

Public-Key Encryption and Digital Signatures

PRNGs: Summary

ITIS 6200 / 8200

- True randomness requires sampling a physical process
 - Slow, expensive, and biased (low entropy)
- PRNG: An algorithm that uses a little bit of true randomness to generate a lot of random-looking output
 - Seed(entropy): Initialize internal state
 - Reseed(entropy): Add additional entropy to the internal state
 - Generate(n): Generate n bits of pseudorandom output
 - Security: Computationally indistinguishable from truly random bits
- HMAC-DRBG: Use repeated applications of HMAC to generate pseudorandom bits
- Application: UUIDs

Summary: Diffie-Hellman Key Exchange

ITIS 6200 / 8200

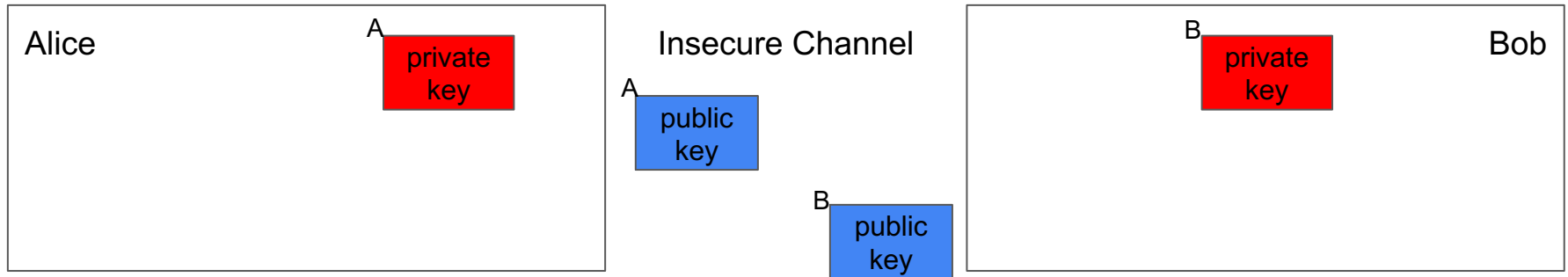
- Algorithm:
 - Alice chooses a and sends $g^a \bmod p$ to Bob
 - Bob chooses b and sends $g^b \bmod p$ to Alice
 - Their shared secret is $(g^a)^b = (g^b)^a = g^{ab} \bmod p$
- Diffie-Hellman provides forwards secrecy: Nothing is saved or can be recorded that can ever recover the key
- Issues
 - *Not* secure against MITM
 - Both parties must be online
 - Does not provide authenticity

Public-Key Cryptography (Asymmetric Key Cryptography)

Public-Key Cryptography

ITIS 6200 / 8200

- A cryptography scheme that both parties in the communication use *different* keys
- In public-key schemes, each person has two keys
 - **Public key**: Known to everybody
 - **Private key**: Only known by that person
 - Keys come in pairs: every public key corresponds to one private key (mathematically related)



