Cryptography

Lecture 20

Arkady Yerukhimovich

November 6, 2024

Outline

Lecture 19 Review

- Number-Theoretic Hardness Assumptions
- Assumptions in $\mathbb{Z}_N^*$: Factoring, RSA
- Assumptions in Cyclic Groups: DLOG, CDH, DDH

Assumptions in $\mathbb{Z}_N^*$

Factoring Problem

Given $N = pq$ when p and q are n-bit primes, find p and q

Assumptions in $\mathbb{Z}_N^*$

Factoring Problem

Given $N = pq$ when p and q are n-bit primes, find p and q

RSA Problem

Given $(N = pq, e)$ s.t. $gcd(e, \phi(N)) = 1$ and $y \in \mathbb{Z}_N^*$, compute $[y^{1/e} \bmod N]$

Assumptions in Cyclic Groups

Let G be a cyclic group of order q with generator g

Discrete Log Problem

Given $h \in G$, find $0 \leq x \leq q-1$ s.t. $g^x = h$. We say $x = \log_g h$

Assumptions in Cyclic Groups

Let G be a cyclic group of order q with generator g

Discrete Log Problem

Given $h \in G$, find $0 \leq x \leq q - 1$ s.t. $g^x = h$. We say $x = \log_g h$

Computational Diffie-Hellman (CDH) Problem

Given $h_1 = g^x$, $h_2 = g^y$, find g^{xy}

Assumptions in Cyclic Groups

Let G be a cyclic group of order q with generator g

Discrete Log Problem

Given $h \in G$, find $0 \leq x \leq q-1$ s.t. $g^x = h$. We say $x = \log_g h$

Computational Diffie-Hellman (CDH) Problem

Given $h_1 = g^x$, $h_2 = g^y$, find g^{xy}

Decisional Diffie-Hellman (DDH) Problem

Given $h_1 = g^x$, $h_2 = g^y$, distinguish g^{xy} from g^z for $z \leftarrow \mathbb{Z}_q$

Outline

PRG from DDH

Let G be a group of order q with generator g

PRG from DDH

Let G be a group of order q with generator g

PRG Construction

Given $s = (x, y) \leftarrow \mathbb{Z}_q^2$, output $G(s) = (g^x, g^y, g^{xy})$

PRG from DDH

Let G be a group of order q with generator g

PRG Construction

Given $s = (x, y) \leftarrow \mathbb{Z}_q^2$, output $G(s) = (g^x, g^y, g^{xy})$

- Expansion:
  - s consists of two random integers in $\mathbb{Z}_q$
  - $G(s)$ outputs 3 field elements in G of order q
  - So $|G(s)| > |s|$

PRG from DDH

Let G be a group of order q with generator g

PRG Construction

Given $s = (x, y) \leftarrow \mathbb{Z}_q^2$, output $G(s) = (g^x, g^y, g^{xy})$

- Expansion:
  - s consists of two random integers in $\mathbb{Z}_q$
  - $G(s)$ outputs 3 field elements in G of order q
  - So $|G(s)| > |s|$
- Pseudorandomness:

PRG from DDH

Let G be a group of order q with generator g

PRG Construction

Given $s = (x, y) \leftarrow \mathbb{Z}_q^2$, output $G(s) = (g^x, g^y, g^{xy})$

- Expansion:
  - s consists of two random integers in $\mathbb{Z}_q$
  - $G(s)$ outputs 3 field elements in G of order q
  - So $|G(s)| > |s|$
- Pseudorandomness:
  - g^x, g^y are random group elements

PRG from DDH

Let G be a group of order q with generator g

PRG Construction

Given $s = (x, y) \leftarrow \mathbb{Z}_q^2$, output $G(s) = (g^x, g^y, g^{xy})$

- Expansion:
  - s consists of two random integers in $\mathbb{Z}_q$
  - $G(s)$ outputs 3 field elements in G of order q
  - So $|G(s)| > |s|$
- Pseudorandomness:
  - g^x, g^y are random group elements
  - If $\mathcal{A}_c$ can distinguish (g^x, g^y, g^{xy}) from random, then $\mathcal{A}_r$ just runs $\mathcal{A}_c$ to break DDH

Fixed-length Compression-Function from DLog

Let G be a group of order q with generator g

Fixed-length Compression-Function from DLog

Let G be a group of order q with generator g

CRHF Construction

Fixed-length Compression-Function from DLog

Let G be a group of order q with generator g

CRHF Construction

Gen_H:

- Pick $h \leftarrow G$
- Output $s = (G, q, g, h)$

Fixed-length Compression-Function from DLog

Let G be a group of order q with generator g

CRHF Construction

Gen_H:

