CSE 565 Computer Security
Fall 2018

Lecture 5: Public Key Cryptography

Department of Computer Science and Engineering
University at Buffalo

Public-Key Cryptography

- What we already know
  - symmetric key cryptography enables confidentiality
    - achieved through secret key encryption
  - symmetric key cryptography enables authentication and integrity
    - achieved through MACs
- In all of the above the sender and received must share a secret key
  - need a secure channel for key distribution
  - not possible for parties with no prior relationship
  - more powerful public-key cryptography can aid with this

Public-Key Cryptography

- **Public-key encryption**
  - **a party creates a public-private key pair**
    - **the public key is pk**
    - **the private or secret key is sk**
  - **the public key is used for encryption and is publicly available**
  - **the private key is used for decryption only**

$$D_{sk}(E_{pk}(M)) = M$$

  - **knowing the public key and the encryption algorithm only, it is computationally infeasible to find the secret key**
  - **public-key crypto systems are also called asymmetric**

Public-Key Cryptography

- **Digital signatures**
  - a party generated a public-private signing key pair
  - private key is used to sign a message
  - public key is used to verify a signature on a message
  - can be viewed as one-way message authentication
- (Public-key) **Key agreement or key distribution**
  - prior to the protocols the parties do not share a common secret
  - after the protocol execution, they hold a key not known to any eavesdropper

How Public-Key Cryptography Works

- **Public-key constructions often use number theory and are based on a special function f with the following properties**
  - **given f and x, it is easy to compute $f(x)$**
  - **given $f(x)$, it is hard to compute x**
  - **given $f(x)$ and an additional secret t, it is easy to find x**
  - **function f is called a one-way trapdoor function and t is called the trapdoor of f**
- **Given such a function f, we construct encryption as follows:**
  - **f is equivalent to encryption E_{pk}**
  - **the private key serves the purpose of the trapdoor**
  - **given $f(x) = E_{pk}(x)$ and the trapdoor sk, decryption of x is easy**

Public-Key Encryption

- Similar to symmetric encryption, we can formulate a number of attacks on public-key encryption
  - ciphertext only attack
  - known plaintext attack
  - chosen plaintext attack
  - chosen ciphertext attack
- Which types are not meaningful and which adequately model adversarial capabilities?

Public-Key Encryption

- Almost all public-key encryption algorithms use number theory and modular arithmetic
  - RSA is based on the hardness of factoring large numbers
  - ElGamal is based on the hardness of solving discrete logarithm problem
- RSA is the most commonly used public-key encryption algorithm invented by Rivest, Shamir, and Adleman in 1978
  - sustained many years of attacks on it
  - relies on the fact that factoring large numbers is hard
    - let $n = pq$, where p and q are large primes
    - given only n, it is hard to find p or q, which are used as a trapdoor

RSA Cryptosystem

- **RSA key generation**
  - **generate two large prime numbers p and q of the same length**
  - **compute $n = pq$**
  - **choose a small prime number e**
  - **compute the smallest d such that $ed \bmod (p-1)(q-1) = 1$**
  - **here $\phi(n) = (p-1)(q-1)$ is Euler's totient function**
- **Public key is (e, n)**
- **Private key is d**

RSA Cryptosystem

- **Encryption**
  - **given a message** m **such that** $0 < m < n$
  - **given a public key** $pk = (e, n)$
  - **encrypt as** $c = E_{pk}(m) = m^e \bmod n$
- **Decryption**
  - **given a ciphertext** c **(**$0 < c < n$**)**
  - **given a public key** $pk = (e, n)$ **and the corresponding private key** $sk = d$
  - **decrypt as** $m = D_{sk}(c) = c^d \bmod n$

RSA Cryptosystem

- **RSA Example**
  - **key generation**
    - $p = 11, q = 7, n = pq = 77, \phi(n) = 60$
    - $e = 37 \Rightarrow d = 13$ **(i.e.,** $ed = 481$**;** $ed \bmod 60 = 1$**)**
    - **public key is** $pk = (37, 77)$ **and private key is** $sk = 13$
  - **encryption**
    - **let** $m = 15$
    - $c = E(m) = m^e \bmod n = 15^{37} \bmod 77 = 71$
  - **decryption**
    - $m = D(c) = c^d \bmod n = 71^{13} \bmod 77 = 15$

Security of RSA

- **Existing attacks on RSA**
  - **brute force search (try all possible keys)**
  - **number theoretic attacks (factor n)**
    - **complicated factoring algorithms that run in sub-exponential (but super-polynomial) time in the length of n exist**
    - **a 768-bit modulus was factored in 2009**
    - **1024-bit moduli could be factored very soon**
    - **moduli of length 2048 are expected to be secure until 2030**
  - **special use cases**
    - **e.g., encrypting small messages with small e**
- **Plain (or textbook) RSA is not close to secure**

