 WILLIAM & MARY

---

**CSCI 454/554 Computer and Network Security**

Topic 5.2 Public Key Cryptography

---

---

---


---

---

---

---

---

 WILLIAM & MARY

---

**Outline**

1. Introduction
2. RSA
3. Diffie-Hellman Key Exchange
4. Digital Signature Standard

2

---

---

---


---

---

---

---

---

 WILLIAM & MARY

---

Introduction

---

---

---

---

---

---

---

---

**Public Key Cryptography** WILLIAM & MARY

The diagram shows a flow from 'plaintext' to 'encryption' (orange box) to 'ciphertext' to 'decryption' (blue box) back to 'plaintext'. Below 'encryption' is a 'Public key' box, and below 'decryption' is a 'Private key' box. A dashed arrow labeled 'different!' points from the Public key to the Private key.

- Invented and published in 1975
- A **public / private key pair** is used
  - public key can be announced to everyone
  - private key is kept secret by the owner of the key
- Also known as **asymmetric** cryptography
- Much **slower** to compute than **secret key cryptography**

4

---

---

---

---

---

---

---

---

**Applications of Public Key Crypto** WILLIAM & MARY

1. Message integrity with **digital signatures**  
 Alice computes hash, signs with her private key (no one else can do this without her key)  
 Bob verifies hash on receipt using Alice's public key using the verification equation

The diagram shows 'Plaintext' entering a box labeled 'Alice Signs'. Below this box is 'Alice's Private Key'. An arrow points from the 'Alice Signs' box to another box labeled 'Bob Verifies Signature'. Above this arrow is 'Plaintext with digital signature'. Below the 'Bob Verifies Signature' box is 'Alice's Public Key'. An arrow points from the 'Bob Verifies Signature' box to the right, labeled 'Valid / Not Valid'.

5

---

---

---

---

---

---

---

---

**Applications (Cont'd)** WILLIAM & MARY

- The digital signature is verifiable by anybody
- Only one person can sign the message: **non-repudiation**
- Non-repudiation is only achievable with public key cryptography

6

---

---

---

---

---

---

---

---

**Applications (Cont'd)** WILLIAM & MARY

2. **Communicating securely** over an insecure channel

- Alice encrypts plaintext using Bob's public key, and Bob decrypts ciphertext using his private key
- No one else can decrypt the message (because they don't have Bob's private key)

The diagram shows a flow from left to right. On the far left, the word 'Plaintext' has an arrow pointing to a yellow box labeled 'Alice Encrypts'. Below this box is a red box labeled 'Bob's Public Key' with an upward-pointing arrow. An arrow labeled 'Ciphertext' points from the 'Alice Encrypts' box to a blue box labeled 'Bob Decrypts'. Below this box is a red box labeled 'Bob's Private Key' with an upward-pointing arrow. Finally, an arrow labeled 'Plaintext' points from the 'Bob Decrypts' box to the right.

7

---

---

---

---

---

---

---

---

**Applications (Cont'd)** WILLIAM & MARY

3. **Secure storage** on insecure medium

- Alice encrypts data using her public key
- Alice can decrypt later using her private key

4. **User Authentication**

- Bob proves his identity to Alice by using his private key to perform an operation (without divulging his private key)
- Alice verifies result using Bob's public key

8

---

---

---

---

---

---

---

---

**Applications (Cont'd)** WILLIAM & MARY

5. **Key exchange** for **secret key** crypto

- Alice and Bob use public key crypto to negotiate a shared secret key between them

9

---

---

---


---

---

---

---

---

 **Public Key Algorithms** WILLIAM & MARY

- Public key algorithms covered in this class, and their applications

System	Encryption / Decryption?	Digital Signatures?	Key Exchange?
RSA	Yes	Yes	Yes
Diffie-Hellman			Yes
DSA		Yes	