Public-Key Cryptography

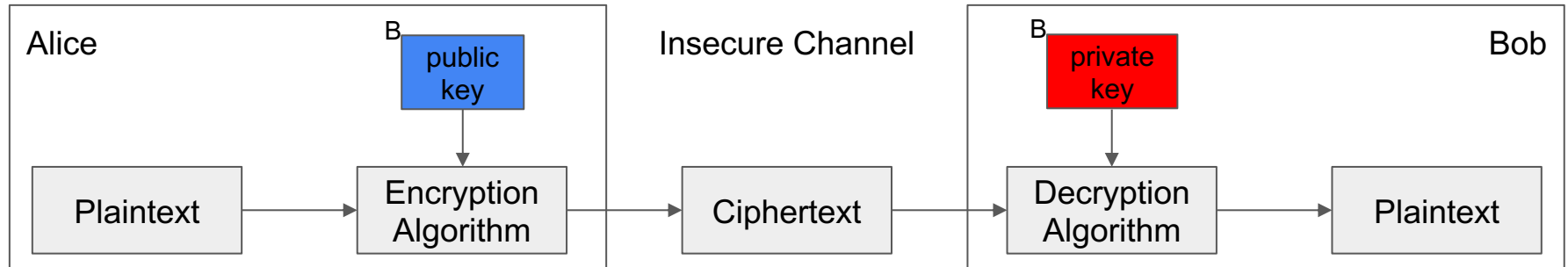
ITIS 6200 / 8200

- Uses number theory
 - Examples: Modular arithmetic, factoring, discrete logarithm problem
 - Contrast with symmetric-key cryptography (uses XORs and bit-shifts)
- Messages are numbers
 - Contrast with symmetric-key cryptography (messages are bit strings)
- Benefit: No longer need to assume that Alice and Bob already share a secret
- Drawback: Much slower than symmetric-key cryptography
 - Number theory calculations are much slower than XORs and bit-shifts

Public-Key Cryptography for Confidentiality

ITIS 6200 / 8200

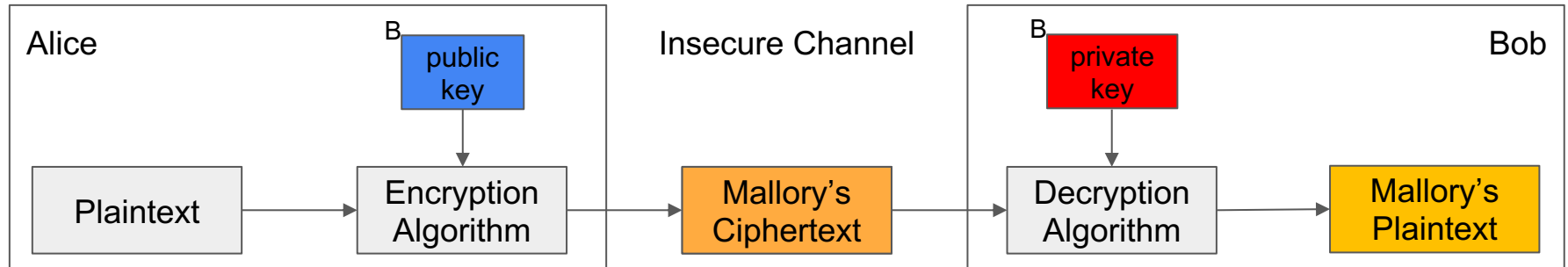
- Scenario
 - Alice wants to send a message to Bob
 - Alice uses Bob's public key to encrypt the message
 - Bob decrypt the message with his private key
- Who can perform the encryption? i.e., send messages to Bob
 - Anyone, because Bob's public key is public
- Who can perform the decryption? i.e., see the message for Bob
 - Only Bob, with his private key



Public-Key Cryptography (MITM)

ITIS 6200 / 8200

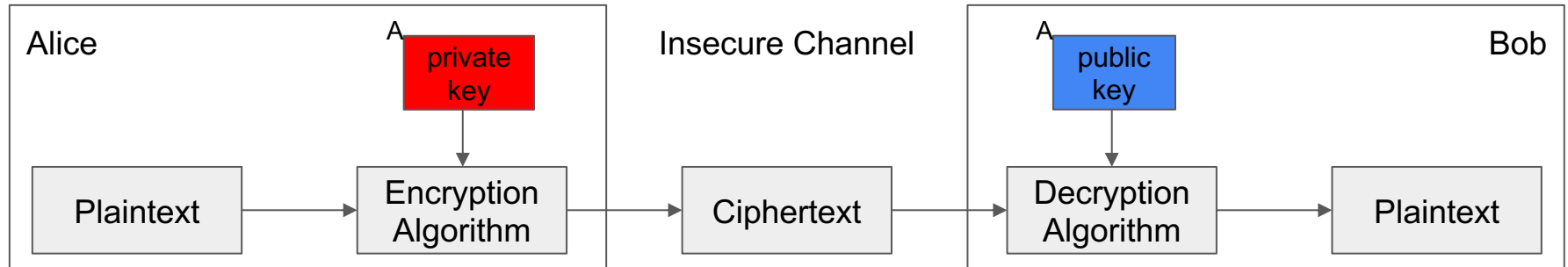
- Scenario
 - Alice wants to send a message to Bob
 - Alice uses Alice's public key to encrypt the message
 - Mallory intercepts the message, changes it into another message encrypted with Bob's public
 - Bob decrypts the message with his private key, cannot tell if it's from Alice