Towards Safe Use of RSA

- **Padded RSA**
  - plain RSA is deterministic
  - this is even worse than in case of symmetric encryption
    - anyone can search for m encrypting various messages
  - we can **randomize ciphertext by padding** each m with random bits
    - now a message can be at most $k - t$ bits long
    - random t bits are added to it such that 2^t work is infeasible
- **PKCS #1 v1.5** is a widely used standard for padded RSA
  - PKCS = RSA Laboratories Public-Key Cryptography Standard
  - it is believed to be CPA-secure

Towards Safe Use of RSA

- **PKCS #1 v2.0** utilizes OAEP (Optimal Asymmetric Encryption Padding)
  - the newer version mitigates some attacks on v1.5 and is known to be CCA-secure
- **Making factoring infeasible**
  - choose n to be long enough (we can choose any n!)
  - for a security parameter k, compute n with $|n| = k$
- A good implementation will also have countermeasures against **implementation-level attacks**
  - timing attacks, special cases of e and d, etc.

Other Public-Key Algorithms

- **Many popular public-key algorithms rely on the difficulty of discrete logarithm problem**
  - **ElGamal encryption and ElGamal signature**
  - **Digital Signature Algorithm (DSA)**
  - **Diffie-Hellman key exchange**
  - **…**
- **Given an appropriate setup with g, p, and $h = g^x \bmod p$, it is difficult for someone to compute x**
  - **x is called the discrete logarithm of h to the base g**
  - **groups in which the discrete logarithm problem is hard use prime modulus p (conventional and elliptic curve settings)**

Symmetric vs Public-Key Encryption

- **Public-key operations are orders of magnitude slower than symmetric encryption**
  - **an exponentiation modulo n requires close to $O(|n|^3)$ work**
  - **public-key encryption is not used to communicate large volumes of data**
  - **it is rather used to communicate (or agree on) a symmetric key**
  - **the data itself is sent encrypted with the symmetric key**

Digital Signatures

- A digital signature scheme is a method of signing messages stored in electronic form and verifying signatures
- Digital signatures can be used in very similar ways conventional signatures are used
  - paying by a credit card and signing the bill
  - signing a contract
  - signing a letter
- Unlike conventional signatures, we have that
  - digital signatures are not physically attached to messages
  - we cannot compare a digital signature to the original signature

Digital Signatures

- **Digital signatures allows us to achieve the following security objectives:**
  - **authentication**
  - **integrity**
  - **non-repudiation**
    - **note that this is the main difference between signatures and MACs**
    - **a MAC cannot be associated with a unique sender since a symmetric shared key is used**
- **What security property do we want from a digital signature scheme?**
- **A digital signature scheme consists of key generation, message signing, and signature verification algorithms**

Digital Signatures

- **Key generation** creates a public-private key pair (pk, sk)
- **Signing algorithm** takes a messages and uses private signing key to output a signature
- **Signature verification algorithm** takes a message, a signature on it, and the signer's public key and outputs a yes/no answer
- **RSA can be used for signing messages**
  - create a key pair as before
  - **signing** is done by decrypting a message with the private key $sig(m) = D_{sk}(m)$
  - **verification** is performed by encrypting the signature with the public key and comparing to the message $E_{pk}(sig(m)) \stackrel{?}{=} m$

Digital Signatures

- **Plain RSA is not a secure signature scheme**
  - **both existential and selective forgeries are easy**
  - **the "hash-and-sign" paradigm is used in many constructions to achieve adequate security**
  - **e.g., in RSA $sig(m) = D_{sk}(h(m))$ and verify $E_{pk}(sig(h(m)) \stackrel{?}{=} h(m)$**
  - **this additionally improves efficiency**
  - **the hash function must satisfy all three security properties**
    - **preimage resistance**
    - **weak collision resistance**
    - **strong collision resistance**

Digital Signatures

- **RSA signatures**
  - **key generation**
    - **choose prime** p **and** q**, compute** $n = pq$
    - **choose prime** e **and compute** d **s.t.** $ed \bmod (p-1)(q-1) = 1$
    - **signing key is** d**, verification key is** (e, n)
  - **message signing**
    - **given** m**, compute** $h(m)$
    - **output** $sig(m) = h(m)^d \bmod n$
  - **signature verification**
    - **given** m **and** $sig(m)$**, first compute** $h(m)$
    - **check whether** $sig(m)^e \bmod n \stackrel{?}{=} h(m)$

Digital Signature Standard (DSS)

- **Digital Signature Standard (DSS) or Digital Signature Algorithm (DSA) was adopted as a standard in 1994**
  - **its design was influenced by prior ElGamal and Schnorr signature schemes**
  - **it assumes the difficulty of the discrete logarithm problem**
  - **no formal security proof exists**

Digital Signature Standard (DSS)

- **DSS was published in 1994 as FIPS PUB 186**
  - **it was specified to hash the message using SHA-1 before signing**
  - **it was specified to produce a 320-bit signature on a 160-bit hash**
- **The current version is FIPS PUB 186-4 (2013)**
  - **DSA can now be used with a 1024-, 2048-, or 3072-bit modulus**
  - **the message size is 320, 448, or 512 bits**