10

---

---

---


---

---

---

---

---

 **Public-Key Requirements** WILLIAM & MARY

- It must be **computationally**
  - easy** to generate a public / private key pair
  - hard** to determine the private key, given the public key
- It must be **computationally**
  - easy** to encrypt using the public key
  - easy** to decrypt using the private key
  - hard** to recover the plaintext message from just the ciphertext and the public key

11

---

---

---


---

---

---

---

---

 **Trapdoor One-Way Functions** WILLIAM & MARY

- Trapdoor** one-way function
  - $Y=f_k(X)$ : easy to compute if  $k$  and  $X$  are known
  - $X=f^{-1}_k(Y)$ : easy to compute if  $k$  and  $Y$  are known
  - $X=f^{-1}_k(Y)$ : hard if  $Y$  is known but  $k$  is **unknown**
- Goal of designing public-key algorithm is to find appropriate trapdoor one-way function

12

---

---

---


---

---

---

---

---



WILLIAM  
& MARY

## The RSA Cipher

---

---

---


---

---

---

---

---



WILLIAM  
& MARY

### RSA (Rivest, Shamir, Adleman)

- The most popular public key method
  - provides both public key encryption and digital signatures
- Basis: **factorization of large numbers** is hard
- Variable key length (**1024 bits** or greater)
- Variable plaintext block size
  - **plaintext** block size must be **smaller** than key size
  - **ciphertext** block size is **same** as key size

14

---

---

---


---

---

---

---

---



WILLIAM  
& MARY

### Generating a Public/Private Key Pair

- Find (using Miller-Rabin) large primes  $p$  and  $q$
- Let  $n = p * q$ 
  - do not disclose  $p$  and  $q$ !
  - $\phi(n) = ???$
- Choose an  $e$  that is **relatively prime to  $\phi(n)$** 
  - **public** key =  $\langle e, n \rangle$
- Find  $d =$  **multiplicative inverse of  $e$  mod  $\phi(n)$**   
(i.e.,  $e * d = 1 \text{ mod } \phi(n)$ )
  - **private** key =  $\langle d, n \rangle$

15

---

---

---



---

---

---

---

---

 **RSA Operations** 

- For plaintext message  $m$  and ciphertext  $c$

Encryption:  $c = m^e \bmod n, m < n$

Decryption:  $m = c^d \bmod n$

Signing:  $s = m^d \bmod n, m < n$

Verification:  $m = s^e \bmod n$

16

---

---

---



---

---

---

---

---

 **RSA Example: Encryption and Signing** 

- Choose  $p = 23, q = 11$  (both primes)
  - $n = p \cdot q = 253$
  - $\phi(n) = (p-1)(q-1) = 220$
- Choose  $e = 39$  (relatively prime to 220)
  - public key =  $\langle 39, 253 \rangle$
- Find  $e^{-1} \bmod 220 = d = 79$   
(note:  $39 \cdot 79 \equiv 1 \pmod{220}$ )
  - private key =  $\langle 79, 253 \rangle$

17

---

---

---



---

---

---

---

---

 **Example (Cont'd)** 

- Suppose plaintext  $m = 80$

Encryption  
 $c = 80^{39} \bmod 253 = \underline{\hspace{2cm}}$  ( $c = m^e \bmod n$ )

Decryption  
 $m = \underline{\hspace{2cm}}^{79} \bmod 253 = 80$  ( $c^d \bmod n$ )

Signing (in this case, for entire message  $m$ )  
 $s = 80^{79} \bmod 253 = \underline{\hspace{2cm}}$  ( $s = m^d \bmod n$ )

Verification  
 $m = \underline{\hspace{2cm}}^{39} \bmod 253 = 80$  ( $s^e \bmod n$ )

