Digital Signatures

CS 5158/6058 Data Security and Privacy

Spring 2018

Instructor: Boyang Wang

- Diffie-Hellman Key Exchange
- Alice outputs public parameters (G, p, g)
    - G is cyclic, generator g, group order p

Alice		Bob
choose a uniform $x \in \mathbb{Z}_p$, compute $h_A = g^x$	$\xrightarrow{\quad h_A \quad}$	choose a uniform $y \in \mathbb{Z}_p$, compute $h_B = g^y$
	$\xleftarrow[\quad h_B \quad]{}$	
compute $k_A = h_B^x = g^{xy}$		compute $k_B = h_A^y = g^{xy}$

- Alice and Bob share a same key g^{xy}

Security of DH Protocol

- If Discrete-Logarithm problem is easy, DH is not secure
  - Eavesdropper has $h_A = g^x$ and $h_B = g^y$
  - Computes $x = \log_g(h_A)$ and $y = \log_g(h_B)$
  - Obtains key $k = g^{xy}$
  - DL is hard is necessary, but not sufficient

- Computational Diffie-Hellman Problem (CDH)
  - Given g^x and g^y, compute g^{xy} is hard

Security of DH Protocol

- If CDH problem is easy, DH is not secure
  - Given $h_A = g^x$ and $h_B = g^y$
  - Adversary computes key $k = g^{xy}$
  - CDH is hard is necessary, but still not sufficient

- Decisional Diffie-Hellman Problem (DDH)
  - Given g^x, g^y and a random element h in G, decide whether $h ?= g^{xy}$ is hard
- DH protocol is secure if DDH problem is hard

Security of DH Protocol

- Computational Diffie-Hellman Problem (CDH)
  - Given g^x and g^y, compute g^{xy} is hard
- Decisional Diffie-Hellman Problem (DDH)
  - Given g^x, g^y and a random element h in G, decide whether $h ?= g^{xy}$ is hard

- True: DH protocol is secure if DDH problem is hard
- False: DH protocol is secure if CDH problem is hard
- False: DH protocol is secure if DL problem is hard

Man-In-The-Middle Attacks

- DH is secure only against an eavesdropper

- DH is not secure under man-in-the-middle attacks
  - An attacker "Eve" between Alice and Bob
  - Alice thinks "I am talking to Bob", but is Eve
  - Bob thinks "I am talking to Alice", but is Eve

Alice

Eve

Bob

- Alice thinks “I am talking to Bob”, but is Eve
- Public parameters (G, p, g)
  - G is cyclic, generator g, group order p

Alice		(Eve)
choose a uniform $x \in \mathbb{Z}_p$, compute $h_A = g^x$	$\xrightarrow{h_A}$	choose a uniform $z \in \mathbb{Z}_p$, compute $h_B = g^z$
	$\xleftarrow{h_B}$	
compute $k_A = h_B^x = g^{xz}$		compute $k_B = h_A^z = g^{xz}$

- Alice and Eve share a same key g^{xz}

- Bob thinks "I am talking to Alice", but is Eve
- Public parameters (G, p, g)
  - generator g, group order p

(Eve)		Bob
choose a uniform $z \in \mathbb{Z}_p$, compute $h_A = g^z$	$\xrightarrow{\quad h_A \quad}$	choose a uniform $y \in \mathbb{Z}_p$, compute $h_B = g^y$
	$\xleftarrow[\quad h_B \quad]{}$	
compute $k_A = h_B^z = g^{yz}$		compute $k_B = h_A^y = g^{yz}$

- Eve and Bob share a same key g^{yz}

Man-In-The-Middle Attacks

- Alice thinks “I share a key with Bob”, but is Eve
- Bob thinks “I share a key with Alice”, but is Eve
- Eve has two keys, one with Alice, one with Bob
  - Eve learns all the later messages between A & B