- Pick $h \leftarrow G$
- Output $s = (G, q, g, h)$

H:

- Given $s = (G, q, g, h)$ and input $(x, y) \in \mathbb{Z}_q^2$, compute

$$H^s(x, y) = g^x h^y$$

Fixed-length Compression-Function from DLog

Let G be a group of order q with generator g

CRHF Construction

Gen_H:

- Pick $h \leftarrow G$
- Output $s = (G, q, g, h)$

H:

- Given $s = (G, q, g, h)$ and input $(x, y) \in \mathbb{Z}_q^2$, compute

$$H^s(x, y) = g^x h^y$$

Proof of security:

Fixed-length Compression-Function from DLog

Let G be a group of order q with generator g

CRHF Construction

Gen_H:

- Pick $h \leftarrow G$
- Output $s = (G, q, g, h)$

H:

- Given $s = (G, q, g, h)$ and input $(x, y) \in \mathbb{Z}_q^2$, compute

$$H^s(x, y) = g^x h^y$$

Proof of security:

- Suppose $\mathcal{A}_c$ can find (x, y) and (x', y') such that $g^x h^y = g^{x'} h^{y'}$

Fixed-length Compression-Function from DLog

Let G be a group of order q with generator g

CRHF Construction

Gen_H:

- Pick $h \leftarrow G$
- Output $s = (G, q, g, h)$

H:

- Given $s = (G, q, g, h)$ and input $(x, y) \in \mathbb{Z}_q^2$, compute

$$H^s(x, y) = g^x h^y$$

Proof of security:

- Suppose $\mathcal{A}_c$ can find (x, y) and (x', y') such that $g^x h^y = g^{x'} h^{y'}$
- Let $h = g^a$, then $H^s(x, y) = g^{x+ay}$ and $H^s(x', y') = g^{x'+ay'}$

Fixed-length Compression-Function from DLog

Let G be a group of order q with generator g

CRHF Construction

Gen_H:

- Pick $h \leftarrow G$
- Output $s = (G, q, g, h)$

H:

- Given $s = (G, q, g, h)$ and input $(x, y) \in \mathbb{Z}_q^2$, compute

$$H^s(x, y) = g^x h^y$$

Proof of security:

- Suppose $\mathcal{A}_c$ can find (x, y) and (x', y') such that $g^x h^y = g^{x'} h^{y'}$
- Let $h = g^a$, then $H^s(x, y) = g^{x+ay}$ and $H^s(x', y') = g^{x'+ay'}$
- Since these are equal, we have $x + ay = x' + ay'$

Fixed-length Compression-Function from DLog

Let G be a group of order q with generator g

CRHF Construction

Gen_H:

- Pick $h \leftarrow G$
- Output $s = (G, q, g, h)$

H:

- Given $s = (G, q, g, h)$ and input $(x, y) \in \mathbb{Z}_q^2$, compute

$$H^s(x, y) = g^x h^y$$

Proof of security:

- Suppose $\mathcal{A}_c$ can find (x, y) and (x', y') such that $g^x h^y = g^{x'} h^{y'}$
- Let $h = g^a$, then $H^s(x, y) = g^{x+ay}$ and $H^s(x', y') = g^{x'+ay'}$
- Since these are equal, we have $x + ay = x' + ay'$
- So, $\mathcal{A}_r$ computes $a = \frac{x-x'}{y'-y}$ breaking DLog of h

PRF from DDH (Naor-Reingold PRF)

Let G be a group of order q with generator g

PRF from DDH (Naor-Reingold PRF)

Let G be a group of order q with generator g

PRF Construction

Let $k = (a_0, a_1, \ldots, a_n) \leftarrow \mathbb{Z}_q^n$. On input $x \in \{0,1\}^n$

$$F_k(x) = g^{a_0 \prod_{i=1}^{n} a_i^{x_i}} \qquad (x_i = i^{th} \text{ bit of } x)$$

PRF from DDH (Naor-Reingold PRF)

Let G be a group of order q with generator g

PRF Construction

Let $k = (a_0, a_1, \ldots, a_n) \leftarrow \mathbb{Z}_q^n$. On input $x \in \{0,1\}^n$

$$F_k(x) = g^{a_0 \prod_{i=1}^n a_i^{x_i}} \qquad (x_i = i^{th} \text{ bit of } x)$$

Pseudorandomness (Intuition):

PRF from DDH (Naor-Reingold PRF)

Let G be a group of order q with generator g

PRF Construction

Let $k = (a_0, a_1, \ldots, a_n) \leftarrow \mathbb{Z}_q^n$. On input $x \in \{0,1\}^n$

$$F_k(x) = g^{a_0 \prod_{i=1}^{n} a_i^{x_i}} \qquad (x_i = i^{th} \text{ bit of } x)$$

