B504/I538: Introduction to Cryptography

Spring 2017 • Lecture 20

(2017-03-23)
Recall: Modular eth roots

**Defn:** Let \( n \) and \( e \) be positive integers and let \( a \in \mathbb{Z}_n^* \). Then \( b \in \mathbb{Z}_n^* \) is called an \( e \)th root of \( a \) modulo \( n \) if \( b \equiv a^{1/e} \mod n \).

We write \( b \equiv a^{1/e} \mod n \) to indicate that \( b \) is an \( e \)th root of \( a \) modulo \( n \).

**Q:** When does a unique solution for \( a^{1/e} \mod n \) exist?

**A:** If \( \gcd(e, \varphi(n)) = 1 \), then \( a^{1/e} \equiv a^d \mod n \) where \( d \equiv e^{-1} \mod \varphi(n) \).

If \( \gcd(e, \varphi(n)) \neq 1 \), then \( a^{1/e} \) may or may not exist; if it does exist, it is not unique.
Recall: The eth root problem

**Defn:** The eth root problem (aka the RSA problem) is:

Given \((n, e, a)\) such that

1. \(n = pq\) for distinct \(s\)-bit primes \(p\) and \(q\),
2. \(a \in \mathbb{Z}_n^*\), and
3. \(\gcd(e, \varphi(n)) = 1\),

compute \(a^{1/e} \mod n\).

One possible solution: compute \(d := e^{-1} \mod \varphi(n)\) and output \(a^d \mod n\).

**Fact:** Computing \(d := e^{-1} \mod \varphi(n)\) is equivalent to factoring \(n\)!
Recall: Discrete logarithms

Defn: Let $G$ be a group with $|G| = n$ and let $g, h \in G$. A discrete logarithm (DL) of $h$ to the base $g$ in $G$ is a number $x \in \mathbb{Z}_n$ such that $h = g^x$ in $G$.

We write $x = \log_g h$ to indicate that $x$ is a DL of $h$ to the base $g$.

Q: Does the DL of $h$ to the base $g$ always exist?
A: No! But it does always exist if $g$ is a generator of $G$.

Q: If the DL of $h$ to the base $g$ exists, is it unique?
A: Sort of... If $x_1$ and $x_2$ are DLs of $h$ to the base $g$, then $x_1 \equiv x_2 \pmod{|g|}$ (i.e., the DL is unique in $\mathbb{Z}_{|g|}$).
Recall: The DL problem

**Defn:** The DL problem is:

Given \((G, q, g, h)\) such that

1. \((G, \cdot)\) is a cyclic group of \(s\)-bit prime order \(q\),
2. \(g\) is a generator of \(G\), and
3. \(h\) is an arbitrary element of \(G\),

compute \(x = \log_g h\).

- If \((G, +)\) is an additive group, then the DL problem is, given \((G, q, g, h)\) as above, to compute \(k\) such that \(k \cdot g = h\) in \(G\).
Intractable problems

- Intuitively, a problem is intractable if no PPT algorithm can solve a uniform random instance the problem, except with negligible probability.

Q: How do we formalize the notion of intractability for problems defined in a finite group?

A: Very carefully...  
  - Algorithm must be PPT in what parameter?
  - Success probability must be negligible in what parameter?
  - Instance must be uniform random instance from what sample space?
Group generating algorithm

- The DL problem is both defined in a finite group ⇒ no parameter to grow large!
- Problem may be easy in some groups and hard in others

**Def**: A group generating algorithm $G$ is a PPT algorithm that, on input a security parameter $1^s \in 1\mathbb{N}$, outputs a description of a finite group $(G, \cdot)$ with $s$-bit prime order $q$ and a fixed generator $g \in G$.

We write $(G, q, g) \leftarrow G(1^s)$ to indicate that $(G, \cdot)$ is a group with $s$-bit prime order $q$ and generator $g$, sampled from the output of $G$. 
Group generating algorithm

Idea: Don’t consider problem instances in a particular fixed group \((G, \cdot)\); rather, consider instances in groups \((G, q, g) \leftarrow G\) \((1^s)\) sampled from a fixed group generating algorithm \(G\).

Q: How do we formalize the notion of intractability for problems defined in a finite group? 