18

---

---

---


---

---

---

---

---

 **Example (Cont'd)** WILLIAM & MARY

- Suppose plaintext  $m = 80$

Encryption  
 $c = 80^{39} \bmod 253 = 37 \quad (c = m^e \bmod n)$

Decryption  
 $m = 37^{79} \bmod 253 = 80 \quad (c^d \bmod n)$

Signing (in this case, for entire message  $m$ )  
 $s = 80^{79} \bmod 253 = 224 \quad (s = m^d \bmod n)$

Verification  
 $m = 224^{39} \bmod 253 = 80 \quad (s^e \bmod n)$

19

---

---

---


---

---

---

---

---

 **Using RSA for Key Negotiation** WILLIAM & MARY

- Procedure
  - $A$  sends random number  $R1$  to  $B$ , encrypted with  $B$ 's public key
  - $B$  sends random number  $R2$  to  $A$ , encrypted with  $A$ 's public key
  - $A$  and  $B$  both decrypt received messages using their respective private keys
  - $A$  and  $B$  both compute  $K = H(R1 \oplus R2)$ , and use that as the shared key

20

---

---

---


---

---

---

---

---

 **Key Negotiation Example** WILLIAM & MARY

- For Alice,  $e = 39$ ,  $d = 79$ ,  $n = 253$
- For Bob,  $e = 23$ ,  $d = 47$ ,  $n = 589 (=19 \cdot 31)$
- Let  $R1 = 15$ ,  $R2 = 55$ 
  - Alice sends  $306 = 15^{23} \bmod 589$  to Bob
  - Bob sends  $187 = 55^{39} \bmod 253$  to Alice
  - Alice computes  $R2 = 55 = 187^{79} \bmod 253$
  - Bob computes  $R1 = 15 = 306^{47} \bmod 589$
  - $A$  and  $B$  both compute  $K = H(R1 \oplus R2)$ , and use that as the shared key

21

---

---

---



---

---

---

---

---

 **Proof of Correctness ( $D(E(m)) = m$ )** 

- Given
  - public key =  $\langle e, n \rangle$  and private key =  $\langle d, n \rangle$
  - $n = p \cdot q$ ,  $\phi(n) = (p-1)(q-1)$
  - $e \cdot d \equiv 1 \pmod{\phi(n)}$
- If encryption is  $c = m^e \pmod n$ , decryption...
  - $c^d \pmod n$
  - $= (m^e)^d \pmod n = m^{ed} \pmod n = m^{ed \pmod{\phi(n)}} \pmod n$
  - $= m \pmod n$  (why?)
  - $= m$  (since  $m < n$ )
- (digital signature proof is similar)

22

---

---

---



---

---

---

---

---

 **Is RSA Secure?** 

- $\langle e, n \rangle$  is public information
- If you could **factor**  $n$  into  $p \cdot q$ , then
  - could compute  $\phi(n) = (p-1)(q-1)$
  - could compute  $d = e^{-1} \pmod{\phi(n)}$
  - would know the private key  $\langle d, n \rangle$ !
- But:** factoring large integers is hard!
  - classical problem worked on for centuries; no **known** reliable, fast method

23

---

---

---



---

---

---

---

---

 **Security (Cont'd)** 

- At present, key sizes of 1024 bits are considered to be secure, but **2048 bits is better**
- Tips** for making  $n$  **difficult to factor**
  - $p$  and  $q$  lengths should be similar (ex.:  $\sim 500$  bits each if key is 1024 bits)
  - both  $(p-1)$  and  $(q-1)$  should contain a "large" prime factor
  - $\gcd(p-1, q-1)$  should be "small"
  - $d$  should be larger than  $n^{1/4}$

24

---

---

---

---


---

---

---

---



 **Attacks Against RSA** WILLIAM & MARY

- Brute force: try all possible private keys
  - can be defeated by using a large enough key space (e.g., 1024 bit keys or larger)
- Mathematical attacks
  - factor  $n$  (possible for special cases of  $n$ )
  - determine  $d$  directly from  $e$ , without computing  $\phi(n)$ 
    - at least as difficult as factoring  $n$

