Applied Cryptography and Computer Security
CSE 664 Spring 2020

Lecture 21: Encryption with Special Properties

Department of Computer Science and Engineering
University at Buffalo

Lecture Outline

- Homomorphic encryption
- ElGamal as homomorphic encryption
- Identity-based encryption as an alternative to PKI
- Boneh-Franklin IBE scheme
- Attribute-based encryption

Homomorphic Encryption

- Homomorphic encryption is a special type of encryption that, given ciphertexts, permits computation on the underlying plaintexts

$$\mathsf{Enc}_k(m_1) \otimes \mathsf{Enc}_k(m_2) = \mathsf{Enc}_k(m_1 \oplus m_2)$$

- Different types of homomorphic encryption are known:
  - partially homomorphic encryption
    - supports a single operation on ciphertexts
    - additively homomorphic encryption
      $\mathsf{Enc}_k(m_1) \cdot \mathsf{Enc}_k(m_2) = \mathsf{Enc}_k(m_1 + m_2)$
    - multiplicatively homomorphic encryption
      $\mathsf{Enc}_k(m_1) \cdot \mathsf{Enc}_k(m_2) = \mathsf{Enc}_k(m_1 \cdot m_2)$

Homomorphic Encryption

- Different types of homomorphic encryption
  - fully homomorphic encryption (FHE)
    - supports two operations on ciphertexts: addition and multiplication
    - allows for any functionality to be evaluated on encrypted data
- Homomorphic encryption enables computation on encrypted data and results in efficient protocols for certain problems

Homomorphic Encryption

- Examples of partially homomorphic encryption
  - additively homomorphic encryption: Paillier, additively homomophic ElGamal
    - property $\mathsf{Enc}_k(m_1) \cdot \mathsf{Enc}_k(m_2) = \mathsf{Enc}_k(m_1 + m_2)$ also implies $\mathsf{Enc}(m)^c = \mathsf{Enc}(m \cdot c)$
  - multiplicatively homomorphic encryption: regular ElGamal
  - fully homomorphic encryption
    - the first working construction is due to Gentry (2009)
    - many others followed
    - speed is presently an issue

Multiplicatively Homomorphic Encryption

- Recall ElGamal encryption
  - key generation
    - given a cyclic group G of order q and a generator $g \in G$, choose a random x from $\mathbb{Z}_q$ and compute $h = g^x$
    - public key $pk = (G, q, g, h)$ and private key $sk = x$
  - encryption
    - to encrypt a message $m \in G$, choose a random number $y \in \mathbb{Z}_q$
    - compute the ciphertext as $c = \mathsf{Enc}_{pk}(m) = (g^y, m \cdot h^y)$
- It enjoys the multiplicatively homomorphic property:

Additively Homomorphic Encryption

- Additively homomorphic ElGamal
  - generate the key as before
  - encrypt as $\mathsf{Enc}_{pk}(m) = (g^y, g^m \cdot h^y)$ instead of $\mathsf{Enc}_{pk}(m) = (g^y, m \cdot h^y)$
  - homomorphic properties:
  - decryption requires solving the discrete logarithm, so the scheme can be used only with messages from a small space

Paillier Encryption Scheme

- The following scheme was introduced by Pascal Paillier in 1999
  - semantically secure public-key encryption scheme
  - enjoys the additively homomorphic property
  - its security is based on the composite residuosity problem
    - let $n = pq$, where p and q are large primes
    - a number y is said to be an n-th residue modulo n^2 if there exists a number x with $gcd(x, n^2) = 1$ such that $y = x^n \bmod n^2$
    - it is believed that deciding n-th residuosity is computationally hard
  - in what follows, $\lambda(x)$ is Carmichael's function
    - for $n = pq$, $\lambda(n) = lcm(p - 1, q - 1)$

Paillier Encryption Scheme

- Key generation
  - choose large prime p and q and set $n = pq$
  - select a random base $g < n^2$ such that $gcd(L(g^\lambda(n) \bmod n^2), n) = 1$
  - the public key is (n, g)
  - the private key is (p, q)
- Encryption
  - to encrypt a plaintext $m < n$, select a random $r < n$
  - the ciphertext is $c = g^m \cdot r^n \bmod n^2$
  - notice that the ciphertext is twice as long as the plaintext

