Lecture 28: Public-key Cryptography

Recall

- In private-key cryptography the secret-key sk is always established ahead of time
- The secrecy of the private-key cryptography relies on the fact that the adversary does not have access to the secret key sk
- For example, consider a private-key encryption scheme
  1. The Alice and Bob generate $\mathsf{sk} \stackrel{\$}{\leftarrow} \mathsf{Gen}()$ ahead of time
  2. Later, when Alice wants to encrypt and send a message to Bob, she computes the cipher-text $c = \mathsf{Enc}_{\mathsf{sk}}(m)$
  3. The adversary see c but gains no additional information about the message m
  4. Bob can decrypt the message $\widetilde{m} = \mathsf{Dec}_{\mathsf{sk}}(c)$
  5. Note that the knowledge of sk distinguishes Bob from the adversary

Perspective

- If $|\mathsf{sk}| \geqslant |m|$, then we can construct private-key encryption schemes (like, one-time pad) that is secure against adversaries with unbounded computational power
- If $|\mathsf{sk}| = O(|m|^{\varepsilon})$, where $\varepsilon \in (0, 1)$ is a constant, then we can construction private-key encryption schemes using pseudorandom generators (PRGs)
- What if, $|\mathsf{sk}| = 0$? That is, what if Alice and Bob never met? How is "Bob" any different from an "adversary"?

In this Lecture

- We shall introduce the Decisional Diffie-Hellmann (DDH) Assumption and the Diffie-Hellman key-exchange protocol,
- We shall introduce the El Gamal (public-key) Encryption Scheme, and
- Finally, abstract out the design principles learned.

Decisional Diffie-Hellman (DDH) Computational Hardness Assumption I

- Let $(G, \circ)$ be a group of size N that is generated by g. We represent it as $(G, \circ) = \langle g \rangle$.
  - We shall represent $g^0 = e$, the identity of the group $(G, \circ)$
  - We shall use the short-hand to represent $g^i = \overbrace{g \circ g \circ \cdots \circ g}^{i\text{-times}}$
  - Then, we have the set $G = \{g^0, g^1, g^2, \dots, g^{N-1}\}$
  - We have already seen how to compute g^a efficiently, for $a \in \{0, 1, \dots, N-1\}$ (Think)
  - We can easily compute the $\text{inv}(g^a)$ (Think)
- Note that we are not providing the entire set G written as a set. This has N entries and is too long (for intuition, think of N as 1024-bit number, so N is roughly 2^{1024}). We only provide a succinct way to generate the group G by providing the generator g

Decisional Diffie-Hellman (DDH) Computational Hardness Assumption II

Definition (Decisional Diffie-Hellman Assumption)

There exists groups $(G, \circ) = \langle g \rangle$ such that no computationally-bounded adversary can efficiently distinguish the following two distributions

- The distribution of (g^a, g^b, g^{ab}), where $a, b \xleftarrow{\$} \{0, 1, \ldots, N-1\}$, and
- The distribution of (g^a, g^b, g^c), where $a, b, c \xleftarrow{\$} \{0, 1, \ldots, N-1\}$

Decisional Diffie-Hellman (DDH) Computational Hardness Assumption III

Remarks:

- Note that DDH Assumption is a "belief" and not a "fact." If it is proven that such groups exist where DDH assumption holds, then this proof will also imply that $P \neq NP$
- We emphasize that the DDH assumption need not hold for *any* group. There are *specially constructed groups* where DDH assumption is believed to hold
- For a fixed value of $A = g^a$ and $B = g^b$, note that there is a unique value of g^{ab}
- The definition, intuitively, states that "Given $A = g^a$ and $B = g^b$, the adversary cannot (efficiently) distinguish g^{ab} from a random $C = g^c$." Alternately, "even given $A = g^a$ and $B = g^b$, the element g^{ab} looks random to a computationally bounded adversary."