25

---

---

---


---

---

---

---

---

 **Attacks (Cont'd)** WILLIAM & MARY

- Probable-message attack** (using  $\langle e, n \rangle$ )
  - encrypt **all possible** plaintext messages
  - try to find a match between the ciphertext and one of the encrypted messages
  - only works for small plaintext message sizes
- Solution: pad plaintext message with random text before encryption
- PKCS #1 v1 specifies this padding format:

00	02	R1	R2	R3	R4	R5	R6	R7	R8	00	data...
----	----	----	----	----	----	----	----	----	----	----	---------

each 8 bits long

26

---

---

---


---

---

---

---

---

 **Timing Attacks Against RSA** WILLIAM & MARY

- Recovers the private key from the **running time** of the decryption algorithm
- Computing  $m = c^d \bmod n$  using repeated squaring algorithm:

```

* m = 1;
* for i = k-1 downto 1
  m = m*m mod n;
  if di == 1
    then m = m*c mod n;
* return m;

```

27

---

---

---


---

---

---

---

---

 **Timing Attacks (Cont'd)** WILLIAM & MARY

- The attack proceeds bit by bit
- Attacker assumed to know  $\mathbf{c}$ ,  $\mathbf{m}$
- Attacker is able to determine bit  $i$  of  $d$  because for some  $\mathbf{c}$  and  $\mathbf{m}$ , the highlighted step is extremely slow if  $d_i = 1$

28

---

---

---


---

---

---

---

---

 **Countermeasures to Timing Attacks** WILLIAM & MARY

1. Delay the result if the computation is too fast
  - disadvantage: ?
2. Add a random delay
  - disadvantage?
3. **Blinding**: multiply the ciphertext by a random number before performing decryption

29

---

---

---


---

---

---

---

---

 **RSA's Blinding Algorithm** WILLIAM & MARY

- To confound timing attacks during decryption
  1. generate a random number  $r$  between 0 and  $n-1$  such that  $\text{gcd}(r, n) = 1$
  2. compute  $\mathbf{c}' = \mathbf{c} * r^e \text{ mod } n$
  3. compute  $\mathbf{m}' = (\mathbf{c}')^d \text{ mod } n$  this is where timing attack would occur
  4. compute  $\mathbf{m} = \mathbf{m}' * r^{-1} \text{ mod } n$
- Attacker will not know what the bits of  $\mathbf{c}'$  are
- Performance penalty: < 10% slowdown in decryption speed

30

---

---

---

---

---

---

---

---

**File Encryption and Authentication**

- Alice sends a large file to Bob without disclosing the content of the file to anybody else.
- Also make sure no other people can modify the message without being noticed.
- Conditions:
  - No secret key shared between Alice and Bob.
  - Alice and Bob know each other's RSA public key.  $(SK_A, PK_A)$  and  $(SK_B, PK_B)$

31

---

---

---

---

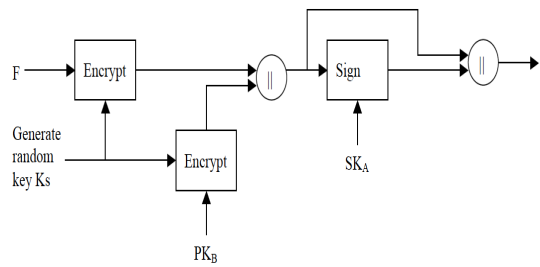
---

---

---

---

**Sender**



$$M = E_{K_s}(F) \parallel E_{PK_B}(K_s) \parallel \text{Sig}_{SK_A}(E_{K_s}(F) \parallel E_{PK_B}(K_s))$$

