CSE 715 Fall 2025
Design and Implementation of Application for Computing with Private Data

Marina Blanton

Department of Computer Science and Engineering
University at Buffalo

Lecture 1: Cryptographic Concepts

Data Protection

Cryptographic techniques are used for

- Protecting data at rest
- Protecting data in transmission
- Protecting data during computation

Data Protection

When protecting data, you typically want both:

- data confidentiality – guarantees that data is remains private and unavailable to unauthorized parties
- data integrity – guarantees that data is authentic and has not be tampered with by unauthorized parties

This is typically achieved by means of symmetric cryptography

- symmetric encryption for confidentiality
- message authentication codes for integrity
- security objectives differ $\Rightarrow$ tools to realize them also differ

Symmetric Encryption

Symmetric (or secret-key) encryption means that the same key is used both for encryption and decryption

The key must be available at both times (and stored securely in between)

Such algorithms are:

- normally very fast
- can be used as primitives in more complex cryptographic protocols
- the key often has a short lifetime

Computational Security

Most of the cryptograhic techniques we'll discuss are computationally secure

- this means they are breakable in principle
- breaking them is very difficult for a computationally limited adversary
- algorithms are parameterized by a security parameter
- legitimate work has to be polynomial time in the security parameter
- breaking has to require super-polynomial effort
- alternatively, a polynomial-time adversary cannot succeed with reasonable probability

Symmetric Encryption Formally

More formally, a computationally secure symmetric key encryption scheme is defined as:

- a symmetric encryption scheme consists of polynomial-time algorithms (Gen, Enc, Dec) such that
  - Gen: on input the security parameter n, outputs key k
  - Enc: on input a key k and a message $m \in \{0,1\}^*$, outputs ciphertext c
  - Dec: on input a key k and ciphertext c, outputs plaintext m
- we write $k \leftarrow \mathsf{Gen}(1^n)$, $c \leftarrow \mathsf{Enc}_k(m)$, and $m := \mathsf{Dec}_k(c)$
  - this notation means that Gen and Enc are probabilistic and Dec is deterministic

Attacks Against Symmetric Encryption

- Encryption and decryption algorithms are assumed to be known to the adversary
- Types of attacks
  - ciphertext only attack: adversary knows a number of ciphertexts
  - known plaintext attack: adversary knows some pairs of ciphertexts and corresponding plaintexts
  - chosen plaintext attack: adversary knows ciphertexts for messages of its choice
  - chosen ciphertext attack: adversary knows plaintexts for ciphertexts of its choice
- We want a general-purpose algorithm to sustain all types of attacks

Security Against Chosen-Plaintext Attacks

- In chosen-plaintext attack (CPA), adversary $\mathcal{A}$ is allowed to ask for encryptions of messages of its choice
  - $\mathcal{A}$ is given black-box access to encryption oracle and can query it on different messages
- $\mathcal{A}$ is asked to distinguish between encryptions of messages of its choice
  - $\mathcal{A}$ chooses two messages m_0 and m_1 and one of them is encrypted
  - $\mathcal{A}$'s task is to determine which message was encrypted
  - $\mathcal{A}$ has only negligible chances of success with CPA-secure encryption

Symmetric Encryption

The above definition allows us to encrypt messages of any length

In practice, there are two types of symmetric key algorithms:

- block ciphers
  - the key has a fixed size
  - prior to encryption, the message is partitioned into blocks
  - each block is encrypted and decrypted separately
- stream ciphers
  - the message is processed as a stream
  - pseudo-random generator is used to produce a long key stream from a short key

Block Ciphers

- The algorithm maps an n-bit plaintext block to an n-bit ciphertext block
- Most modern block ciphers are product ciphers
  - we sequentially apply more than one operation to the message
- Often a sequence of permutations and substitutions is used
- A common design for an algorithm is to proceed in iterations
  - one iteration is called a round
  - each round consists of similar operations
  - ith round key k_i is derived from the secret key k using a fixed, public algorithm