Pseudorandomness (Intuition):

- For every new input x', $F_k(x')$ differs from $F_k(x)$ by at least one term in the exponent

PRF from DDH (Naor-Reingold PRF)

Let G be a group of order q with generator g

PRF Construction

Let $k = (a_0, a_1, \ldots, a_n) \leftarrow \mathbb{Z}_q^n$. On input $x \in \{0,1\}^n$

$$F_k(x) = g^{a_0 \prod_{i=1}^{n} a_i^{x_i}} \qquad (x_i = i^{th} \text{ bit of } x)$$

Pseudorandomness (Intuition):

- For every new input x', $F_k(x')$ differs from $F_k(x)$ by at least one term in the exponent
- By DDH, we cannot distinguish such terms from random

Outline

The Biggest Challenge of Using Private-Key Crypto

The Biggest Challenge of Using Private-Key Crypto

Sharing Keys

To use private-key crypto, every pair of parties needs to privately share a secret key.

The Biggest Challenge of Using Private-Key Crypto

Sharing Keys

To use private-key crypto, every pair of parties needs to privately share a secret key.

Resulting Challenges:

- Key distribution – how to share the keys in the first place

The Biggest Challenge of Using Private-Key Crypto

Sharing Keys

To use private-key crypto, every pair of parties needs to privately share a secret key.

Resulting Challenges:

- Key distribution – how to share the keys in the first place
- Key storage and management – many keys to store

The Biggest Challenge of Using Private-Key Crypto

Sharing Keys

To use private-key crypto, every pair of parties needs to privately share a secret key.

Resulting Challenges:

- Key distribution – how to share the keys in the first place
- Key storage and management – many keys to store
- “Open systems” – users don’t know each other (e.g., shopping on Amazon)

Solution 1: Key-distribution Centers (KDC)

Assume a trusted KDC that shares a secret-key with each user

Solution 1: Key-distribution Centers (KDC)

Assume a trusted KDC that shares a secret-key with each user

Needham-Schroeder / Kerberos Protocol

Suppose A wants to talk privately to B:

Solution 1: Key-distribution Centers (KDC)

Assume a trusted KDC that shares a secret-key with each user

Needham-Schroeder / Kerberos Protocol

Suppose A wants to talk privately to B:

1. A sends authenticated message to KDC (using k^A_{KDC})

Solution 1: Key-distribution Centers (KDC)

Assume a trusted KDC that shares a secret-key with each user

Needham-Schroeder / Kerberos Protocol

Suppose A wants to talk privately to B:

1. A sends authenticated message to KDC (using k^A_{KDC})
2. KDC generates $k \leftarrow \mathsf{Gen}$, produces authenticated encryptions $\mathsf{Enc}_{k^A_{KDC}}(k)$ and $\mathsf{Enc}_{k^B_{KDC}}(k)$, and sends both to A.

Solution 1: Key-distribution Centers (KDC)

Assume a trusted KDC that shares a secret-key with each user

Needham-Schroeder / Kerberos Protocol

Suppose A wants to talk privately to B:

1. A sends authenticated message to KDC (using k^A_{KDC})
2. KDC generates $k \leftarrow \mathsf{Gen}$, produces authenticated encryptions $\mathsf{Enc}_{k^A_{KDC}}(k)$ and $\mathsf{Enc}_{k^B_{KDC}}(k)$, and sends both to A.
3. A sends $\mathsf{Enc}_{k^B_{KDC}}(k)$ to B

Solution 1: Key-distribution Centers (KDC)

Assume a trusted KDC that shares a secret-key with each user

Needham-Schroeder / Kerberos Protocol

Suppose A wants to talk privately to B:

1. A sends authenticated message to KDC (using k_{KDC}^{A})
2. KDC generates $k \leftarrow \text{Gen}$, produces authenticated encryptions $\text{Enc}_{k_{KDC}^{A}}(k)$ and $\text{Enc}_{k_{KDC}^{B}}(k)$, and sends both to A.
3. A sends $\text{Enc}_{k_{KDC}^{B}}(k)$ to B
4. A and B now have shared key k

Solution 1: Key-distribution Centers (KDC)

Assume a trusted KDC that shares a secret-key with each user

Needham-Schroeder / Kerberos Protocol

Suppose A wants to talk privately to B:

1. A sends authenticated message to KDC (using k^A_{KDC})
2. KDC generates $k \leftarrow \mathrm{Gen}$, produces authenticated encryptions $\mathrm{Enc}_{k^A_{KDC}}(k)$ and $\mathrm{Enc}_{k^B_{KDC}}(k)$, and sends both to A.
3. A sends $\mathrm{Enc}_{k^B_{KDC}}(k)$ to B
4. A and B now have shared key k