32

---

---

---

---

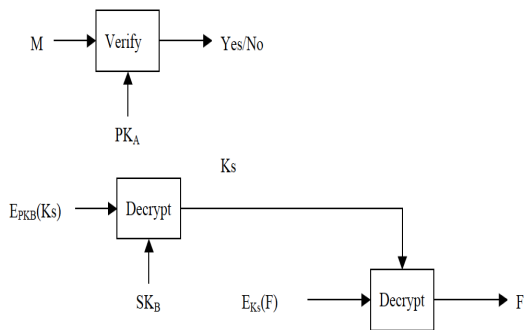
---

---

---

---

**Receiver**



33

---

---

---

---

---

---

---

---

WILLIAM & MARY

## Diffie-Hellman Key Exchange

---

---

---

---

---

---

---

---

WILLIAM & MARY

## Diffie-Hellman Protocol

- For negotiating a shared secret key using only public communication
- Does **not** provide authentication of communicating parties
- What's involved?
  - $p$  is a large prime number (about 512 bits)
  - $g$  is a **primitive root** of  $p$ , and  $g < p$
  - $p$  and  $g$  are **publicly known**

35

---

---

---

---

---

---

---

---

WILLIAM & MARY

## D-H Key Exchange Protocol

Alice	Bob
Publishes or sends $g$ and $p$	Reads $g$ and $p$
Picks random number $S_A$ <i>(and keeps private)</i>	Picks random number $S_B$ <i>(and keeps private)</i>
Computes public key $T_A = g^{S_A} \text{ mod } p$	Computes public key $T_B = g^{S_B} \text{ mod } p$
Sends $T_A$ to Bob, reads $T_B$ from Bob	Sends $T_B$ to Alice, reads $T_A$ from Alice
Computes $T_B^{S_A} \text{ mod } p$	Computes $T_A^{S_B} \text{ mod } p$
<b>=</b>	

36

---

---

---

---

---

---

---

---



## Key Exchange (Cont'd)

WILLIAM & MARY

• Alice and Bob have now both computed **the same secret**  $g^{S_A S_B} \bmod p$ , which can then be used as the **shared secret key K**

•  $S_A$  is the discrete logarithm of  $g^{S_A} \bmod p$  and  $S_B$  is the discrete logarithm of  $g^{S_B} \bmod p$

37

---

---

---

---

---

---

---

---



## D-H Example

WILLIAM & MARY

- Let  $p = 353, g = 3$
- Let random numbers be  $S_A = 97, S_B = 233$
- Alice computes  $T_A = \_\_ \bmod \_\_ = 40 = g^{S_A} \bmod p$
- Bob computes  $T_B = \_\_ \bmod \_\_ = 248 = g^{S_B} \bmod p$
- They exchange  $T_A$  and  $T_B$
- Alice computes  $K = \_\_ \bmod \_\_ = \mathbf{160} = T_B^{S_A} \bmod p$
- Bob computes  $K = \_\_ \bmod \_\_ = \mathbf{160} = T_A^{S_B} \bmod p$

38

---

---

---

---

---

---

---

---



## D-H Example

WILLIAM & MARY

- Let  $p = 353, g = 3$
- Let random numbers be  $S_A = 97, S_B = 233$
- Alice computes  $T_A = 3^{97} \bmod 353 = 40 = g^{S_A} \bmod p$
- Bob computes  $T_B = 3^{233} \bmod 353 = 248 = g^{S_B} \bmod p$
- They exchange  $T_A$  and  $T_B$
- Alice computes  $K = 248^{97} \bmod 353 = \mathbf{160} = T_B^{S_A} \bmod p$
- Bob computes  $K = 40^{233} \bmod 353 = \mathbf{160} = T_A^{S_B} \bmod p$

39

---

---

---

---

---

---

---

---



## Why is This Secure?

WILLIAM & MARY

- Discrete log problem:
  - given  $T_A (= g^{S_A} \text{ mod } p)$ ,  $g$ , and  $p$ , it is **computationally infeasible** to compute  $S_A$
  - (note: as always, to the best of our knowledge; doesn't mean there isn't a method out there waiting to be found)
  - same statement can be made for  $T_B$ ,  $g$ ,  $p$ , and  $S_B$

40

---

---

---

---

---

---

---

---



## D-H Limitations

WILLIAM & MARY

- Expensive exponential operation is required
  - possible timing attacks??
- Algorithm is useful for **key negotiation only**
  - i.e., not for public key encryption
- Not** for user authentication
  - In fact, you can negotiate a key with a complete stranger!