Alice

$$k = g^{xz}$$

Eve

$$k = g^{xz}$$

$$k' = g^{yz}$$

Bob

$$k' = g^{yz}$$

Alice

$$k = g^{xz}$$

Eve

$$k = g^{xz}$$

$$k' = g^{yz}$$

Bob

$$k' = g^{yz}$$

- Example: $Z_7^* = \{1, 2, 3, 4, 5, 6\}$
  - Group order p = 6, generator g = 3
  - Alice: x = 3, Bob: y = 2, Eve: z = 5
  - What is the correct key between Alice and Bob without man-in-the-middle attack?
  - What is k and k' in man-in-the-middle attack?

Alice	Eve	Bob
		
$k = g^{xz}$	$k = g^{xz}$	
	$k' = g^{yz}$	$k' = g^{yz}$

- <u>Example</u>: $Z_7^* = \{1, 2, 3, 4, 5, 6\}$
  - Group order p = 6, generator g = 3
  - Alice: x = 3, Bob: y = 2, Eve: z = 5
  - A<—>B correct key: $g^{xy} = 3^6 = 1 \bmod 7$
  - Alice has $k = g^{xz} = 3^{15} = 6 \bmod 7$
  - Bob has $k' = g^{yz} = 3^{10} = 4 \bmod 7$

Alice | Eve | Bob

Alice: $k = g^{xz}$

Eve: $k = g^{xz}$, $k' = g^{yz}$

Bob: $k' = g^{yz}$

- <u>Practice</u>: $Z_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$
  - Group order p = 10, generator g = 2
  - Alice: x = 3, Bob: y = 2, Eve: z = 5
  - What is the correct key between Alice and Bob without man-in-the-middle attack?
  - What is k and k' in man-in-the-middle attack?

Alice

$$k = g^{xz}$$

Eve

$$k = g^{xz}$$

$$k' = g^{yz}$$

Bob

$$k' = g^{yz}$$

- <u>Practice</u>: $Z_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$
  - Group order p = 10, generator g = 2
  - Alice: x = 3, Bob: y = 2, Eve: z = 5
  - A<—>B correct key: $g^{xy} = 2^6 = 9 \bmod 11$
  - Alice has $k = g^{xz} = 2^{15} = 10 \bmod 11$
  - Bob has $k' = g^{yz} = 2^{10} = 1 \bmod 11$

Man-In-The-Middle Attacks

- DH is not secure under man-in-the-middle attacks
  - Alice thinks “I am talking to Bob”, but is Eve
  - Bob thinks “I am talking to Alice”, but is Eve
  - The channels are <u>not authenticated</u>