Block Ciphers

Confusion-diffusion design paradigm

- split a block into small chunks
- define a permutation on each chunk separately (confusion)
- mix outputs from different chunks by rearranging bits (diffusion)
- repeat to strengthen the result

Block cipher standards

- Data Encryption Standard DES (historical)
- Advanced Encryption Standard AES (current)

Secure Encryption

Using a strong block cipher is not enough for secure encryption

- If you are to send more than 1 block (16 bytes) over the key lifetime, applying plain block cipher to the message as

$$\mathsf{Enc}_k(b_1), \mathsf{Enc}_k(b_2), \ldots$$

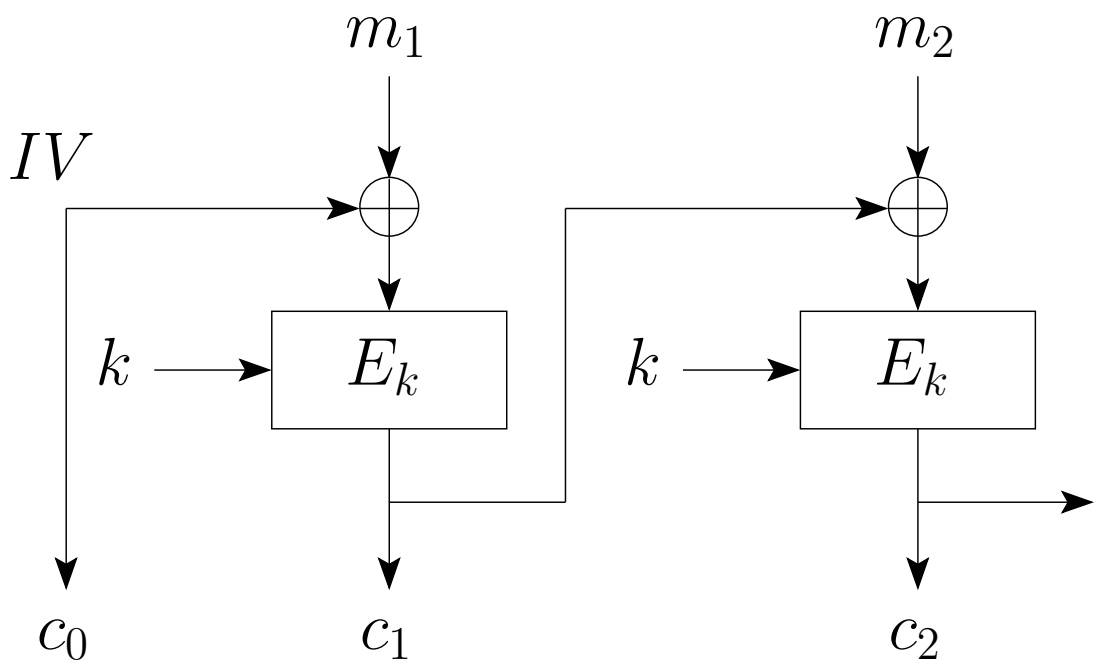
  will fail even weak definitions of secure encryption
- No deterministic encryption can be secure if multiple blocks are sent

Encryption Modes

- Encryption modes indicate how messages longer than one block are encrypted and decrypted
- 4 modes of operation were standardized in 1980 for Digital Encryption Standard (DES)
  - can be used with any block cipher
  - electronic codebook mode (ECB), cipher feedback mode (CFB), cipher block chaining mode (CBC), and output feedback mode (OFB)
- 5 modes were specified with the current standard Advanced Encryption Standard (AES) in 2001
  - the 4 above and counter mode

Cipher Block Chaining (CBC) Mode

- Set $c_0 = IV \xleftarrow{R} \{0,1\}^n$ (initialization vector)
- Encryption: for $i = 1, \ldots, \ell$, $c_i = E_k(m_i \oplus c_{i-1})$
- Decryption: for $i = 1, \ldots, \ell$, $m_i = c_{i-1} \oplus D_k(c_i)$, where D is block cipher decryption