41

---

---

---

---

---

---

---

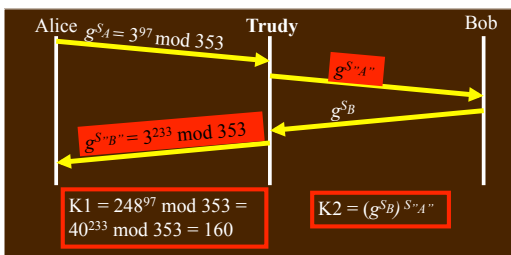
---



## Man-In-The-Middle Attack

WILLIAM & MARY

- Trudy impersonates as Alice to Bob, and also impersonates as Bob to Alice



42

---

---

---

---

---

---

---

---



## MITM Attack (Cont'd)

WILLIAM & MARY

- Now, Alice thinks K1 is the shared key, and Bob thinks K2 is the shared key
- Trudy intercepts messages from Alice to Bob, and
  - decrypts (using K1), substitutes her own message, and encrypts for Bob (using K2)
  - likewise, intercepts and substitutes messages from Bob to Alice
- Solution???

43

---

---

---

---

---

---

---

---



## Authenticating D-H Messages

WILLIAM & MARY

- That is, you know who you're negotiating with, and that the messages haven't been modified
- Requires that communicating parties **already** share some kind of a secret
- Then use encryption, or a MAC (based on this previously-shared secret), of the D-H messages

44

---

---

---

---

---

---

---

---



## Using D-H in "Phone Book" Mode

WILLIAM & MARY

1. Alice and Bob each choose a **semi-permanent** secret number, generate  $T_A$  and  $T_B$
2. Alice and Bob **publish**  $T_A$ ,  $T_B$ , i.e., Alice can get Bob's  $T_B$  at any time, Bob can get Alice's  $T_A$  at any time
3. Alice and Bob can then generate a semi-permanent shared key without communicating
  - but, they must be using the **same  $p$  and  $g$**
  - Essential requirement: **reliability** of the published values (no one can substitute false values)
  - how accomplished???

45

---

---

---

---

---

---

---

---



## Encryption Using D-H?

WILLIAM & MARY

- How to do key distribution + message encryption **in one step**
- Everyone computes and **publishes** their own individual  $\langle p_i, g_i, T_i \rangle$ , where  $T_i = g_i^{S_i} \text{ mod } p_i$
- For Alice to communicate with Bob...
  - Alice picks a random secret  $S_A$
  - Alice computes  $g_B^{S_A} \text{ mod } p_B$
  - Alice uses  $K_{AB} = T_B^{S_A} \text{ mod } p_B$  to encrypt the message
  - Alice sends encrypted message **along with** (unencrypted)  $g_B^{S_A} \text{ mod } p_B$

46

---

---

---

---

---

---

---

---



## Encryption (Cont'd)

WILLIAM & MARY

- For Bob to decipher the encrypted message from Alice
  - Bob computes  $K_{AB} = (g_B^{S_A})^{S_B} \text{ mod } p_B$
  - Bob decrypts message using  $K_{AB}$