Alice

$k = g^{xz}$

Eve

$k = g^{xz}$

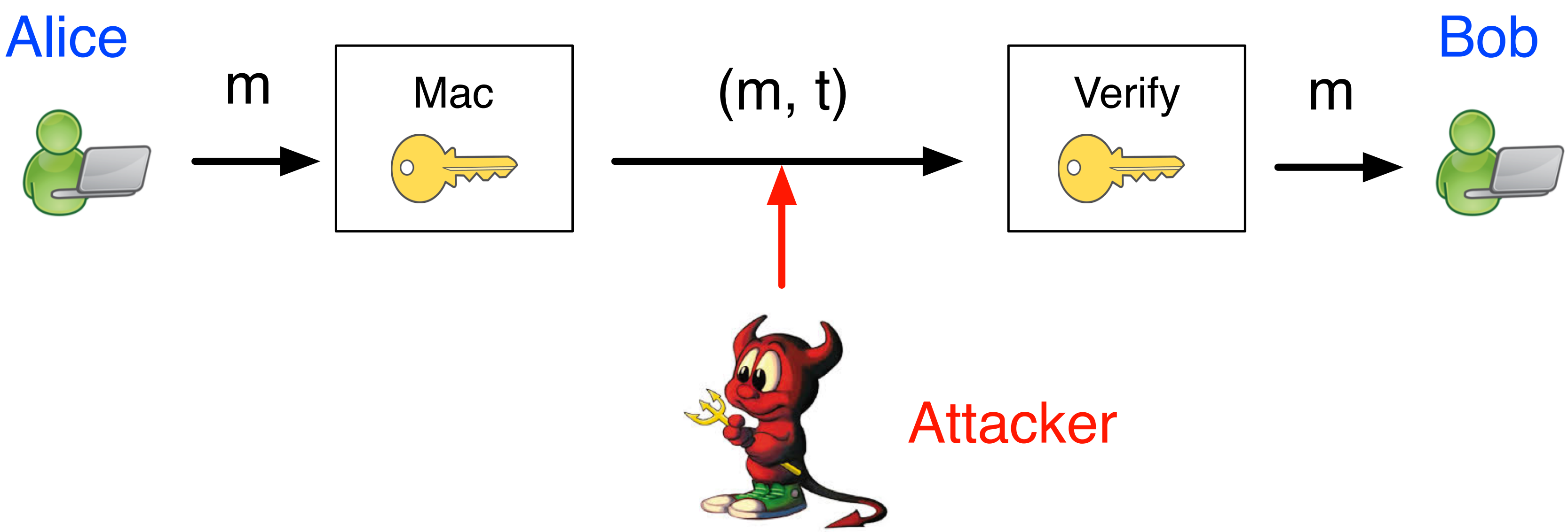
$k' = g^{yz}$

Bob

$k' = g^{yz}$

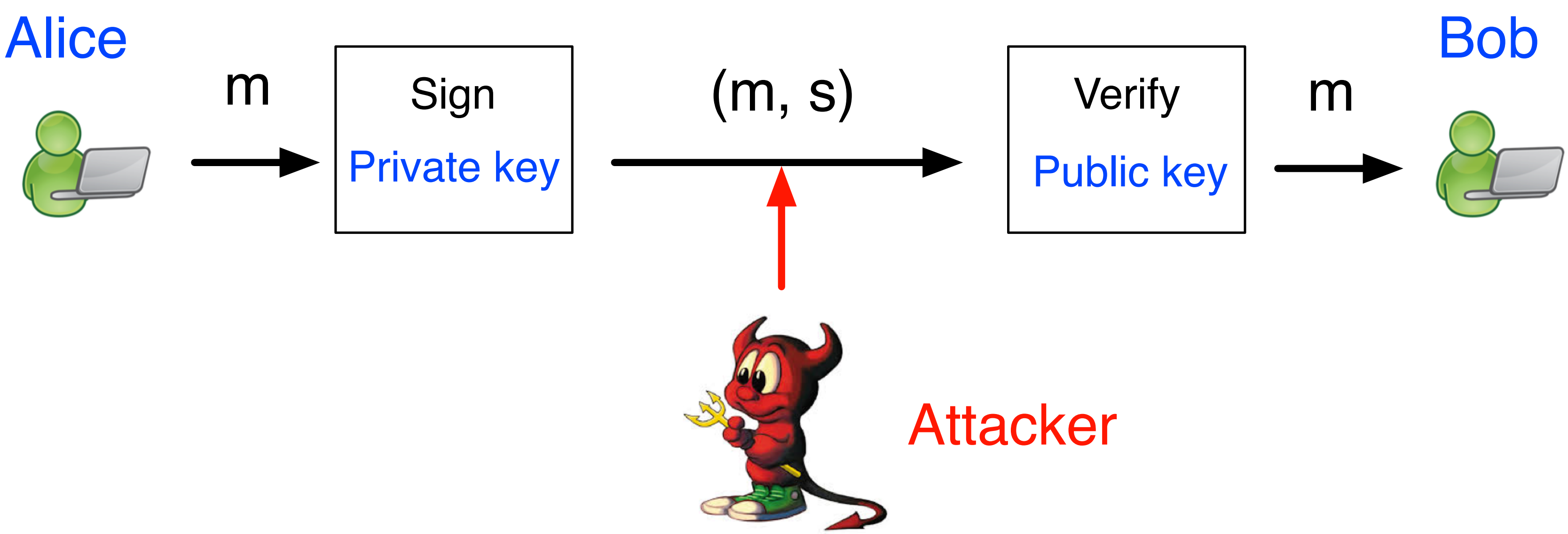
Message Authentication Code

- Alice computes a tag for a message
- Bob can verify a message m using its tag t
  - If valid, accept a message
  - Otherwise, drop or ignore a message