Public-Key Cryptography for Integrity

ITIS 6200 / 8200

- Scenario
 - Alice wants to send a message to Bob
 - Alice uses her private key to encrypt the message
 - Bob decrypts the message with Alice public key
- Who can perform the encryption? i.e. who can produce the message
 - Only Alice, with her private key
- Who can perform the decryption? i.e., who can verify the message
 - Anyone, because Alice's public key is public



Public-Key Cryptography

ITIS 6200 / 8200

- Encryption with the public key, e.g., send message to Alice
 - $C = \text{Enc}(\text{pub-alice}, M)$
 - $M = \text{Dec}(\text{priv-alice}, C)$
- Encryption with the private key, e.g., Alice signs the message
 - $C = \text{Enc}(\text{priv-alice}, M)$
 - $M = \text{Dec}(\text{pub-alice}, C)$
- Two common encryption/decryption pairs, other pairs do not work

Public-Key Encryption

Public-Key Encryption: Definition

ITIS 6200 / 8200

- Three parts:
 - $\text{KeyGen}() \rightarrow PK, SK$: Generate a public/private keypair, where PK is the public key, and SK is the private (secret) key
 - $\text{Enc}(PK, M) \rightarrow C$: Encrypt a plaintext M using public key PK to produce ciphertext C
 - $\text{Dec}(SK, C) \rightarrow M$: Decrypt a ciphertext C using secret key SK
- Properties
 - **Correctness**: Decrypting a ciphertext should result in the message that was originally encrypted
 - $\text{Dec}(SK, \text{Enc}(PK, M)) = M$ for all $PK, SK \leftarrow \text{KeyGen}()$ and M
 - **Efficiency**: Encryption/decryption should be fast
 - **Security**: Similar to IND-CPA, but Alice (the challenger) just gives Eve (the adversary) the public key, and Eve doesn't request encryptions, except for the pair M_0, M_1
 - You don't need to worry about this game (it's called "semantic security")

RSA Encryption

Cryptography Roadmap

ITIS 6200 / 8200

	Symmetric-key	Asymmetric-key
Confidentiality	<ul style="list-style-type: none">● One-time pads● Block ciphers with chaining modes (e.g. AES-CBC)	<ul style="list-style-type: none">● RSA encryption
Integrity, Authentication	<ul style="list-style-type: none">● MACs (e.g. HMAC)	<ul style="list-style-type: none">● Digital signatures (e.g. RSA signatures)

- Hash functions
- Pseudorandom number generators
- Public key exchange (e.g. Diffie-Hellman)

- Key management (certificates)
- Password management

RSA Encryption

ITIS 6200 / 8200

- The first public key cryptosystem
- Invented by Rivest, Shamir, and Adleman
- Any bit size is OK
 - Bit size of the prime numbers used to create public and private keys
 - Different from symmetric key encryption
 - 512 was standard when it was released
 - 2048 or 4096 is standard now
- Based on prime numbers and factoring

RSA Encryption: Definition

ITIS 6200 / 8200

- **KeyGen():**

- Randomly pick two large primes, p and q
 - Done by picking random numbers and then using a test to see if the number is (probably) prime
- Compute $N = pq$
 - N is usually between 2048 bits and 4096 bits long
- Choose e
 - Requirement: e is not a factor of $(p - 1)(q - 1)$
 - Requirement: $2 < e < (p - 1)(q - 1)$
- Compute $d = e^{-1} \bmod (p - 1)(q - 1)$
 - d is the modular multiplicative inverse of e
 - $1 = (d * e) \bmod (p - 1)(q - 1)$
 - Algorithm: Extended Euclid's algorithm
- **Public key:** N and e
- **Private key:** d

- **Example**

- Randomly pick two (large) primes, p and q
 - $p = 3, q = 7$
- Compute $N = pq$
 - $N = p * q = 21$
- Choose e
 - $e = 5$ (not a factor of $2 * 6 = 12$)
- Compute $d = e^{-1} \bmod (p - 1)(q - 1)$
 - $d = 5$ since $(d * e) \bmod (p - 1)(q - 1) = 5 * 5 \bmod (2 * 6) = 1$
- **Public key:** $N=21$ and $e=5$
- **Private key:** $d = 5$