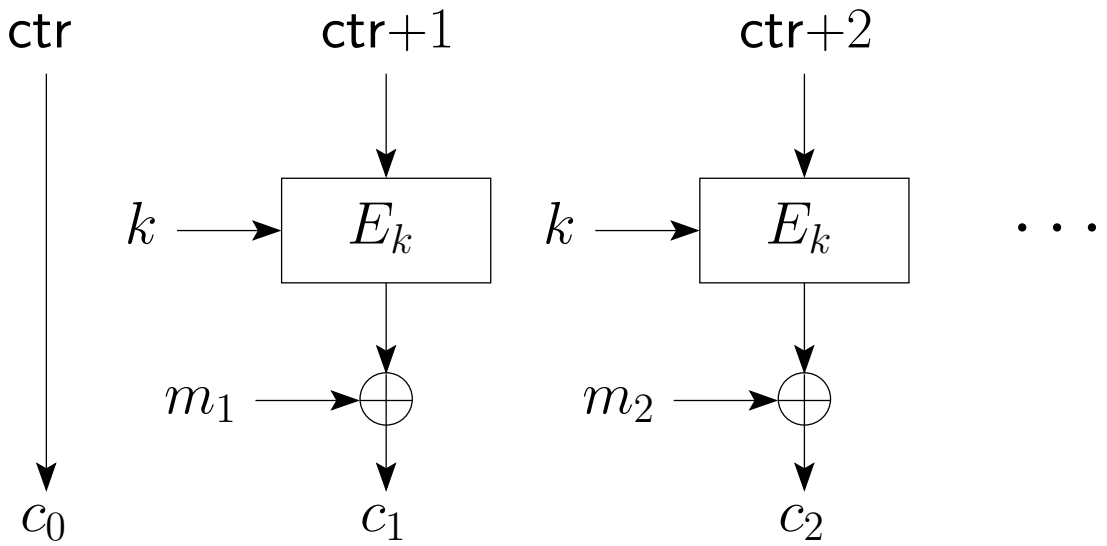
CBC Mode

Properties of the CBC mode:

- this mode is CPA-secure (has a formal proof) if the block cipher can be assumed to produce pseudorandom output
- a ciphertext block depends on all preceding plaintext blocks
- sequential encryption, cannot use parallel hardware
- IV must be random and communicated intact
  - if the IV is not random, security quickly degrades
  - if someone can fool the receiver into using a different IV, security issues arise

Counter (CTR) Mode

- A counter is encrypted and XORed with a plaintext block
- No feedback into the encryption function
- Initially set $\mathsf{ctr} = IV \xleftarrow{R} \{0,1\}^n$



CTR Mode

Counter (CRT) mode

- encryption: for $i = 1, \ldots, \ell$, $c_i = E_k(\mathsf{ctr} + i) \oplus m_i$
- decryption: for $i = 1, \ldots, \ell$, $m_i = E_k(\mathsf{ctr} + i) \oplus c_i$

Properties:

- there is no need to pad the last block to full block size
- if the last plaintext block is incomplete, we just truncate the last cipherblock and transmit it

CTR Mode

Advantages of the counter mode

- Hardware and software efficiency: multiple blocks can be encrypted or decrypted in parallel
- Preprocessing: encryption can be done in advance; the rest is only XOR
- Random access: ith block of plaintext or ciphertext can be processed independently of others
- Security: at least as secure as other modes (i.e., CPA-secure)
- Simplicity: doesn't require decryption or decryption key scheduling

The counter can't be reused

Randomness Generation

- All cryptographic constructions that are non-deterministic or produce key material require randomness
  - key generation for any type of cryptographic tool
  - drawing randomness during probabilistic encryption
- What do we expect from a random bit sequence?
  - uniform distribution: all possible values are equally likely
  - independence: no part of the sequence depends on its other parts
- Where do we find randomness?