Digital Signatures

- Public-key: Digital Signatures
  - Alice computes a signature with her sk
  - Bob can verify a message with Alice's pk



Signature Scheme

1. KeyGen: takes a security parameter 1^n, outputs a pair of keys (pk, sk).
2. Sign: takes a private key sk and a message m, outputs a signature s
3. Verify: takes a public key pk, a message m, and a signature s, outputs 1 if valid and 0 invalid

Assumptions on Adversary

- Assumptions on an <u>adversary</u>:
  - Knows messages (is not about privacy)
  - Knows Sign and Verify algorithm,
  - Can collect previous message-sig pairs
  - Knows public key, but not private key

  - Has <u>limited</u> computational power
    - Run efficient (polynomial-time) algorithms

Security of Signatures

- Unforgeable under chosen-message attacks
  - A PPT adversary has access to a Sign oracle
    - Submits a message m, obtains a valid signature s
  - It is hard for adversary to generate a valid signature s’ for a “new” message m’
    - A message that was not submitted to oracle
    - Valid if Verify(m’, s’) = 1
    - Happens with a negligible probability

Security of Signature

- <u>Example</u>: A sent 3 messages with signatures to B
    - (`packers, 0x34dt`)
    - (`patriots,0xd5ac`)
    - (`eagles,  0xa70b`)

- Adversary learns above 3 message-sig pairs, sends
    - (`patriots,0xd5ac`) to Bob, Bob takes it
    - (`patriots,0xd5ab`) to Bob, Bob drops it
    - (`bengals,  0x1234`) to Bob, Bob drops it

Signatures v.s. MACs

- Both protect data integrity

- Signatures have more properties:
  - Public verifiable: everyone can verify (everyone can have a public key)
  - Non-repudiation: Alice cannot deny a message signed by herself. (only Alice has her private key)

- Both suffer replay attacks

Textbook RSA Signature

- KeyGen: given a security parameter 1^n, generate two n-bit primes p, q, compute $N = pq$, choose e s.t. $\gcd(e, \phi(N)) = 1$, compute $d = e^{-1} \mod \phi(N)$, output public key $pk = (N, e)$, private key $sk = d$

- Sign: given a message m and a private key $sk = d$, return $s = m^d \mod N$

- Verify: given a public key $pk = (N, e)$, a message m and a signature s, return 1 iff $m = s^e \mod N$

Textbook RSA Signature

- Correctness:
  - Number theory basic: $a^{\phi(N)} = 1 \mod N$
  - We know in RSA: $ed = 1 \mod \phi(N)$

$$\begin{aligned} s^e &= (m^d)^e \\ &= m^{de \mod \phi(N)} \\ &= m \mod N \end{aligned}$$

- m, s are elements in group Z_N^*
- e, d are integers

Textbook RSA Signature

- Example: $p*q = 11*7 = 77 = N$
  - $\phi(N) = |Z_N^*| = (p-1)(q-1) = 10*6 = 60$
  - choose e, s.t. $\gcd(e, \phi(N)) = 1$
  - $e = 7$, $d = 43$, $ed = 7*43 = 1 \bmod 60$
  - $pk = (N, e) = (77, 7)$, $sk = d = 43$

  - Given $m=2$, signature $s=m^d=2^{43}=30 \bmod 77$
    - $(m, s) = (2, 30)$
  - Given $m=2$, $s=30$, pk, $s^e = 30^7 = 2 \bmod 77 = m$

Textbook RSA Signature

- Textbook RSA: pk = (N, e) = (77, 7), sk = d = 43
  - Sign: $s=m^d$; Verify: $m?=s^e$

- <u>Practice:</u> Plaintext RSA
  - Give (m, s) = (3, 38), is s a valid signature?
  - Give (m, s) = (4, 50), is s a valid signature?