Q1: Algorithm must be \(\text{PPT}\) in what parameter?
A1: The security parameter \(s\) such that \((G, q, g) \leftarrow G(1^s)\)

Q2: Success probability must be negligible in what parameter?
A2: The security parameter \(s\) such that \((G, q, g) \leftarrow G(1^s)\)

Q3: Instance must be uniform random instance from what sample space?
A3: The set of problem instances in \((G, \cdot)\)
The DL assumption

Defn: Let $G$ be a fixed group generating algorithm. The discrete logarithm (DL) assumption holds with respect to $G$ if, for every PPT algorithm $A$, there exists a negligible function $\epsilon: \mathbb{N} \rightarrow \mathbb{R}^+$ such that $\text{Adv}^{\text{DL}, G}(A) \leq \epsilon(s)$.
Average-case hardness of DL

Thm (informal): Let \((G, \cdot)\) be a group with prime order \(q\). If the DL problem in is hard for some instance, then it is hard for almost every instance!

Proof (sketch): Assume that it is hard to compute \(x \equiv \log_g h\) in \((G, \cdot)\)

Idea: Reduce hardness of computing \(x \equiv \log_g h\) to hardness of solving a uniform random instance in \((G, \cdot)\)

- Randomize \(g\): \((G, q, g, h) \rightarrow (G, q, g^r, h)\) for \(r \in \mathbb{Z}_q^*\)
  \[\Rightarrow \log_g h \equiv (\log_{g^r} h) \cdot r \pmod{q}\]
- Randomize \(h\): \((G, q, g, h) \rightarrow (G, q, g, h^s)\) for \(s \in \mathbb{Z}_q^*\)
  \[\Rightarrow \log_g h \equiv (\log_g h^s) \cdot s^{-1} \pmod{q}\]
RSA instance generating algorithm

**Defn:** An RSA instance generating algorithm $G$ is a PPT algorithm that, on input a security parameter $1^s \in \mathbb{1}^\mathbb{N}$, outputs a pair of distinct $s$-bit primes $(p, q)$ and $e \in \mathbb{Z}_{\phi(pq)}^*$. We write $(p, q, e) \leftarrow G(1^s)$ to indicate that $s$-bit primes $(p, q)$ and $e \in \mathbb{Z}_{\phi(pq)}^*$ are sampled from the output of $G$. 
The _eth root assumption_

**Definition:** Let $G$ be a fixed RSA instance generating algorithm. The _eth root (RSA) assumption_ holds with respect to $G$ if, for every PPT algorithm $A$, there exists a negligible function $\varepsilon : \mathbb{N} \to \mathbb{R}^+$ such that $\text{Adv}^{RSA,G}(A) \leq \varepsilon(s)$.

**Diagram:**
- **Challenger (C):** $(p,q,e) \leftarrow G(1^s)$, $N = p \cdot q$, $a \in \mathbb{Z}_N^*$, $b = a^e \mod N$
- **Attacker (A):** $(N,e,b) \rightarrow (1^s)$
- Define $E$ be the event that $a \equiv a' \mod N$
- Define $A$'s advantage to be $\text{Adv}^{RSA,G}(A) = \Pr[E]$
The RSA trapdoor permutation

Observation: The \( e \)th root assumption is qualitatively different from the DL assumption
- In DL assumption, attacker gets complete output of \( G \)
- In \( e \)th root assumption, attacker learns \( N = p \cdot q \) but not \( p \) and \( q \); if attacker knows \( (p, q) \) the \( e \)th root problem is easy!

- This is our first example of a trapdoor permutation with \( (p, q) \) being the trapdoor
- More on this later...
Diffie-Hellman key exchange protocol

What if Alice and Bob don’t already share a secret key?
Diffie-Hellman key exchange protocol

\begin{align*}
a & \in_R \mathbb{Z}_q^* \\
& : = h((g^b)^a) \\
b & \in_R \mathbb{Z}_q^* \\
& : = h((g^a)^b)
\end{align*}

Suppose \((G, q, g) \leftarrow G(1^s)\) for some group generating algorithm \(G\).

Ryan Henry
Security of Diffie-Hellman key exchange

Q: Suppose the DL assumption holds with respect to \( G \).

Is the Diffie-Hellman protocol “secure” in the presence of an eavesdropper?

A: Errr...well, probably?