Randomness Generations

- Randomness can be gathered from physical, unpredictable processes
- Example sources of true randomness
  - least significant bits of time between key strokes
  - noise from a mouse, video camera, and microphone
  - variation in response times of raw read requests from a disk
- Amount of required randomness may not be small
  - example: choosing a 1024-bit prime
- Instead of a true randomness we can use a pseudo-random generator (PRG)

Pseudo-Random Generators

- A pseudo-random generator is an algorithm that
  - takes a short value, called a seed, as its input
  - produces a long string that is statistically close to a uniformly chosen random string
  - the bitlength of a seed is based on a security parameter
- The security requirement is that
  - a computationally bounded adversary cannot tell the output of a PRG apart from a truly random string of the same size
  - in practice, a number of statistical tests are used to test the strength of a PRG

Pseudo-Random Generators

PRGs are deterministic

- the output is always the same on the same seed
- for cryptographic purposes, it is crucial that the seed is hard to guess
  - i.e., use strong true randomness to generate a seed

Example of a PRG

- symmetric block ciphers, such as AES, can be used as PRGs
- given a key k (seed), produce a stream as $\mathsf{Enc}_k(0), \mathsf{Enc}_k(1), \ldots$, where Enc is block cipher encryption

Data Integrity

- Encryption protects data only from a passive attack
  - we often also want to protect message from active attacks (modification or falsification of data)
  - such protection is called message or data authentication
- Goals of message authentication
  - a message is authentic if it came from its alleged source in its genuine form
  - message authentication allows verification of message authenticity

Message Authentication

- How can message authentication be performed?
  - in addition to the message itself, another token that authenticates the message, often called a tag, is transmitted
  - the cryptographic primitive is called a Message Authentication Code (MAC)
- Message authentication is independent of encryption
  - it can be used with or without encryption
  - a number of applications benefit from message authentication alone

Message Authentication

- What do we want from a message authentication code?
  - a tag should be easy to compute by legitimate parties, but hard to forge by an adversary
- MAC constructions use a secret key
  - a secret key is shared by two communicating parties
  - a MAC cannot be computed (or verified) without the key
- To achieve source authentication and message integrity:
  - the sender computes $t = \mathsf{MAC}_k(m)$ and sends (m, t)
  - the receiver recomputes $t' = \mathsf{MAC}_k(m)$ for received m and compares it to t

Message Authentication Codes

- A MAC scheme is defined by three algorithms:
  - key generation: a randomized algorithm, which on input a security parameter n, produces key a k
  - MAC generation: a possibly randomized algorithm, which on input a message m and key k, produces a tag t
  - MAC verification: a deterministic algorithm, which on input a message m, tag t, and key k, outputs a bit b

Message Authentication Codes

- Properties of MAC algorithms
  - most fundamentally, we desire correctness and security
  - correctness requires that a correctly computed tag will always verify
  - security will be defined later and intuitively requires that it is hard to forge a tag on a new message without the key
  - from a performance point of view, we desire tags of a fixed size (i.e., independent of the message length)

Message Authentication Codes

- Classification of attacks on MACs:
  - known-text attack: one or more pairs $(m_i, \mathsf{Mac}_k(m_i))$ are available
  - chosen-text attack: one of more pairs $(m_i, \mathsf{Mac}_k(m_i))$ are available for m_i's chosen by the adversary
  - adaptive chosen-text attack: the m_i's are chosen by the adversary, where successive choices can be based on the results of prior queries
- Against which kind of attack do we want to be resilient?

Message Authentication Codes

- Classification of forgeries:
  - selective forgery: an adversary is able to produce a new MAC pair for a message under her control
  - existential forgery: an adversary is able to produce a new MAC pair but with no control of the value of the message
- Resilience against which type would be preferred?
- And, as usual, key recovery is the most damaging attack on MAC

Message Authentication Codes

- We desire for a MAC to be existentially unforgeable under an adaptive chosen-message attack
  - an adversary is allowed to query tags on messages of its choice
  - at some point it outputs a pair (m, t)
  - the forgery is considered successful if m hasn't been queried before and t is a valid tag for it
  - as with encryption, security guarantees depend on the security parameter
- MACs do not prevent all traffic injections
  - a replayed message will pass verification process
  - it is left to the application to make each message unique