Digital Signature Security

- Thorough evaluation of security of a signature scheme is crucial
  - often a message can be encrypted and decrypted once and long-term security for the key is not required
  - signatures can be used on legal documents and may need to be verified many years after signing
  - choose the key length to be secure against future computing speeds

The Big Picture

- How we address security goals using different tools

Security goal	Symmetric key setting	Public key setting
Secrecy / confidentiality	block ciphers with encryption modes (AES); stream ciphers	public key encryption (RSA, ElGamal, etc.)
Authenticity / integrity	message authentication codes (CBC-MAC, HMAC)	digital signatures (RSA, DSA, etc.)

Diffie-Hellman Key Exchange

- **Diffie-Hellman key exchange protocol**
  - **Alice and Bob want to compute a shared key, which must be unknown to eavesdroppers**
  - **Alice and Bob share public parameters: modulus p, element $1 < g < p$, and modulus q for computation in the exponent**
  - **Alice randomly chooses $x \in \mathbb{Z}_q$ and sends $g^x \bmod p$ to Bob:**
    $A \overset{g^x \bmod p}{\longrightarrow} B$
  - **Bob randomly chooses $y \in \mathbb{Z}_q$ and sends $g^y \bmod p$ to Alice:**
    $A \overset{g^y \bmod p}{\longleftarrow} B$

Diffie-Hellman Key Exchange

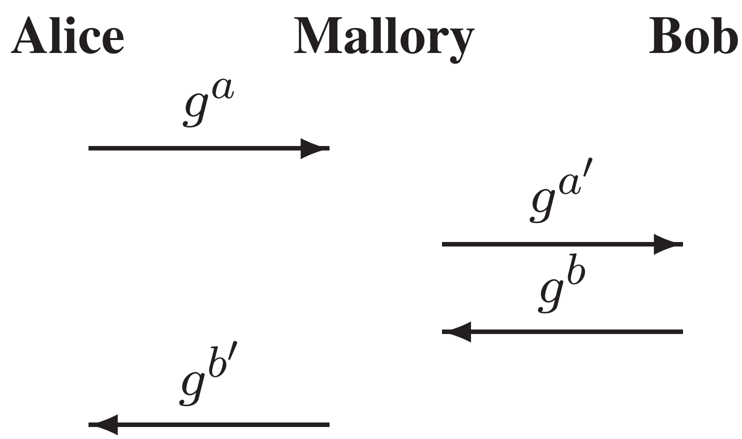
- **Diffie-Hellman key exchange protocol**
  - **the shared secret is set to** $g^{xy} \bmod p$
    - **Alice computes it as** $(g^y)^x \bmod p = g^{xy} \bmod p$
    - **Bob computes it as** $(g^x)^y \bmod p = g^{xy} \bmod p$
  - **it is believed to be infeasible for an eavesdropper to compute** g^{xy} **given** g^x **and** g^y

Diffie-Hellman Key Exchange

- **Diffie-Hellman key exchange**
  - the security property holds only against a passive attacker
  - the protocol has a serious weakness in the presence of an active adversary
    - this is called a **man-in-the-middle attack**
    - Mallory will intercept messages between Alice and Bob and substitute her own
    - Alice establishes a shared key with Mallory and Bob also establishes a shared key with Mallory

Diffie-Hellman Key Exchange

- **Man-in-the-middle attack on Diffie-Hellman key exchange**



  - **Alice shares the key $g^{ab'}$ with Mallory**
  - **Bob shares the key $g^{a'b}$ with Mallory**
  - **Alice and Bob do not share any key**
  - **what is Mallory capable of doing?**

Diffie-Hellman Key Exchange

- Alice and Bob need to make sure they are exchanging messages with each other
  - there is a need for authentication
  - preceding this protocol with an authentication scheme is not guaranteed to solve the problem
    - authentication needs to be a part of the key exchange
    - this is called authenticated key exchange
- A solution that addresses the problem relies on certificates and digital signatures

Bit Security

- All constructions studied so far rely on the fact that an adversary is limited in computational power
  - if it has more resources than we anticipate, cryptographic algorithms can be broken
- Today, 112–128-bit security is considered sufficient
  - this means approximately that for 128-bit security, 2^{128} operations are needed to violate security with high probability
- This translates into the following parameters
  - symmetric key encryption: the key size is at least 112 bits
  - hash functions: the hash size is at least 224 bits
  - public key encryption: the modulus is at least 2048 bits long

Conclusions

- **Proper use of cryptographic tools requires great care**
- **Safe use of such algorithms involves**
  - **familiarity with known attacks**
  - **adequate choice of parameters**
  - **including countermeasures against known attacks on implementations**
  - **using a cryptographically strong source of randomness**
- **No security by obscurity!**