- DL assumption implies that it is hard for an eavesdropper to compute \( a \) from \( g^a \) or \( b \) from \( g^b \).
- But, is it hard to compute \( g^{ab} \) from \( g^a \) and \( g^b \)?
Diffie-Hellman assumption

**Definition:** Let $G$ be a group generating algorithm. The (computational) Diffie-Hellman (CDH) assumption holds with respect to $G$ if, for every PPT algorithm $A$, there exists a negligible function $\epsilon : \mathbb{N} \to \mathbb{R}^+$ such that $\text{Adv}_{\text{CDH},G}(A) \leq \epsilon(s)$.

Challenger $(C)$

- $1^s \rightarrow (G,q,g) \leftarrow G(1^s)$
- $a,b \in \mathbb{Z}_q^*$
- $h_1 = g^a$, $h_2 = g^b$

Attacker $(A)$

- $(G,q,g,h_1,h_2) \rightarrow h$

Let $E$ be the event that $h = g^{ab}$

Define $A$'s advantage to be $\text{Adv}_{\text{CDH},G}(A) = \Pr[E]$
Decision Diffie-Hellman assumption

A tuple \((G, q, g, g^a, g^b, h)\) is a DH tuple if and only if \(h = g^{ab}\)

**Game 0**: (input to A is a DH tuple)

- Challenger
- \((G, q, g) \leftarrow G(1^s)\)
- \(a, b \in \mathbb{Z}_q^*\)

- Distinguisher (D)
- \((G, q, g, g^a, g^b, g^{ab})\)

**Game 1**: (input to A is not a DH tuple)

- Challenger
- \((G, q, g) \leftarrow G(1^s)\)
- \(a, b, c \in \mathbb{Z}_q^*\)

- Distinguisher (D)
- \((G, q, g, g^a, g^b, g^c)\)

Let \(E\) be the event that \(b' = 0\) in Game 0 or \(b' = 1\) in Game 1

**Def**: \(\text{Adv}^{\text{DDH}, G}(D) = |\text{Pr}[E] - 1/2|\)
Decision Diffie-Hellman assumption

Defn: Let $G$ be a group generating algorithm. The Decision Diffie-Hellman (DDH) assumption holds with respect to $G$ if, for every PPT algorithm $A$, there exists a negligible function $\varepsilon : \mathbb{N} \rightarrow \mathbb{R}^+$ such that $\text{Adv}^{\text{DDH},G}(A) \leq \varepsilon(s)$. 
The DL, CDH, and DDH assumptions

**Thm:** Let $G$ be a group generating algorithm.

- **Fact 1:** If the DDH assumption holds with respect to $G$, then the CDH assumption also holds with respect to $G$. The converse is believed to be false.

- **Fact 2:** If the CDH assumption holds with respect to $G$, then the DL assumption also holds with respect to $G$. The converse is not known to be true.
Public-key encryption schemes

**Defn:** A public-key encryption scheme is a triple of algorithms \((Gen, Enc, Dec)\) such that

- \(Gen : \mathbb{N} \rightarrow K_e \times K_d\) is a randomized "keypair generation" algorithm;
- \(Enc : K_e \times M \rightarrow C\) is an (often randomized) "encryption" algorithm;
- \(Dec : K_d \times C \rightarrow M\) is a deterministic "decryption" algorithm.

Usually write \(Enc_{k_e}(m)\) and \(Dec_{k_d}(m)\) instead of \(Enc(k_e,m)\) and \(Dec(k_d,m)\).

- \(K_e\) is the encryption key space (set of possible encryption keys)
- \(K_d\) is the decryption key space (set of possible decryption keys)
- \(M\) is the message space (set of possible messages)
- \(C\) is the ciphertext space (set of possible ciphertexts)
Correctness

- Intuitively: Correctness is the property of being able to decrypt (given the appropriate decryption key)

**Defn:** A public-key encryption scheme $(Gen, Enc, Dec)$ with message space $M$ is correct if there exists a negligible function $\varepsilon: \mathbb{N} \rightarrow \mathbb{R}^+$ such that, $\forall s \in \mathbb{N}$ and $\forall m \in M$,

$$
\Pr[Dec_{kd}(Enc_{ke}(m)) = m \mid (k_e, k_d) \leftarrow Gen(1^s)] \geq 1 - \varepsilon(s)
$$
That's all for today, folks!