Message Authentication Codes

- There are two most common (standardized) implementations of MAC functions
  - CBC-MAC: based on a symmetric encryption (e.g., AES) in the CBC mode with important modifications
  - HMAC: based on a hash function
    - it is important to understand properties of cryptographic hash functions
    - read about the HMAC construction

Confidentiality + Integrity

- How do we use a MAC in combination with encryption?
  - message authentication

    $m,\ \mathsf{Mac}_k(m)$
  - encrypt and authenticate

    $\mathsf{Enc}_{k_1}(m),\ \mathsf{Mac}_{k_2}(m)$
  - authenticate then encrypt

    $\mathsf{Enc}_{k_1}(m,\ \mathsf{Mac}_{k_2}(m))$
  - encrypt then authenticate

    $\mathsf{Enc}_{k_1}(m),\ \mathsf{Mac}_{k_2}(\mathsf{Enc}_{k_1}(m))$

Confidentiality + Integrity

- The goal is now to achieve both confidentiality and integrity properties at once
  - this is called authenticated encryption
- Analysis of prior constructions:
  - encrypt and authenticate
    - transmitting $\mathsf{Mac}_{k_2}(m)$ may leak information about m
  - authenticate then encrypt
    - has a chosen-ciphertext attack against the general version, which has been successfully applied in practice
  - encrypt then authenticate
    - satisfies the definition of authenticated encryption and is CCA-secure
- The keys k_1 and k_2 must be different!

Authenticated Encryption

- Do I have to use encryption and MAC separately or are there authenticated encryption modes?
- Good options to consider:
  - Offset Codebook (OCB) mode
    - state of the art in authenticated encryption
    - proposed internet standard
    - used to have licensing restrictions
    - see http://web.cs.ucdavis.edu/~rogaway/ocb/ocb-faq.htm for more information
  - Galois/Counter Mode (GCM)
    - does not have licensing restrictions
    - can be used as an alternative to commercial software

Putting It All Together

- AES is the current block cipher standard for symmetric encryption
  - it offers strong security and fast performance
- Secure encryption requires the use of a secure encryption mode
- Integrity protection is often required together with confidentiality
  - use an authenticated encryption mode
- Strong randomness is required for cryptographic purposes
  - key generation, IV generation, etc.
- Implementing cryptographic constructions is hard

Proteching Data in Transit

- We discussed many aspects of protecting data at rest
- Similar mechanisms are used for protecting data in transit
- The biggest missing piece is how to distribute keys for symmetric cryptography over the insecure internet
- This requires additional tools

Key Distribution Mechanisms

- Assume users communicate over an insecure network
- There are different possibilities for key distribution
  - they fundamentally differ based on whether the participants are within the same administrative domain
- When users are within the same adminstrative domain, we can rely on a trusted authority (TA)
  - the TA can verify user identities, issue certificates, transmit keys, etc.

Key Distribution Mechanisms

The difference between key distribution and key agreement:

- in key distribution, one party (a TA) chooses a key and transmits it to one or more parties
  - key transmission is performed in an encrypted form
- in key agreement, two or more parties jointly establish a secret key
  - communication is performed over a public channel
  - each participant contributes to the value of the resulting key
  - the key is not sent from one party to another

We'll talk about key agreement (key exchange) next

- this is achieved by means of public-key cryptography

Public-Key Cryptography

Public-key encryption

- a party creates a public-private key pair (pk, sk)
- the public key pk is used for encryption and is publicly available
- the private key sk is used for decryption only

$$\mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(m)) = m$$

- 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 (pk, sk)
- private key sk is used to sign a message
- public key pk 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

Background

Before we proceed with the computation, we need to discuss the setup

Groups are a convenient way to represent sets and work with them

Background

A group G is a set of elements together with a binary operation $\circ$ such that

- the set is closed under the operation $\circ$, i.e., for every $a, b \in G$, $a \circ b$ is a unique element of G
- the associative law holds, i.e., for all $a, b, c \in G$, $a \circ (b \circ c) = (a \circ b) \circ c$
- the set has a unique identity element e such that $a \circ e = e \circ a = a$ for every $a \in G$
- every element has a unique inverse a^{-1} in G such that $a \circ a^{-1} = a^{-1} \circ a = e$