  - $s^e = 38^7 \bmod 77 = 3 == m$; Valid
  - $s^e = 50^7 \bmod 77 = 8 \mathrel{!=} m$; Not valid

Textbook RSA Signature

- “Inverse” of Textbook RSA Encryption
  - Sign same as Dec, Verify (almost) same as Enc

- No-Message Attack
  - Given a public key pk = (N, e) only
  - Choose a random s’ from Z_N^*
  - Output a message m’ = $s'^e \bmod N$
  - s’ is a valid signature of m’
    - No control on m’, could be meaningless

Textbook RSA Signature

- Textbook RSA: pk = (N, e) = (77, 7), sk = d = 43

- Example: No-Message Attack
  - generate a random s' = 20
  - compute m' = $s'^e = 20^7 = 48 \bmod 77$
  - output (m', s') = (48, 20),
    - valid signature for a 'new' message

- Attacker has no control on message m'

Textbook RSA Signature

- Homomorphic Attack
  - Obtain (m_1, s_1) and (m_2, s_2)
  - Compute m’ = m_1*m_2
  - Compute s’ = s_1*s_2
  - Output (m’, s’), s’ is valid, but m’ is ‘new’

$$\text{Given } s_1 = m_1^d, s_2 = m_2^d$$

$$\begin{aligned} s'^e &= (s_1 \cdot s_2)^e = (m_1^d \cdot m_2^d)^e \\ &= m_1^{ed} \cdot m_2^{ed} = m_1 \cdot m_2 \\ &= m' \mod N \end{aligned}$$

Textbook RSA Signature

- Textbook: RSA pk = (N, e) = (77, 7), sk = d = 43

- Example: Homomorphic Attack
  - Given $(m_1, s_1) = (2, 30)$ and $(m_2, s_2) = (3, 38)$
  - $m' = m_1 * m_2 = 2*3 = 6$
  - $s' = s_1 * s_2 = 30*38 = 62 \bmod 77$
  - output $(m', s') = (6, 62)$
    - s' is valid, but m' is a 'new' message
- Attacker has control on message m'

Textbook RSA Signature

- Textbook: RSA pk = (N, e) = (77, 7), sk = d = 43

- Practice: Homomorphic Attack
  - Given $(m_1, s_1) = (4, 53)$ and $(m_2, s_2) = (3, 26)$
  - output (m’, s’) = (??, ??)
    - s’ is valid, but m’ is a ‘new’ message

  - $m' = m_1 * m_2 = 4*3 = 12$
  - $s' = s_1 * s_2 = 53*26 = 69 \bmod 77$

Textbook RSA Signature

- Homomorphic Attack
  - Obtain (m_1, s_1)
  - Compute m' = m_1*m_1
  - Compute s' = s_1*s_1
  - Output (m', s'), s' is valid, but m' is 'new'

Given $s_1 = m_1^d$

$$\begin{aligned} s'^e &= (s_1 \cdot s_1)^e = (m_1^d \cdot m_1^d)^e \\ &= m_1^{ed} \cdot m_1^{ed} = m_1 \cdot m_1 \\ &= m' \mod N \end{aligned}$$

Textbook RSA Signature

- Textbook: RSA pk = $(N, e) = (77, 7)$, sk = $d = 43$

- Practice: Homomorphic Attack
  - Given $(m_1, s_1) = (4, 53)$
  - output $(m', s') = (??, ??)$
    - s' is valid, but m' is a 'new' message