Pros:

Solution 1: Key-distribution Centers (KDC)

Assume a trusted KDC that shares a secret-key with each user

Needham-Schroeder / Kerberos Protocol

Suppose A wants to talk privately to B:

1. A sends authenticated message to KDC (using k^A_{KDC})
2. KDC generates $k \leftarrow \mathsf{Gen}$, produces authenticated encryptions $\mathsf{Enc}_{k^A_{KDC}}(k)$ and $\mathsf{Enc}_{k^B_{KDC}}(k)$, and sends both to A.
3. A sends $\mathsf{Enc}_{k^B_{KDC}}(k)$ to B
4. A and B now have shared key k

Pros:

- Allows parties to share keys using only private-key crypto

Solution 1: Key-distribution Centers (KDC)

Assume a trusted KDC that shares a secret-key with each user

Needham-Schroeder / Kerberos Protocol

Suppose A wants to talk privately to B:

1. A sends authenticated message to KDC (using k^A_{KDC})
2. KDC generates $k \leftarrow \mathsf{Gen}$, produces authenticated encryptions $\mathsf{Enc}_{k^A_{KDC}}(k)$ and $\mathsf{Enc}_{k^B_{KDC}}(k)$, and sends both to A.
3. A sends $\mathsf{Enc}_{k^B_{KDC}}(k)$ to B
4. A and B now have shared key k

Pros:

- Allows parties to share keys using only private-key crypto
- Works well in organizations where KDC is centralized admin

Solution 1: Key-distribution Centers (KDC)

Assume a trusted KDC that shares a secret-key with each user

Needham-Schroeder / Kerberos Protocol

Suppose A wants to talk privately to B:

1. A sends authenticated message to KDC (using k^A_{KDC})
2. KDC generates $k \leftarrow \text{Gen}$, produces authenticated encryptions $\text{Enc}_{k^A_{KDC}}(k)$ and $\text{Enc}_{k^B_{KDC}}(k)$, and sends both to A.
3. A sends $\text{Enc}_{k^B_{KDC}}(k)$ to B
4. A and B now have shared key k

Pros:

- Allows parties to share keys using only private-key crypto
- Works well in organizations where KDC is centralized admin

Cons:

Solution 1: Key-distribution Centers (KDC)

Assume a trusted KDC that shares a secret-key with each user

Needham-Schroeder / Kerberos Protocol

Suppose A wants to talk privately to B:

1. A sends authenticated message to KDC (using k^A_{KDC})
2. KDC generates $k \leftarrow \mathsf{Gen}$, produces authenticated encryptions $\mathsf{Enc}_{k^A_{KDC}}(k)$ and $\mathsf{Enc}_{k^B_{KDC}}(k)$, and sends both to A.
3. A sends $\mathsf{Enc}_{k^B_{KDC}}(k)$ to B
4. A and B now have shared key k

Pros:

- Allows parties to share keys using only private-key crypto
- Works well in organizations where KDC is centralized admin

Cons:

- At some point, all parties need to have a secure channel to KDC

Solution 1: Key-distribution Centers (KDC)

Assume a trusted KDC that shares a secret-key with each user

Needham-Schroeder / Kerberos Protocol

Suppose A wants to talk privately to B:

1. A sends authenticated message to KDC (using k_{KDC}^{A})
2. KDC generates $k \leftarrow \mathsf{Gen}$, produces authenticated encryptions $\mathsf{Enc}_{k_{KDC}^{A}}(k)$ and $\mathsf{Enc}_{k_{KDC}^{B}}(k)$, and sends both to A.
3. A sends $\mathsf{Enc}_{k_{KDC}^{B}}(k)$ to B
4. A and B now have shared key k

Pros:

- Allows parties to share keys using only private-key crypto
- Works well in organizations where KDC is centralized admin

Cons:

- At some point, all parties need to have a secure channel to KDC
- Does not work with open systems – KDC must know all users

Solution 2: Key Exchange

Goals:

- Two parties A and B do not share any apriori secrets

Solution 2: Key Exchange

Goals:

- Two parties A and B do not share any apriori secrets
- A and B run a protocol Π after which both of them output a shared (private) key k

Solution 2: Key Exchange

Goals:

- Two parties A and B do not share any apriori secrets
- A and B run a protocol Π after which both of them output a shared (private) key k

Key Exchange Security

Eavesdropper observing Π should not be able to learn anything about the agreed upon private-key k.

Solution 2: Key Exchange

Goals:

- Two parties A and B do not share any apriori secrets
- A and B run a protocol Π after which both of them output a shared (private) key k

Key Exchange Security

Eavesdropper observing Π should not be able to learn anything about the agreed upon private-key k.