Decisional Diffie-Hellman (DDH) Computational Hardness Assumption IV

- Note that it is implicit in the DDH assumption that given $A = g^a$ and g, it is computationally inefficient to compute $a = \log_g A$, i.e., computing the *discrete logarithm* is hard in the group (Think)
- Note that if $a = 0$ (i.e., $A = e$) then it is clear that $g^{ab} = e$ as well. Then the adversary can distinguish between g^{ab} and g^c (random c). But it is unlikely that $a = 0$ (or, $b = 0$) will be chosen. It is possible that there are particular values of a and b when an adversary can distinguish g^{ab} from g^c, but the DDH assumption says that those bad values of a and b are unlikely to be chosen. Thus, it is extremely crucial that a, b are picked at random from the set $\{0, 1, \ldots, N-1\}$

Alice		Bob
$a \xleftarrow{\$} \{0, 1, \ldots, N-1\}$		$b \xleftarrow{\$} \{0, 1, \ldots, N-1\}$
$A = g^a$		$B = g^b$
	$\xleftarrow{\quad B \quad}$	
	$\xrightarrow{\quad A \quad}$	
Compute $\mathrm{sk} = B^a$		Compute $\mathrm{sk} = A^b$

DDH Key-Agreement Protocol II

- Note that both parties can computed the key g^{ab}
- An adversary sees $A = g^a$ and $B = g^b$. From this adversary's perspective, the key g^{ab} is indistinguishable from the random element g^c. So, the key $\mathsf{sk} = g^{ab}$ is perfectly hidden from the adversary

DDH Key-Agreement Protocol III

Remarks.

- Why is this algorithm efficient? Alice can compute A from the generator g and a using the "repeated squaring technique" that you proved in HW0. Similarly, Alice can also compute the key $\mathsf{sk} = B^a$ by repeated squaring technique.
- What advantage does the parties have over the adversary? Alice knows a, therefore she can compute A and B^a efficiently. Bob knows b, therefore he can compute B and A^b efficiently. Adversary, however, only sees A and B, and DDH states that it is computationally infeasible to distinguish g^{ab} from a random group element g^c. Note that if the adversary can compute the discrete log $\log_g A$, then it can easily compute $B^{\log_g A}$, the key.

How to use the Secret Key

- At the end of the Diffie-Hellman key-exchange protocol, Alice and Bob has established a secret key sk that is hidden from the adversary
- Note that Alice and Bob did not have to meet earlier to establish this secret key (contrast this with the private-key encryption scenario, where Alice and Bob have to meet first to establish a secret-key sk)
- Now, we can use the key sk generated by the Diffie-Hellman key-exchange protocol and run any private-key cryptographic primitive using the secret key sk
  - The benefit is that Alice and Bob did not have to meet earlier
  - The downside is that the scheme is secure only against computationally bounded adversaries

ElGamal Public-key Encryption I

Summary of this Scheme. Run the one-time pad private-key encryption over the group $(G, \circ)$ using the key generate by the Diffie-Hellman key-exchange protocol.

ElGamal Public-key Encryption II

Recall the Diffie-Hellman key-exchange protocol.

Alice		Bob
$a \xleftarrow{\$} \{0, 1, \ldots, N-1\}$		$b \xleftarrow{\$} \{0, 1, \ldots, N-1\}$
$A = g^a$		$B = g^b$
	$\xleftarrow{\quad B \quad}$	
	$\xrightarrow{\quad A \quad}$	
Compute sk $= B^a$		Compute sk $= A^b$

ElGamal Public-key Encryption III

- To encrypt a message $m \in G$, Alice encrypts as follows $c = m \circ \mathsf{sk} = m \circ g^{ab}$
- To decrypt a cipher-text $c \in G$, Bob decrypts as follows $\widetilde{m} = c \circ \mathsf{inv}(\mathsf{sk}) = c \circ g^{-ab}$

ElGamal Public-key Encryption IV

We summarize this protocol (ElGamal Encryption) below.

Alice		Bob
		$b \xleftarrow{\$} \{0, 1, \ldots, N-1\}$
		$B = g^b$
	$\xleftarrow{\quad B \quad}$	
$a \xleftarrow{\$} \{0, 1, \ldots, N-1\}$		
$A = g^a$		
Compute $c = m \circ B^a$		
	$\xrightarrow{\quad (A, c) \quad}$	

ElGamal Public-key Encryption V

- The element B sent by Bob is Bob's public-key. It is announced to the world by Bob only once.
- Whoever wants to send an encrypted message to Bob, uses Bob's public-key B
- The pair of elements (A, c) sent by Alice is the cipher-text
- Bob can easily decrypt by computing $\widetilde{m} = c \circ \text{inv}(A^b)$
- The algorithm followed by Alice is her encryption algorithm. To encrypt a new message m', Alice will choose a fresh random a' and compute $A' = g^{a'}$ and $c' = m' \circ B^{a'}$