RSA Encryption: Definition

ITIS 6200 / 8200

- $\text{Enc}(e, N, M)$:
 - Output: $M^e \bmod N$
- $\text{Dec}(d, C)$:
 - Output: $C^d \bmod N$
 - $C^d \bmod N = (M^e)^d \bmod N$

Example

- $\text{Enc}(e, N, M)$:
 - Output $M^e \bmod N$
 - $M = 12, e = 5, N = 21$
 - $C = 12^5 \bmod 21 = 3$
- $\text{Dec}(d, C)$:
 - Output $C^d \bmod N$
 - $C = 3, d = 5, N = 21$
 - $M = 3^5 \bmod 21 = 243 \bmod 21 = 12$

RSA Encryption: Correctness

ITIS 6200 / 8200

1. **Theorem:** $M^{ed} \bmod N \equiv M \bmod N$
2. Euler's theorem: $a^{\varphi(N)} \equiv 1 \bmod N$
 - $\varphi(N)$ is the totient function of N
 - If N is prime, $\varphi(N) = N - 1$ (Fermat's little theorem)
 - For a semi-prime pq , where p and q are prime, $\varphi(pq) = (p - 1)(q - 1)$
3. Notice: $e * d \equiv 1 \bmod (p - 1)(q - 1)$ so $ed \equiv 1 \bmod \varphi(N)$
 - This means that $ed = k\varphi(n) + 1$ for some integer k
4. (1) can be written as $M^{k\varphi(N) + 1} \equiv M \bmod N$
5. $M^{k\varphi(N)} M^1 \equiv M \bmod N$
6. $1M^1 \equiv M \bmod N$ by Euler's theorem
7. $M \equiv M \bmod N$

RSA Encryption: Security

ITIS 6200 / 8200

- **RSA problem:** Given N and $C = M^e \bmod N$, it is hard to find M
 - No harder than the factoring problem
 - If you can factor N , you can recover d , because $1 = (d * e) \bmod (p - 1)(q - 1)$, and $N = p * q$
- A brute-force attack is basically trying to factor the public key into two prime numbers
- Current best solution is to factor N , but unknown whether there is an easier way
 - If the RSA problem is as hard as the factoring problem, then the scheme is secure as long as the factoring problem is hard
 - Factoring problem is assumed to be hard, but we have no proof

RSA Encryption: Issues

ITIS 6200 / 8200

- Is RSA encryption IND-CPA secure?
 - No. It's deterministic. No randomness was used at any point!
- Sending the same message encrypted with different public keys also leaks information
 - $m^{e_a} \bmod N_a, m^{e_b} \bmod N_b, m^{e_c} \bmod N_c$
 - Small m and e leaks information
 - e is usually small (~ 16 bits) and often constant (3, 17, 65537)
- Side channel: A poor implementation leaks information
 - The time it takes to decrypt a message depends on the message and the private key
 - This attack has been successfully used to break RSA encryption in OpenSSL
- Result: We need a probabilistic padding scheme

OAEP

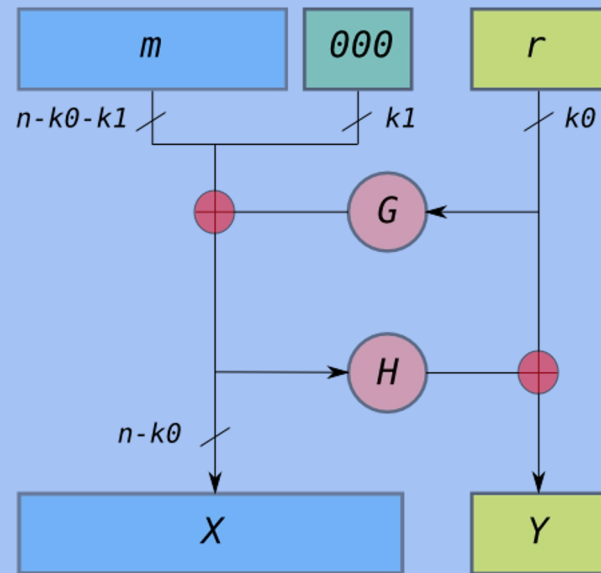
ITIS 6200 / 8200

- **Optimal asymmetric encryption padding (OAEP):** A variation of RSA that introduces randomness
 - Different from “padding” used for symmetric encryption, used to add randomness instead of dummy bytes
- Idea: RSA can only encrypt “random-looking” numbers, so encrypt the message with a random key