Observations:

- Since A and B do not share a secret, not clear how to use private-key crypto

Solution 2: Key Exchange

Goals:

- Two parties A and B do not share any apriori secrets
- A and B run a protocol Π after which both of them output a shared (private) key k

Key Exchange Security

Eavesdropper observing Π should not be able to learn anything about the agreed upon private-key k.

Observations:

- Since A and B do not share a secret, not clear how to use private-key crypto
- Eavesdropper sees everything sent, no secret channel

Solution 2: Key Exchange

Goals:

- Two parties A and B do not share any apriori secrets
- A and B run a protocol Π after which both of them output a shared (private) key k

Key Exchange Security

Eavesdropper observing Π should not be able to learn anything about the agreed upon private-key k.

Observations:

- Since A and B do not share a secret, not clear how to use private-key crypto
- Eavesdropper sees everything sent, no secret channel

The Power of Key Exchange

Key agreement allows generation of shared secrets without private communication.

Defining Key Exchange Security

Let Π be a protocol between A and B, and let trans be the full transcript of all messages sent by Π.

Defining Key Exchange Security

Let Π be a protocol between A and B, and let trans be the full transcript of all messages sent by Π.

Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\mathsf{KE}^{eav}_{\mathcal{A},\Pi}(n)$

Defining Key Exchange Security

Let Π be a protocol between A and B, and let trans be the full transcript of all messages sent by Π.

Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\mathsf{KE}^{eav}_{\mathcal{A},\Pi}(n)$

- $A(1^n)$ and $B(1^n)$ run Π, resulting in transcript trans and output k

Defining Key Exchange Security

Let Π be a protocol between A and B, and let trans be the full transcript of all messages sent by Π.

Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\mathsf{KE}^{eav}_{\mathcal{A},\Pi}(n)$

- $A(1^n)$ and $B(1^n)$ run Π, resulting in transcript trans and output k
- Challenger chooses $b \leftarrow \{0,1\}$
  - If $b = 0$, he sets $\hat{k} = k$
  - If $b = 1$, he sets $\hat{k} \leftarrow \{0,1\}^n$

Defining Key Exchange Security

Let Π be a protocol between A and B, and let trans be the full transcript of all messages sent by Π.

Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\mathsf{KE}^{eav}_{\mathcal{A},\Pi}(n)$

- $A(1^n)$ and $B(1^n)$ run Π, resulting in transcript trans and output k
- Challenger chooses $b \leftarrow \{0,1\}$
  - If $b = 0$, he sets $\hat{k} = k$
  - If $b = 1$, he sets $\hat{k} \leftarrow \{0,1\}^n$
- $\mathcal{A}$ gets trans, $\hat{k}$ and outputs a guess bit b'

Defining Key Exchange Security

Let Π be a protocol between A and B, and let trans be the full transcript of all messages sent by Π.

Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\mathsf{KE}^{eav}_{\mathcal{A},\Pi}(n)$

- $A(1^n)$ and $B(1^n)$ run Π, resulting in transcript trans and output k
- Challenger chooses $b \leftarrow \{0,1\}$
  - If $b = 0$, he sets $\hat{k} = k$
  - If $b = 1$, he sets $\hat{k} \leftarrow \{0,1\}^n$
- $\mathcal{A}$ gets trans, $\hat{k}$ and outputs a guess bit b'
- We say that $\mathsf{KE}^{eav}_{\mathcal{A},\Pi}(n) = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

Defining Key Exchange Security

Let Π be a protocol between A and B, and let trans be the full transcript of all messages sent by Π.

Consider the following game between an adversary $\mathcal{A}$ and a challenger:

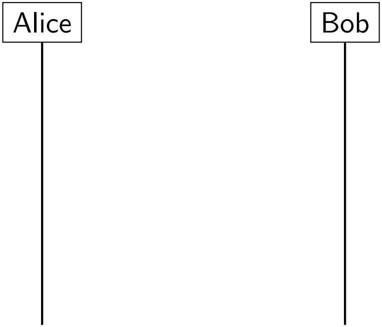
$\mathsf{KE}^{eav}_{\mathcal{A},\Pi}(n)$

- $A(1^n)$ and $B(1^n)$ run Π, resulting in transcript trans and output k
- Challenger chooses $b \leftarrow \{0,1\}$
  - If $b = 0$, he sets $\hat{k} = k$
  - If $b = 1$, he sets $\hat{k} \leftarrow \{0,1\}^n$
- $\mathcal{A}$ gets trans, $\hat{k}$ and outputs a guess bit b'
- We say that $\mathsf{KE}^{eav}_{\mathcal{A},\Pi}(n) = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