47

---

---

---

---

---

---

---

---



## Example

WILLIAM & MARY

- Bob publishes  $\langle p_B, g_B, T_B \rangle = \langle 401, 5, 51 \rangle$  and keeps secret  $S_B = 58$
- Steps
  - Alice picks a random secret  $S_A = 17$
  - Alice computes  $g_B^{S_A} \text{ mod } p_B = \underline{\quad} \text{ mod } \underline{\quad} = 173$
  - Alice uses  $K_{AB} = T_B^{S_A} \text{ mod } p_B = \underline{\quad} \text{ mod } \underline{\quad} = \mathbf{360}$  to encrypt message M
  - Alice sends encrypted message along with (unencrypted)  $g_B^{S_A} \text{ mod } p_B = 173$
  - Bob computes  $K_{AB} = (g_B^{S_A})^{S_B} \text{ mod } p_B = \underline{\quad} \text{ mod } \underline{\quad} = \mathbf{360}$
  - Bob decrypts message M using  $K_{AB}$

48

---

---

---

---

---

---

---

---



WILLIAM & MARY

## Example

- Bob publishes  $\langle p_B, g_B, T_B \rangle = \langle 401, 5, 51 \rangle$  and keeps secret  $S_B = 58$
- Steps
  1. Alice picks a random secret  $S_A = 17$
  2. Alice computes  $g_B^{S_A} \bmod p_B = 5^{17} \bmod 401 = 173$
  3. Alice uses  $K_{AB} = T_B^{S_A} \bmod p_B = 51^{17} \bmod 401 = \mathbf{360}$  to encrypt message M
  4. Alice sends encrypted message along with (unencrypted)  $g_B^{S_A} \bmod p_B = 173$
  5. Bob computes  $K_{AB} = (g_B^{S_A})^{S_B} \bmod p_B = 173^{58} \bmod 401 = \mathbf{360}$
  6. Bob decrypts message M using  $K_{AB}$

49

---

---

---

---

---

---

---

---

WILLIAM & MARY

## Picking $g$ and $p$

- Advisable to change  $g$  and  $p$  periodically
  - the longer they are used, the more info available to an attacker
- Advisable **not** to use **same**  $g$  and  $p$  for everybody
- For "obscure mathematical reasons"...
  - $(p-1)/2$  should be prime
  - $g^{(p-1)/2}$  should be  $\equiv -1 \pmod p$

50

---

---

---

---

---

---

---

---

WILLIAM & MARY

## Digital Signature Standard (DSS)

---

---

---


---

---

---

---

---

 **Digital Signature Standard (DSS)** WILLIAM & MARY

- Useful only for digital signing (no encryption or key exchange)
- Components
  - SHA-1** to generate a hash value (some other hash functions also allowed now)
  - Digital Signature Algorithm (DSA)** to generate the digital signature from this hash value
- Designed to be **fast** for the **signer** rather than verifier
  - e.g., for use in *smart cards*

52

---

---

---


---

---

---

---

---

 **Digital Signature Algorithm (DSA)** WILLIAM & MARY

- Announce public parameters used for signing
  - pick  $p$  (a prime with  $\geq 1024$  bits) ex.:  $p = 103$
  - pick  $q$  (a 160 bit prime) such that  $q|(p-1)$ 

ex.:  $q = 17$  (divides 102)
  - choose  $g = h^{(p-1)/q} \bmod p$ , where  $1 < h < (p-1)$ , such that  $g > 1$ 

ex.: if  $h = 2, g = 2^6 \bmod 103 = 64$
  - note:  $g$  is of **order**  $q \bmod p$ 

ex.: powers of 64 mod 103 =  
 64 79 9 61 93 81 34 13 8 100 14 72 76 23 30 66 1

17 values

53

---

---

---


---

---

---

---

---

 **DSA (Cont'd)** WILLIAM & MARY

- User Alice generates a long-term private key  $x_M$ 
  - random integer with  $0 < x_M < q$  ex.:  $x_M = 13$
- Alice generates a long-term public key  $y_M$ 
  - $y_M = g^{x_M} \bmod p$ 