OAEP: Padding

ITIS 6200 / 8200

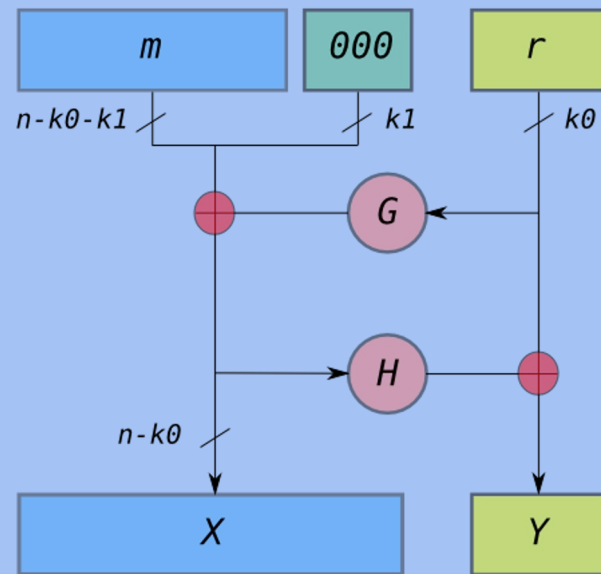
1. k_0 and k_1 constants defined in the standard, and G and H are hash functions
 - M can only be $n - k_0 - k_1$ bits long
 - G produces a $(n - k_0)$ -bit hash, and H produces a k_0 -bit hash
2. Pad M with k_1 0's
 - Idea: We should see 0's here when unpadding, or else someone tampered with the message
3. Generate a random, k_1 -bit string r
4. Compute $X = M || 00\dots0 \oplus G(r)$
5. Compute $Y = r \oplus H(X)$
6. Result: $X || Y$



OAEP: Unpadding

ITIS 6200 / 8200

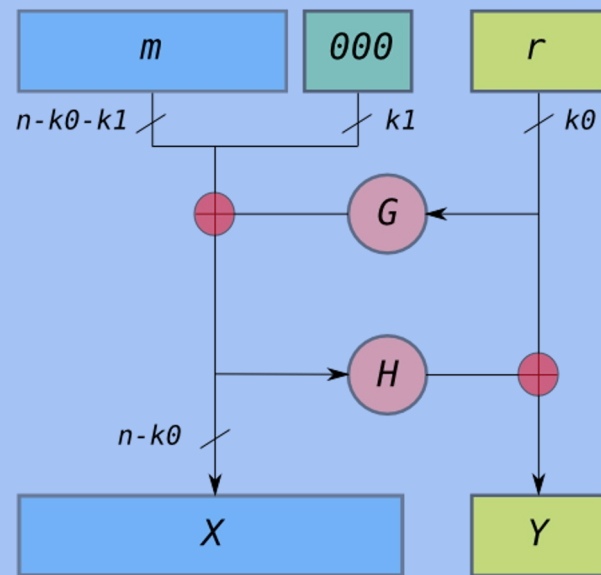
1. Compute $r = Y \oplus H(X)$
2. Compute $M || 00...0 = X \oplus G(r)$
3. Verify that $M || 00...0$ actually ends in k_1 0's
 - Error if not



OAEP

ITIS 6200 / 8200

- Even though G and H are irreversible, we can recover their inputs using XOR and work backwards
- This structure is called a **Feistel network**
 - Can be used for encryption algorithms if G and H depend on a key
 - Example: DES (out of scope)
- **Takeaway:** To fix the problems with RSA (it's only secure encrypting random numbers and isn't IND-CPA), use RSA with OAEP, abbreviated as RSA-OAEP