Definition: A key exchange protocol Π is secure against an eavesdropper if for all PPT $\mathcal{A}$ it holds that

$$\Pr[\mathsf{KE}^{eav}_{\mathcal{A},\Pi}(n) = 1] \leq 1/2 + \mathsf{negl}(n)$$

Diffie-Hellman Key Exchange



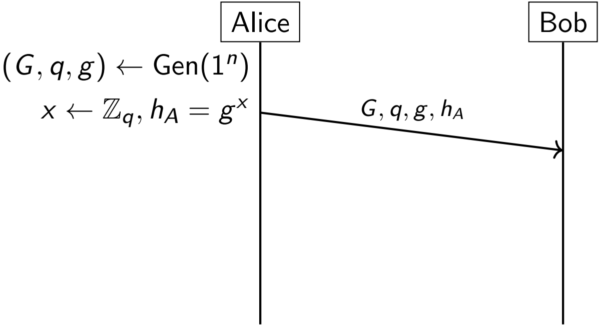
Diffie-Hellman Key Exchange

Alice	Bob

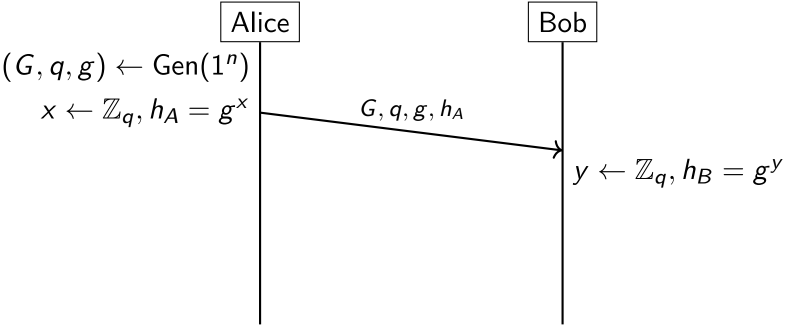
$(G, q, g) \leftarrow \mathsf{Gen}(1^n)$

$x \leftarrow \mathbb{Z}_q, h_A = g^x$

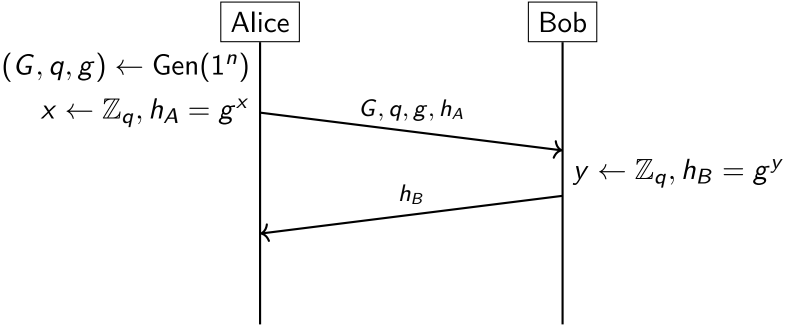
Diffie-Hellman Key Exchange



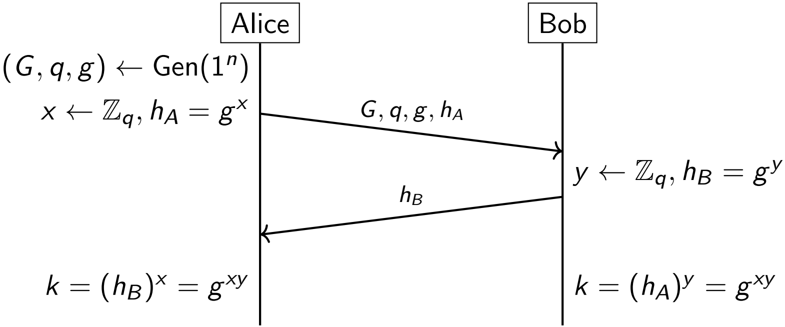
Diffie-Hellman Key Exchange



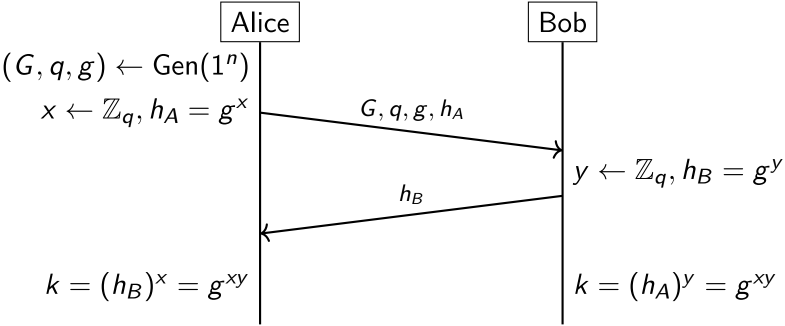
Diffie-Hellman Key Exchange



Diffie-Hellman Key Exchange



Diffie-Hellman Key Exchange



Correctness

Easy to see that A and B output the same key k

Diffie-Hellman Key Exchange – Proof of Security

What $\mathcal{A}_c$ sees:

Diffie-Hellman Key Exchange – Proof of Security

What $\mathcal{A}_c$ sees:

- trans $= (G, q, g), g^x, g^y$

Diffie-Hellman Key Exchange – Proof of Security

What $\mathcal{A}_c$ sees:

- trans $= (G, q, g), g^x, g^y$
- $\hat{k}$ which is either g^{xy} or $\hat{k} \leftarrow G$.

Diffie-Hellman Key Exchange – Proof of Security

What $\mathcal{A}_c$ sees:

- trans $= (G, q, g), g^x, g^y$
- $\hat{k}$ which is either g^{xy} or $\hat{k} \leftarrow G$.

Construct $\mathcal{A}_r$ breaking DDH:

Diffie-Hellman Key Exchange – Proof of Security

What $\mathcal{A}_c$ sees:

- trans $= (G, q, g), g^x, g^y$
- $\hat{k}$ which is either g^{xy} or $\hat{k} \leftarrow G$.

Construct $\mathcal{A}_r$ breaking DDH:

- $\mathcal{A}_r$ receives as input either $(G, q, g, g^x, g^y, c = g^{xy})$ or $(G, q, g, g^x, g^y, c = g^z)$

Diffie-Hellman Key Exchange – Proof of Security

What $\mathcal{A}_c$ sees:

- trans $= (G, q, g), g^x, g^y$
- $\hat{k}$ which is either g^{xy} or $\hat{k} \leftarrow G$.

Construct $\mathcal{A}_r$ breaking DDH:

- $\mathcal{A}_r$ receives as input either $(G, q, g, g^x, g^y, c = g^{xy})$ or $(G, q, g, g^x, g^y, c = g^z)$
- He plays the roles of A and B, producing transcript trans, sets $\hat{k} = c$ and gives both to $\mathcal{A}_c$

Diffie-Hellman Key Exchange – Proof of Security

What $\mathcal{A}_c$ sees:

- trans $= (G, q, g), g^x, g^y$
- $\hat{k}$ which is either g^{xy} or $\hat{k} \leftarrow G$.

Construct $\mathcal{A}_r$ breaking DDH:

- $\mathcal{A}_r$ receives as input either $(G, q, g, g^x, g^y, c = g^{xy})$ or $(G, q, g, g^x, g^y, c = g^z)$
- He plays the roles of A and B, producing transcript trans, sets $\hat{k} = c$ and gives both to $\mathcal{A}_c$
- When $\mathcal{A}_c$ outputs a bit b, $\mathcal{A}_r$ outputs the same bit.

Diffie-Hellman Key Exchange – Proof of Security

What $\mathcal{A}_c$ sees:

- trans $= (G, q, g), g^x, g^y$
- $\hat{k}$ which is either g^{xy} or $\hat{k} \leftarrow G$.

Construct $\mathcal{A}_r$ breaking DDH:

- $\mathcal{A}_r$ receives as input either $(G, q, g, g^x, g^y, c = g^{xy})$ or $(G, q, g, g^x, g^y, c = g^z)$
- He plays the roles of A and B, producing transcript trans, sets $\hat{k} = c$ and gives both to $\mathcal{A}_c$
- When $\mathcal{A}_c$ outputs a bit b, $\mathcal{A}_r$ outputs the same bit.

Analysis: This is a perfect simulation of the DDH security game

Diffie-Hellman Key Exchange – Proof of Security

What $\mathcal{A}_c$ sees:

- trans $= (G, q, g), g^x, g^y$
- $\hat{k}$ which is either g^{xy} or $\hat{k} \leftarrow G$.

Construct $\mathcal{A}_r$ breaking DDH:

- $\mathcal{A}_r$ receives as input either $(G, q, g, g^x, g^y, c = g^{xy})$ or $(G, q, g, g^x, g^y, c = g^z)$
- He plays the roles of A and B, producing transcript trans, sets $\hat{k} = c$ and gives both to $\mathcal{A}_c$
- When $\mathcal{A}_c$ outputs a bit b, $\mathcal{A}_r$ outputs the same bit.

Analysis: This is a perfect simulation of the DDH security game

Observations:

Diffie-Hellman Key Exchange – Proof of Security

What $\mathcal{A}_c$ sees:

- trans $= (G, q, g), g^x, g^y$
- $\hat{k}$ which is either g^{xy} or $\hat{k} \leftarrow G$.