  - $m' = m_1 * m_1 = 4*4 = 16$
  - $s' = s_1 * s_1 = 53*53 = 37 \bmod 77$

RSA-FDH

- RSA with Full-Domain Hash (e.g., SHA256)
  - Hash a message, then sign hash value with RSA
- KeyGen: Same
- Sign: given a message m and a private key $sk = d$, return $s = H(m)^d \mod N$
- Verify: given a public key $pk = (N, e)$, a message m and a signature s, return 1 iff $H(m) = s^e \mod N$

Security of RSA-FDH

- RSA-FDH prevents No-Message Attack
  - Given a public key pk = (N, e) only
  - Choose a random s' from Z_N^*
  - Compute $s'^e = H(m') \bmod N$
  - But hard to obtain m' since H is hard to inverse
    - Based on H(m'), find m' (a collision) with a negligible probability

Security of RSA-FDH

- RSA-FDH prevents Homomorphic Attack
  - Obtain (m_1, s_1) and (m_2, s_2)
  - Compute $s' = s_1 * s_2$, then $s'^e = H(m_1)H(m_2)$
  - Hard to find an m', s.t. $H(m') == H(m_1)H(m_2)$
    - find m' with a negligible probability

Given $s_1 = H(m_1)^d, s_2 = H(m_2)^d$

$$\begin{aligned} s'^e &= (s_1 \cdot s_2)^e = (H(m_1)^d \cdot H(m_2)^d)^e \\ &= H(m_1)^{ed} \cdot H(m_2)^{ed} = H(m_1) \cdot H(m_2) \\ &\stackrel{?}{=} H(m') \mod N \end{aligned}$$

Public-Key Infrastructure

- Alice's public key
    - pk = 0x42eacb567812 30e0e23….
- How can Bob ensure pk is Alice's public key

- Public-Key Infrastructure (PKI)
    - Certificate Authority (CA) generates certificates;
    - A certificate has the name of a user and the public key of this user
    - A certificate binds a user and a pk together

Public-Key Infrastructure

- CA has its own public key and private key
  - CA signs each certificate with its private key
  - Users use CA's public key to verify

- E.g., Alice and Bob want to talk
  - Alice shows Bob her certificate
    - Bob verifies Alice's certificate with CA's pk
  - Bob shows Alice his certificate
    - Alice verifies Bob's certificate with CA's pk
  - Alice and Bob confirm each other's pk

Public-Key Infrastructure

- Certificate Authority (UC); certificate (bearcat card)
  - UC generates bearcat cards;
  - A bearcat card has your name and M number
  - A bearcat card binds your name with M number
  - Everyone trusts UC

  - Alice shows her card to Bob;
  - Bob shows his card to Alice;
  - Alice and Bob will start to talk

Subject Name	
Common Name	com.apple.idms.appleid.prd.57306437347276776169424d344f
Issuer Name	
Country	US
Organization	Apple Inc.
Organizational Unit	Apple Certification Authority
Common Name	Apple Application Integration Certification Authority
Serial Number	1957307747427178022
Version	3
Signature Algorithm	SHA-256 with RSA Encryption (1.2.840.113549.1.1.11)
Parameters	none
Not Valid Before	Thursday, July 6, 2017 at 8:30:45 AM Eastern Daylight Time
Not Valid After	Saturday, July 6, 2019 at 8:30:45 AM Eastern Daylight Time
Public Key Info	
Algorithm	RSA Encryption (1.2.840.113549.1.1.1)
Parameters	none
Public Key	256 bytes : E4 A3 B2 A6 5E E4 02 F2 ...
Exponent	65537
Key Size	2048 bits
Key Usage	Encrypt, Verify, Derive
Signature	256 bytes : 7F 7B 2F DD C2 AD CC F6 ...

Additional Reading

Chapter 12, *Introduction to Modern Cryptography, Drs. J. Katz and Y. Lindell, 2nd edition*