Paillier Encryption Scheme

- Decryption
  - given a ciphertext $c < n^2$
  - compute the plaintext m as

$$m = \frac{L(c^{\lambda(n)} \bmod n^2)}{L(g^{\lambda(n)} \bmod n^2)} \bmod n$$

  - here $L(x) = \frac{x-1}{n}$

- Homomorphic properties
  - $\mathsf{Enc}(m_1) \cdot \mathsf{Enc}(m_2) = \mathsf{Enc}(m_1 + m_2)$
  - $\mathsf{Enc}(m)^c = \mathsf{Enc}(c \cdot m)$

Paillier Encryption Scheme

- Homomorphic properties
  - first consider $\mathsf{Enc}(m_1) \cdot \mathsf{Enc}(m_2)$

$$\mathsf{Enc}(m_1) = g^{m_1} \cdot r_1^n \bmod n^2 \qquad \mathsf{Enc}(m_2) = g^{m_2} \cdot r_2^n \bmod n^2$$

$$\begin{aligned}\mathsf{Enc}(m_1) \cdot \mathsf{Enc}(m_2) &= g^{m_1} \cdot r_1^n \cdot g^{m_2} \cdot r_2^n \bmod n^2 \\ &= g^{m_1+m_2}(r_1 \cdot r_2)^n \bmod n_2 \\ &= \mathsf{Enc}(m_1 + m_2)\end{aligned}$$

  - now let us compute $\mathsf{Enc}(m)^c$

$$\begin{aligned}\mathsf{Enc}(m)^c &= (g^m \cdot r^n)^c \bmod n^2 = g^{mc} \cdot r^{nc} \bmod n^2 \\ &= g^{(mc)}(r^c)^n \bmod n^2 = g^{m_1} \cdot r_1^n \bmod n^2 = \mathsf{Enc}(m_1)\end{aligned}$$

where $m_1 = cm$ and $r_1 = r^c \bmod n$

Additively Homomorphic Encryption

- Equality testing using homomorphic encryption
  - Alice and Bob each know an important secret
  - they would like to determine whether Alice's secret s_A is the same as Bob's secret s_B without giving up any other information
    - i.e., they want to compute $s_A \stackrel{?}{=} s_B$ and obtain a true/false answer
  - this can be done using a public-key homomorphic encryption scheme
- The protocol's idea:
  - they compute, over encrypted data, the difference between s_A and s_B and multiply it by a random value
  - then after decryption, if the result is 0, the secrets are the same; and they are different otherwise

Equality Testing Protocol

- Protocol steps:
  - Alice chooses a public-private key pair (pk_A, sk_A) and gives the public key pk_A to Bob
  - Alice encrypts her secret and sends $\mathsf{Enc}_A(s_A)$ to Bob
  - Bob computes $\mathsf{Enc}_A(-s_B)$ and then $X = \mathsf{Enc}_A(s_A) \cdot \mathsf{Enc}_A(-s_B) = \mathsf{Enc}_A(s_A - s_B)$
  - Bob picks a large random r, computes $Y = X^r = \mathsf{Enc}_A(r(s_A - s_B))$, and sends Y to Alice
  - Alice decrypts the value and announces the result
    - if she decrypted a 0, $s_A = s_B$
    - if she decrypted anything else (a random value), $s_A \neq s_B$

Equality Testing Protocol

- Is this protocol secure?
  - what does Bob see?
  - what does Alice see?
  - why do we need to randomize the difference?
  - the protocol works only when Alice and Bob follow the directions
    - they follow the protocol, but might try to store intermediate values and try to compute extra information using them
    - such players are called semi-honest or honest-but-curious
    - a stronger model that maintains security under arbitrary behavior is called malicious model

Secure Multi-Party Computation

- More generally, secure multi-party computation allows for any desired function f to be securely evaluated on private data without revealing it
  - a number of parties hold private inputs $x_1, \ldots, x_n$
  - we evalute $f(x_1, \ldots, x_n)$ to obtain one or more outputs $y_1, \ldots$
  - each output y_i is revealed to a party or parties entitled to learning it
  - no other information about any x_i is available to any participant
    - more precisely, given your x_i and the output, you may deduce something about other x_is
    - but no additional information is revealed during the computation
  - this should hold even if a number of participants conspire against others and combine their information