Hybrid Encryption

ITIS 6200 / 8200

- Issues with public-key encryption
 - Notice: We can only encrypt small messages because of the modulo operator
 - Notice: There is a lot of math, and computers are slow at math
 - Result: Asymmetric doesn't work for large messages
- **Hybrid encryption:** Encrypt data under a randomly generated key K using symmetric encryption, and encrypt K using asymmetric encryption
 - Benefit: Now we can encrypt large amounts of data quickly using symmetric encryption, and we still have the security of asymmetric encryption
- Almost all cryptographic systems use hybrid encryption
 - Scenario:
 - Alice wants to send a message to Bob
 - Alice chooses / generates a random symmetric key K
 - Alice computes $C1 = \text{Enc}(K, M)$ and sends it to Bob (Symmetric encryption)
 - Alice computes $C2 = \text{Enc}(\text{pub_bob}, K)$ and sends it to Bob (Asymmetric encryption)
 - Bob receives both messages
 - uses his private key to decrypt $C2$ and get K , and then
 - use K to decrypt $C1$ and get M

Digital Signatures

Cryptography Roadmap

ITIS 6200 / 8200

	Symmetric-key	Asymmetric-key
Confidentiality	<ul style="list-style-type: none">● One-time pads● Block ciphers with chaining modes (e.g. AES-CBC)	<ul style="list-style-type: none">● RSA encryption● ElGamal encryption
Integrity, Authentication	<ul style="list-style-type: none">● MACs (e.g. HMAC)	<ul style="list-style-type: none">● Digital signatures (e.g. RSA signatures)

- Hash functions
- Pseudorandom number generators
- Public key exchange (e.g. Diffie-Hellman)

- Key management (certificates)
- Password management

Digital Signatures

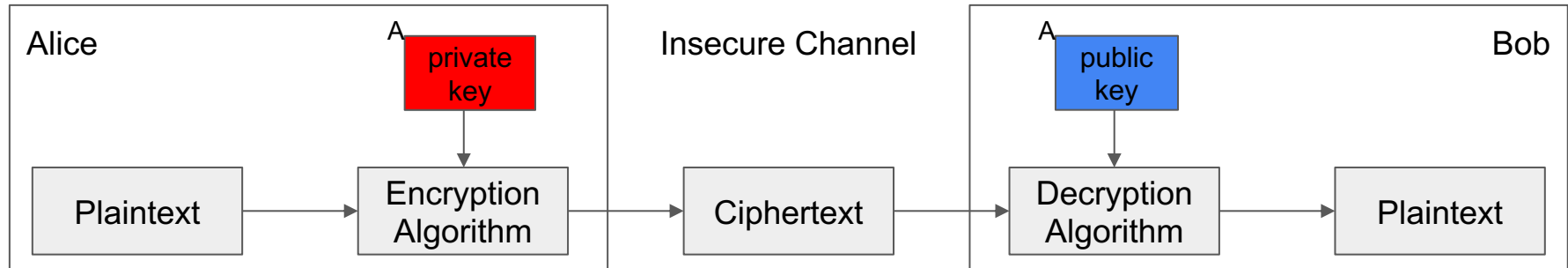
ITIS 6200 / 8200

- Asymmetric cryptography is good because we don't need to share a secret key
- Digital signatures are the asymmetric way of providing integrity/authenticity to data
- Assume that Alice and Bob can communicate public keys without Mallory interfering
 - We will see how to fix this limitation later

Public-key Signatures

ITIS 6200 / 8200

- Only the owner of the private key can sign messages with the private key
- Everybody can verify the signature with the public key



Digital Signatures: Definition

ITIS 6200 / 8200

- Three parts:
 - $\text{KeyGen}() \rightarrow PK, SK$: Generate a public/private keypair, where PK is the verify (public) key, and SK is the signing (secret) key
 - $\text{Sign}(SK, M) \rightarrow sig$: Sign the message M using the signing key SK to produce the signature sig
 - $\text{Verify}(PK, M, sig) \rightarrow \{0, 1\}$: Verify the signature sig on message M using the verify key PK and output 1 if valid and 0 if invalid
- Properties
 - **Correctness**: Verification should be successful for a signature generated over any message
 - $\text{Verify}(PK, M, \text{Sign}(SK, M)) = 1$ for all $PK, SK \leftarrow \text{KeyGen}()$ and M
 - **Efficiency**: Signing/verifying should be fast
 - **Security**: EU-CPA, same as for MACs