Groups

- A group is finite if it has only a finite number of elements
- The number of elements of a finite group is called the order of the group
- If a is an element of a finite group with identity 1, then there is a unique smallest positive integer i with $a^i = 1$ (using multiplicative notation)
  - such i is called the order of a (different from the order of the group)
- The element a has infinite order is there is no positive integer i with $a^i = 1$
- A cyclic group is one that contains an element a whose powers a^i and a^{-i} make up the entire group
- An element a with such property is called a generator of the group

Discrete Logarithm Problem

Discrete logarithms

- we are given a cyclic group G of order q and its generator g
- for each $h \in G$ there is a unique x such that $g^x = h$
- such x is called the discrete logarithm of h with respect to g, written as $x = \log_g h$

The discrete logarithm problem

- in a cyclic group G with given generator g, compute unique $\log_g h$ for a random element $h \in G$

We want the discrete logarithm problem to be hard (relative to its setup)

Key Agreement Schemes

- The best-known key exchange protocol is due to Diffie and Hellman
- Alice and Bob want to establish a shared key
  - the common parameters are (G, q, g)
  - Alice chooses a random number a from $\mathbb{Z}_q$, computes g^a, and sends g^a to Bob
  - Bob chooses a random number b from $\mathbb{Z}_q$, computes g^b, and sends g^b to Alice
  - Alice computes the shared key as $(g^b)^a = g^{ab}$
  - Bob computes the shared key as $(g^a)^b = g^{ab}$

Diffie-Hellman Key Exchange

Diffie-Hellman key exchange properties

- Alice and Bob compute the same key
- it is computationally difficult for someone else (without a or b) to compute their key
- the security property holds only against a passive attacker
  - a passive adversary simply monitors network communication
- a more powerful adversary can do severe damage
  - an active attacker can modify messages, prevent them from being delivered, masquerate as another user, ...

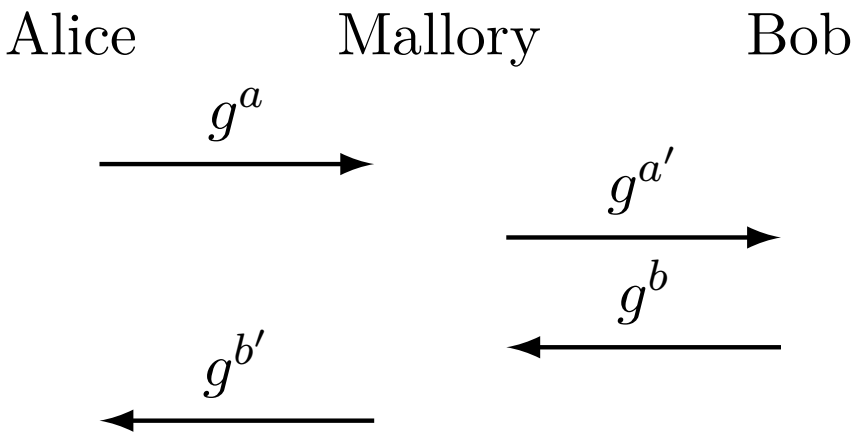
Diffie-Hellman Key Exchange

Security properties of Diffie-Hellman key exchange fall apart in the presense of an active adversary

- the attack 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
  - after they authenticate, the same attack can be carried out

We need a protocol that authenticates the participants at the same time the key is being established

- such a protocol is called an authenticated key agreement scheme
- it should simultaneously guarantee secure mutual authentication and secure key computation

Authenticated Diffie-Hellman Key Exchange

Authenticated Diffie-Hellman key exchange uses

- digital signatures
- certificates

We need to discuss those topics next

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 allow us to achieve the following security objectives:

- authentication
- integrity
- non-repudiation
  - this is the main difference between signatures and MACs

What security property do we want from a digital signature scheme? How does it relate to that of MACs?