Secure Multi-Party Computation

- To model security, we compare a real protocol execution with an ideal execution
  - in the ideal setting, no interaction takes place
    - the computation is performed by trusted party that received all inputs and computes outputs
  - showing security consists of demonstrating that real protocol execution can be simulated by querynig the trusted party in the ideal setting
  - this implies that messages transmitted by the protocol reveal no information about inputs
    - i.e., a participant cannot tell whether an intermediate message was simulated or computed using actual data

Secure Multi-Party Computation

- To summarize, security is shown as follows
  - we define adversarial capabilities
    - we assume either semi-honest or malicious behavior
  - we define what fraction of participants the adversary can corrupt
  - we show that the view of the participants controlled by the adversary is indistinguishable from the view in the ideal model
    - in the ideal model, we have access only to the inputs of corrupt parties and their outputs
    - needs to ensure that this property holds regardless of who is corrupt
- Besides homomorphic encryption, other common techniques are garbled circuit evaluation and secret sharing

Homomorphic Encryption

- Homomorphic encryption is a common tool used for secure computation and outsourcing
  - FHE allows for evaluation of any functionality, but is not performant
  - reduced versions that support any number of additions, but a limited number of sequential multiplications can be faster and suitable for some computations
    - this is called somewhat homomorphic encryption
  - partially HE can be used to evaluate any functionality by 2 or more parties
    - e.g., we can realize multiplication interactively

Identity-Based Encryption

- The development of large-scale PKIs has proceeded slowly and, as of today, no global infrastructure is available
  - thus, it is logical to seek alternatives to a PKI
- Identity-based encryption was proposed in the 1980s as an alternative to PKIs
  - the goal is to eliminate the need for managing public keys and the requirement of verifying their authenticity
  - instead, a user identity (e.g., an email address) can be used as her public key
  - a message can be encrypted and sent to any user without having to maintain their public keys

Development of Identity-Based Encryption

- The idea of using an arbitrary string as a public key was proposed in 1984 by Shamir
- Since then several constructions for identity-based encryption (IBE) have been proposed, but the first efficient working IBE scheme was published only in 2001
  - it is based on new cryptographic groups called bilinear maps or groups with pairings
- In an IBE scheme, a central trusted authority (TA) generates public parameters and a master key
- A user's identity is used as the public key, and the user obtains the corresponding private key from the TA

Identity-Based Encryption

- An identity-based encryption scheme consists of the following algorithms
  - setup: the TA generates public parameters params and the master key mkey
  - user key generation: when a user with identity ID identifies himself to the TA, the TA computes the private decryption key of the user d_{ID}
    - often the public key of the user is computed as $h(ID)$ and d_{ID} will correspond to $h(ID)$ as well
  - encryption: given a message m, ID, and params, encryption of m for user ID can be computed $c = \mathsf{Enc}_{ID}(m)$
  - decryption: given a ciphertext c encrypted for user ID, params, and d_{ID}, it can be decrypted to recover the message $m = \mathsf{Dec}_{ID}(c)$

Identity-Based Encryption

- We'll study Boneh-Franklin IBE scheme (2001)
- It uses bilinear maps which are defined over elliptic curves
  - instead of using EC notation P, Q, aP, we'll use more familiar notation g, h, g^x
  - let G and G_T be two groups of order q for some large prime q
  - a bilinear map is a function $e : G \times G \to G_T$ with the following properties
    - bilinear: for any $g, h \in G$ and $a, b \in \mathbb{Z}_q^*$, $e(g^a, h^b) = e(g, h)^{ab}$
    - non-degenerate: if g is a generator of G, $e(g, g)$ is a generator of G_T
    - computable: there is an efficient algorithm for computing $e(g, h)$ for any $g, h \in G$

Identity-Based Encryption

- More about bilinear maps
  - bilinear maps can be asymmetric $e : G_1 \times G_2 \rightarrow G_T$, where G_1 and G_2 are two different groups
  - for the purpose of this lecture, we'll use only symmetric groups
  - complexity assumptions in groups with bilinear maps
    - these groups are different from other groups we studied
    - the Computational DH problem is hard in G, but the Decision DH problem is easy in this group
    - given g^a and g^b, it is still difficult to compute g^{ab}
    - given g^a, g^b, and g^c, it is easy to test whether $g^c = g^{ab}$
    - such testing is done as $e(g^a, g^b) \stackrel{?}{=} e(g^c, g)$

