be somewhat efficiently computed using the square and multiply algorithm, it results in performance issues

RSA: comments

- The performance issues related to modular exponentiation lead to the fact that RSA is about 100 to 1000 times slower than symmetric ciphers such as AES
- This is why RSA is typically used to encrypt a shared secret key, that can be safely transmitted over an insecure channel, for use in a symmetric cipher like AES

RSA in practice: padding

- What we have discussed so far is the “schoolbook RSA” cryptosystem, which has several weaknesses:

a) The encryption is deterministic; for a specific public key, a particular plaintext is always mapped to a particular ciphertext
   - So the attacker Oscar can derive some statistical properties of the plaintext from the ciphertext

b) Schoolbook RSA is *malleable*, which means that Oscar can transform a RSA ciphertext into another ciphertext that results in a <u>known</u> transformation of the original plaintext
   - Oscar does not need to be able to decrypt the ciphertext; he just needs to be capable of manipulating the plaintext in a predictable manner
   - For example, Oscar can replace the ciphertext y with $g^e \cdot y$, where g is some integer; when Bob decrypts the modified ciphertext, he gets $(g^e \cdot y)^d = g^{ed} \cdot x^{ed} \equiv g \cdot x \bmod n$

RSA in practice: padding

- These weaknesses can be addressed using padding, which embeds a random structure into the plaintext before encryption
- Modern techniques like *Optimal Asymmetric Encryption Padding* (OAEP) are used for padding RSA messages
- When the message is decrypted, the structure of the decrypted message is verified to ensure that no decryption error has occurred

Security of RSA

- How might the attacker Eve/Oscar compromise the security of DHKE?

- There are three general forms of attacks against RSA:

a. Protocol attacks:
    - These are attacks that exploit weaknesses in the way RSA is used
    - Most of these attacks exploit the malleability of RSA, and these attacks can be thwarted by the use of padding

Security of RSA

b. Mathematical attacks:

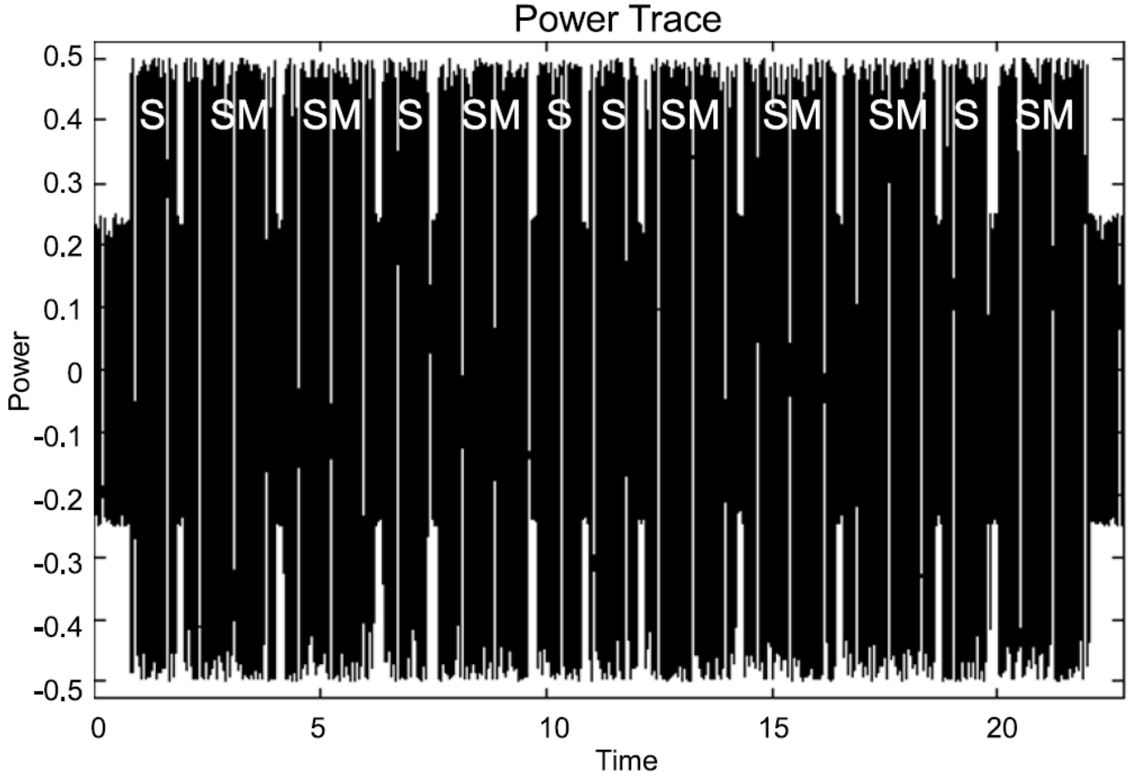
- The best cryptanalytical method involves factoring the modulus n
- Oscar only knows the modulus n, the public key e and the ciphertext y
- His goal: Calculate the private key d, where $d \cdot e \equiv 1 \; mod \; \Phi(n)$
- However, he **cannot** use the EEA to compute d, which is the inverse of e, because he does **not** know the value of $\Phi(n)$
- At best, he can try to factorize n into its primes p and q. If he is successful in factorizing n, then he can do the following:
  1. Compute $\Phi(n) = (p - 1) \cdot (q - 1)$
  2. Calculate $d \equiv e^{-1} \; mod \; \Phi(n)$
  3. Obtain the plaintext $x \equiv y^d \; mod \; n$

- This kind of attack can be defeated by making the modulus n sufficiently large, at least 1024 bits long, so that the factorization of n is very difficult

Security of RSA

c. Side-channel attacks:
   - These attacks exploit information about the private key that is leaked through the timing behaviour or the power consumption
   - Such attacks require direct physical access to the RSA implementation, e.g. a smart card or microprocessor
   - For example, the next slide shows a power trace of a microprocessor executing the square and multiply algorithm during modular exponentiation

Security of RSA



- From the power trace, one can deduce the bits of the private key: ‘0’ for Square (S) and ‘1’ for Square and Multiply (SM) → the key is $\underline{1}011010011101_2$
- This attack can be prevented by using dummy operations or power masking

Digital signatures: prelude

- Recall in a previous lecture that MACs can provide *message integrity* and *authentication*, but **cannot** provide *non-repudiation*
- However, we can use an asymmetric cryptosystem to provide *non-repudiation* as a service
- The general idea:
  - Bob can use his <u>private key</u> to "decrypt" a <u>plaintext</u> message *x* – this is essentially a digital signature. He then sends *x* and the digital signature over to Alice
  - Alice can then verify that Bob was the sender of *x*, by "encrypting" the digital signature using Bob's <u>public key</u>
  - Since only Bob is in possession of his private key, he cannot later deny that he was the sender of the plaintext *x*
- We will discuss digital signatures in more detail in the next lecture