Digital Signatures

It is meaningful to consider the following attack models

- key-only attack
- known message attack
- chosen message attack

Adversarial goals might be

- total break
- selective forgery
- existential forgery

As with MACs, we want existiential unforgeability under an adaptive chosen message attack

Digital Signatures

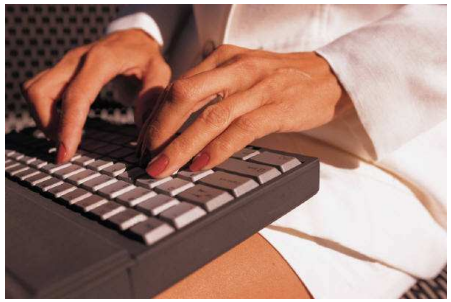
A digital signature scheme consists of three algorithms:

- key generation: given a security parameter κ, creates a public-private key pair (pk, sk)
- signing algorithm takes a messages m and uses private signing key sk and output a signature σ
- signature verification algorithm takes a message m, a signature on it σ, and the signer's public key pk and outputs a yes/no answer

Signature Algorithms

- RSA signature scheme
  - relies on the difficulty of factoring large composite moduli and hashing
- ElGamal signature scheme
  - was published in 1985 and works in groups where the discrete logarithm problem is hard
- Schnorr signature scheme
  - modifies ElGamal signature scheme to sign a digest of a message in a subgroup modulo p
- Digital Signature Algorithm (DSA)
  - a signature standard adopted by NIST

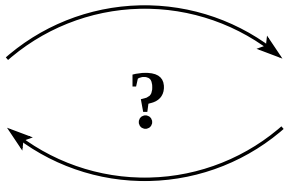
Public Keys and Trust



Alice

public key pk_A

secret key sk_A



Bob

public key pk_B

secret key sk_B

- If we want to use public-key cryptography, we are facing the key distribution problem
  - how/where are public keys stored?
  - how do I obtain someone's public key?
  - how can Bob know or "trust" that pk_A is indeed Alice's public key?

Public-Key Certificates

Digital certificates are used to address this problem

- A certificate binds identity (and/or other information) to a public key
- Assume there is a central authority CA with a known public key pk_{CA}
- CA produces certificate for Bob as $\mathsf{cert}_B = \mathsf{sig}_{CA}(\mathsf{pk}_B||\text{Bob})$
- Bob distributes $(\mathsf{pk}_B, \mathsf{cert}_B)$
- Alice can verify that her copy of Bob's key is genuine
- This technique is used in many applications
  - TLS/SSL, ssh, email, IPsec, etc.
- In practice, certificates contain other fields
  - e.g., the type of signing algorithm, expiration date, etc.

Diffie-Hellman Key Exchange

Authenticated Diffie-Hellman key exchange

- each user U has a private signing key sk_U and the corresponding public verification key pk_U
- there is a trusted authority TA that signs keys
- user U holds a certificate $\mathsf{cert}(U)$ issued by the TA

$$\mathsf{cert}(U) = (U, pk_U, \sigma_{TA}(U, pk_U))$$

- the protocol is also known as station-to-station key agreement
- it combines the key exchange with a mutual authentication scheme

Authenticated Diffie-Hellman Key Exchange (simplified)

- public parameters are as before (G, q, g)
- Alice chooses random a, computes $x_A = g^a$, and sends $\mathsf{cert}(A)$ and x_A to Bob
- Bob chooses random b, computes

$$x_B = g^b,\ k = (x_A)^b = g^{ab}\ , \text{and}\ \ y_B = \sigma_B(A||x_B||x_A)$$

  and sends $\mathsf{cert}(B)$, x_B, and y_B to Alice
- Alice verifies y_B; if the signature is valid, she computes

$$k = (x_B)^a = g^{ab}\ \ \text{and}\ \ y_A = \sigma_A(B||x_A||x_B)$$

  and sends y_A to Bob
- Bob verifies y_A; if the signature is valid, he accepts

Secure Communicatioen

Key components of secure communication:

- key agreement to generate a shared key
- communication protection using authenticated encryption
- secure randomness generation
- key management