Digital Signatures in Practice

ITIS 6200 / 8200

- If you want to sign message M :
 - First hash M
 - Then sign $H(M)$
- Why do digital signatures use a hash?
 - Allows signing arbitrarily long messages
- Digital signatures provide integrity *and authenticity* for M
 - The digital signature acts as proof that the private key holder signed $H(M)$, so you know that M is authentically endorsed by the private key holder

RSA Signatures

RSA Signatures

ITIS 6200 / 8200

- Recall RSA encryption: $M^{ed} \equiv M \pmod{N}$
 - There is nothing special about using e first or using d first!
 - If we encrypt using d , then anyone can “decrypt” using e
 - Given x and $x^d \pmod{N}$, can't recover d because of discrete-log problem, so d is safe

RSA Signatures: Definition

ITIS 6200 / 8200

- **KeyGen():**
 - Same as RSA encryption:
 - **Public key:** N and e
 - **Private key:** d
- **Sign(d, M):**
 - Compute $H(M)^d \bmod N$
- **Verify(e, N, M, sig)**
 - Verify that $H(M) \equiv sig^e \bmod N = (H(M))^{d*e} \bmod N$

DSA Signatures

DSA Signatures

ITIS 6200 / 8200

- A signature scheme based on Diffie-Hellman
 - The details of the algorithm are out of scope
- Usage
 - Alice generates a public-private key pair and publishes her public key
 - To sign a message, Alice generates a random, secret value k and does some computation
 - Note: k is not Alice's private key
 - Note: k is sometimes called a nonce but it is not: it must be *random* and never reused
 - The signature itself does not include k !
- k must be random and secret for each message
 - An attacker who learns k can also learn Alice's private key
 - If Alice reuses k on two signatures, an attacker can learn k (and use k to learn her private key)

DSA Signatures: Attacks

ITIS 6200 / 8200

- Sony PlayStation 3 (PS3)
- Digital rights management (DRM)
 - Prevent unauthorized code (e.g. pirated software) from running
 - The PS3 was designed to only run signed code
 - Signature algorithm: Elliptic-curve DSA
- Running alternate operating systems
 - The PS3 had an option to run alternate operating systems (Linux) that was later removed
 - This was catnip to reverse engineers
- One of the authentication keys used to sign the firmware reused k for multiple signatures → security lost!

DSA Signatures: Attacks

ITIS 6200 / 8200

- Android OS vulnerability (2013)
 - The "SecureRandom" function in its random number generator (RNG) wasn't actually secure!
 - Not only was it low entropy, it would sometimes return the same value multiple times
- Multiple Bitcoin wallet apps on Android were affected
 - Bitcoin payments are signed with elliptic-curve DSA and published publicly
 - Insecure RNG caused multiple payments to be signed with the same k
- Attack: Someone scanned for all Bitcoin transactions signed insecurely
 - Recall: When multiple signatures use the same k , the attacker can learn k and the private key
 - In Bitcoin, your private key unlocks access to all your money

DSA Signatures: Attacks

ITIS 6200 / 8200

- Chromebooks have a built-in U2F (universal second factor) security key
 - Uses signatures to let the user log in to particular websites
 - Signature algorithm: 256-bit elliptic-curve DSA
- There was a bug in the secure hardware!
 - Instead of using 256-bit k , a bug caused k to be 32 bits long!
 - An attacker with a signature could simply try all possible values of k
- Fortunately the damage was slight
 - Each signature is only valid for logging into a single website
 - Each website used its own private key
- **Takeaway:** DSA (or ECDSA) is particularly vulnerable to incorrect implementations, compared with RSA signatures

Summary: Public-Key Cryptography

ITIS 6200 / 8200

- Public-key cryptography: Two keys; one undoes the other
- Public-key encryption: One key encrypts, the other decrypts
 - Security properties similar to symmetric encryption
 - RSA: Produce a pair e and d such that $M^{ed} = M \bmod N$
 - Not IND-CPA secure on its own
- Hybrid encryption: Encrypt a symmetric key, and use the symmetric key to encrypt the message
- Digital signatures: Integrity and authenticity for asymmetric schemes
 - RSA: Same as RSA encryption, but encrypt the hash with the *private* key