Boneh-Franklin IBE Scheme

- A simple version of the Boneh-Franklin IBE scheme
  - setup
    - given a security parameter k, generate a prime q and two groups G and G_T of order q with a bilinear map $e : G \times G \to G_T$
    - choose a generator $g \in G$ and a secret random $s \in \mathbb{Z}_q^*$, compute $h = g^s$
    - choose cryptographic hash functions $H_1 : \{0,1\}^* \to G$ and $H_2 : G_T \to \{0,1\}^n$ for some n
    - the public parameters are $\mathsf{params} = \{q, G, G_T, e, n, g, h, H_1, H_2\}$
    - the master key is $\mathsf{mkey} = s$

Boneh-Franklin IBE Scheme

- Simple Boneh-Franklin IBE scheme (cont.)
  - user key generation
    - for a given string $ID \in \{0,1\}^*$, compute $g_{ID} = H_1(ID)$
    - compute the private key d_{ID} as $d_{ID} = (g_{ID})^s$
  - encryption
    - to encrypt a message $m \in \{0,1\}^n$ under the public key ID, first compute $g_{ID} = H_1(ID)$
    - choose a random $r \in \mathbb{Z}_q$ and set the ciphertext to

$$c = (g^r, m \oplus H_2(y_{ID}^r)), \quad \text{where} \quad y_{ID} = e(g_{ID}, h)$$

Boneh-Franklin IBE Scheme

- Simple Boneh-Franklin IBE scheme (cont.)
  - decryption
    - let $c = (c_1, c_2)$ be a ciphertext encrypted using the public key ID
    - to decrypt c using d_{ID}, compute $m = c_2 \oplus H_2(e(d_{ID}, c_1))$
- Correctness
  - let's see that decryption of an encryption of m indeed yields m

$$\begin{aligned} m &= c_2 \oplus H_2(e(d_{ID}, c_1)) \\ &= m \oplus H_2(y_{ID}^r) \oplus H_2(e(g_{ID}^s, g^r)) \\ &= m \oplus H_2(e(g_{ID}, h)^r) \oplus H_2(e(g_{ID}^s, g^r)) \\ &= m \oplus H_2(e(g_{ID}, g^s)^r) \oplus H_2(e(g_{ID}^s, g^r)) \\ &= m \oplus H_2(e(g_{ID}, g)^{rs}) \oplus H_2(e(g_{ID}, g)^{rs}) = m \end{aligned}$$

Boneh-Franklin IBE Scheme

- Security
  - this scheme is a semantically secure encryption scheme under the chosen plaintext attack
  - its security relies on the bilinear version of the Computational DH problem called Bilinear Diffie-Hellman (BDH) problem
    - given G and G_T of order q with a bilinear map $e : G \times G \to G_T$ and a generator $g \in G$
    - given g^a, g^b, and g^c, compute $e(g, g)^{abc}$
  - it is believed that the BDH problem is hard in these groups
  - security of the scheme holds only in the random oracle model due to the use of hash functions H_1 and H_2
  - this scheme can be modified to be chosen ciphertext secure

Is the PKI Problem Solved?

- Identity-based encryption allows any string to be used as a public key
- But there are still problems
  - since all private keys are known to the TA, a single global setup is not feasible
  - an IBE solution can be setup at an organization level, but not across corporations
  - thus, a user will need to reliably retrieve public parameters associated with another user's public key
- Thus, if IBE schemes are used across different domains, certification at the level of organizations is needed

Is the PKI Problem Solved?

- To limit the power of the TA, Goyal proposed the following solution (2007)
  - for a single public key ID, there are exponentially many corresponding decryption keys d_{ID}
  - when a user obtains her decryption key d_{ID}, the TA doesn't know what key the user obtained
  - this still allows the TA to read messages encrypted for different users
  - but if a corrupt TA issues decryption keys to two different users for the same ID, it is caught with high probability
- This solution still requires the TA to be trusted, but somewhat reduces the trust requirements

Capabilities of IBE Schemes

- Since any string can be used as a public key, it can include more information than a user's ID
  - for example, a key can have a limited validity period if a date is a part of the key
  - suppose that an ID is now "email_address||year"
  - then each year the user with the corresponding email address will request a decryption key that corresponds to that string
  - in general, the sender can compose the public key by including different conditions in it
  - the recipient asks the TA to issue the corresponding decryption key (if the conditions are met)
- Composing public keys in this way has limitations, is there a more flexible way of expressing policies?