ex.:  $y_M = 64^{13} \bmod 103 = 76$

54

---

---

---


---

---

---

---

---

 **DSA (Cont'd)** WILLIAM & MARY

ex.:  $p = 103, q = 17, g = 64, x_M = 13, Y_M = 76$

- Alice randomly picks a private key  $k$  such that  $0 < k < q$ , and generates  $k^{-1} \bmod q$   
 ex.:  $k = 12, 12^{-1} \bmod 17 = 10$
- Signing message  $M$  ex.:  $H(M) = 75$ 
  - public key  $r = (g^k \bmod p) \bmod q$   
 ex.:  $r = (64^{12} \bmod 103) \bmod 17 = 4$
  - signature  $s = [k^{-1}(H(M) + x_M r)] \bmod q$   
 ex.:  $s = [10 * (75 + 13 * 4)] \bmod 17 = 12$
  - transmitted info =  $M, r, s$   
 ex.:  $M, 4, 12$

55

---

---

---


---

---

---

---

---

 **Verifying a DSA Signature** WILLIAM & MARY

ex.:  $p = 103, q = 17, g = 64, x_M = 76, H(M) = 75$

- Known :  $g, p, q, Y_M$
- Received from signer:  $M, r, s$  ex.:  $M, 4, 12$

- $w = (s)^{-1} \bmod q$  ex.:  $w = 12^{-1} \bmod 17 = 10$
- $u_1 = [H(M)w] \bmod q$  ex.:  $u_1 = 75 * 10 \bmod 17 = 2$
- $u_2 = (r * w) \bmod q$  ex.:  $u_2 = 4 * 10 \bmod 17 = 6$
- $v = [(g^{u_1} * Y_M^{u_2}) \bmod p] \bmod q$   
 ex.:  $v = [(64^2 * 76^6) \bmod 103] \bmod 17 = 4$
- If  $v = r$ , then the signature is verified

56

---

---

---


---

---

---

---

---

 **Verifying DSA Signature** WILLIAM & MARY

- Received:  $M, r=13, s=24$

- $w = (s)^{-1} \bmod q = 24$
- $u_1 = [H(M)w] \bmod q = 22 * 24 \bmod 25 = 3$
- $u_2 = (r)w \bmod q = 13 * 24 \bmod 25 = 12$
- $v = [(g^{u_1} Y_A^{u_2}) \bmod p] \bmod q = [5^3 * 56^{12} \bmod 101] \bmod 25 = 13$
- If  $v = r$ , then the signature is verified

57

---

---

---

---

---

---

---

---



## Why Does it Work?

WILLIAM & MARY

- Correct? The signer computes
  - $s = k^{-1} * (H(m) + x*r) \text{ mod } q$
  - so  $k \equiv H(m)*s^{-1} + x*r*s^{-1}$
  - $\equiv H(m)*w + x*r*w \text{ mod } q$
- Since  $g$  has order  $q$ :
  - $g^k \equiv g^{H(m)w} * g^{xrw}$
  - $\equiv g^{H(m)w} * y^{rw}$
  - $\equiv g^{u1} * y^{u2} \text{ mod } p$ , and
- $r = (g^k \text{ mod } p) \text{ mod } q = (g^{u1} * y^{u2} \text{ mod } p) \text{ mod } q = v$

58

---

---

---

---

---

---

---

---



## Is it Secure?

WILLIAM & MARY

- Given  $y_M$ , it is difficult to compute  $x_M$ 
  - $x_M$  is the discrete log of  $y_M$  to the base  $g$ , mod  $p$
- Likewise, given  $r$ , it is difficult to compute  $k$
- Cannot forge a signature without  $x_M$
- Signatures are not repeated (only used once per message) and cannot be replayed

59

---

---

---

---

---

---

---

---



## Assessment of DSA

WILLIAM & MARY

- Slower to verify than RSA, but faster signing than RSA
- Key lengths of 2048 bits and greater are also allowed

60

---

---

---

---

---

---

---

---