Attribute-Based Encryption

- In IBE, decryption keys can be issued on a number of user attributes instead of a single identity
  - such encryption schemes are called attribute-based encryption (ABE) schemes
  - now each user has n descriptive attributes
  - the user obtains a decryption key corresponding to these attributes
  - how the decryption key is formed depends on the type of policies the scheme can support
- In the simplest case, the user is able to decrypt messages encrypted under n attributes if her attributes match the attributes used during encryption
  - this is equivalent to IBE schemes

Attribute-Based Encryption

- ABE schemes exist that support the following policies
  - fuzzy or approximate matching
    - a message is encrypted using n attributes $X = \{x_1, \ldots, x_n\}$
    - a user has a decryption key corresponding to n attributes $Y = \{y_1, \ldots, y_n\}$
    - a user is able to decrypt only if $|X \cap Y| \geq d$, where $1 \geq d \geq n$ is a fixed threshold
      - in other words, X and Y must have at least d elements in common
    - this type of matching is useful, e.g., for biometrics

Attribute-Based Encryption

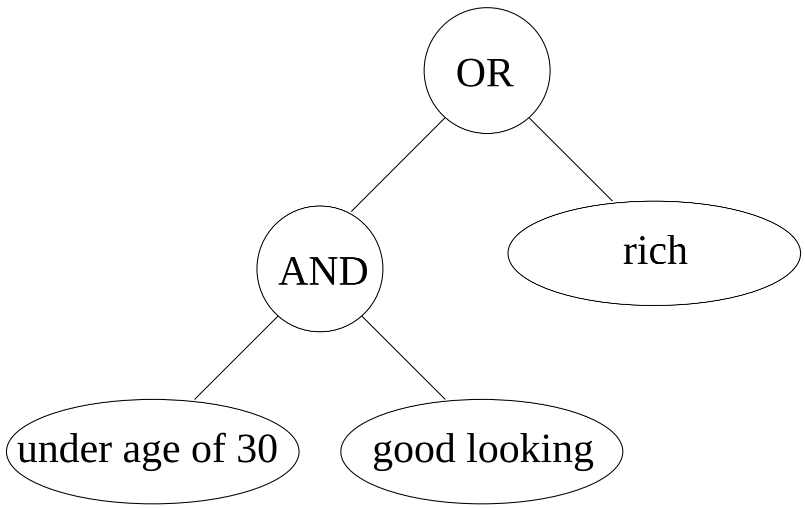
- Policies that ABE schemes can support (cont.)
  - attributes issued by different authorities
    - often, we can have different attributes certified by different authorities
      - e.g., UB certifies that you are a student, DMV certifies that you have a valid driver's license, etc.
    - then it makes sense for parts of your key to be issued by different TAs
    - it turns out that it is possible to do so, but the last TA to issue the key must enforce consistency of the overall key

Attribute-Based Encryption

- Policies that ABE schemes can support (cont.)
  - ciphertext-policy ABE
    - a user still has a decryption key corresponding to her n attributes
    - but now the policies are formulas consisting of attributes, conjunctions (AND), and disjunctions (OR)
    - the ciphertext of a message encodes the sender's policy
    - if the user's attribute satisfy the formula, decryption will be successful
    - example: Alice encrypts her phone number under the following policy and places it on a matching site http://singlebobs.com

Attribute-Based Encryption

- Policies that ABE schemes can support (cont.)
  - example policy that can be encoded in a ciphertext



  - key-policy ABE
    - a ciphertext contains n attributes
    - the policy is encoded in the decryption key

Summary

- Homomorphic encryption allows for computing on encrypted data
  - FHE can be used for securely outsourcing any function
  - other types of HE are often require interactive computation
- Identity-based encryption was proposed as an alternative solution to the PKI problem
  - IBE products are commercially available, but no global infrastructure exists
    - Voltage Security Inc. was founded by the designers of the first practical IBE scheme
  - the expressive power of IBE can be significantly improved through the use of attributes