Construct $\mathcal{A}_r$ breaking DDH:

- $\mathcal{A}_r$ receives as input either $(G, q, g, g^x, g^y, c = g^{xy})$ or $(G, q, g, g^x, g^y, c = g^z)$
- He plays the roles of A and B, producing transcript trans, sets $\hat{k} = c$ and gives both to $\mathcal{A}_c$
- When $\mathcal{A}_c$ outputs a bit b, $\mathcal{A}_r$ outputs the same bit.

Analysis: This is a perfect simulation of the DDH security game

Observations:

- In KE definition, we chose $\hat{k} \leftarrow \{0, 1\}^n$, while here we chose $\hat{k} \leftarrow G$

Diffie-Hellman Key Exchange – Proof of Security

What $\mathcal{A}_c$ sees:

- $\text{trans} = (G, q, g), g^x, g^y$
- $\hat{k}$ which is either g^{xy} or $\hat{k} \leftarrow G$.

Construct $\mathcal{A}_r$ breaking DDH:

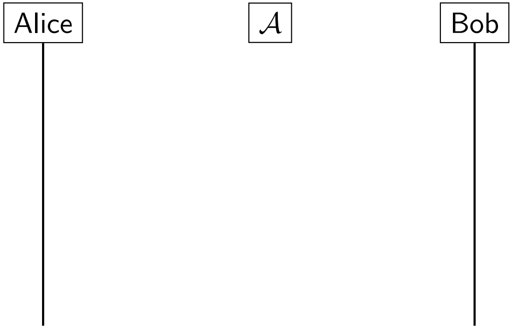
- $\mathcal{A}_r$ receives as input either $(G, q, g, g^x, g^y, c = g^{xy})$ or $(G, q, g, g^x, g^y, c = g^z)$
- He plays the roles of A and B, producing transcript trans, sets $\hat{k} = c$ and gives both to $\mathcal{A}_c$
- When $\mathcal{A}_c$ outputs a bit b, $\mathcal{A}_r$ outputs the same bit.

Analysis: This is a perfect simulation of the DDH security game

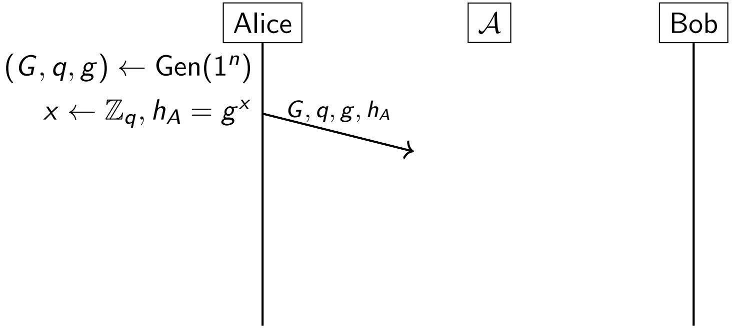
Observations:

- In KE definition, we chose $\hat{k} \leftarrow \{0, 1\}^n$, while here we chose $\hat{k} \leftarrow G$
- Can use hash to convert random group element to a random string

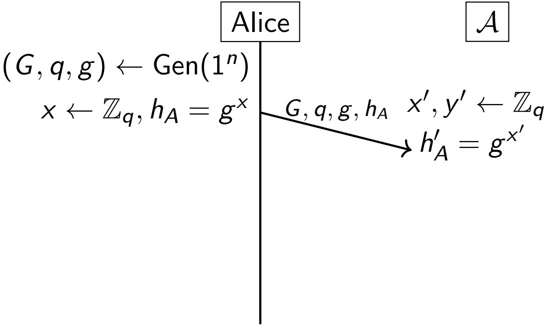
Man-in-the-middle Attack



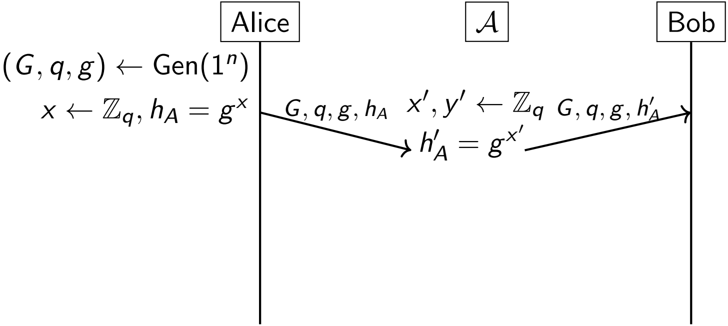
Man-in-the-middle Attack



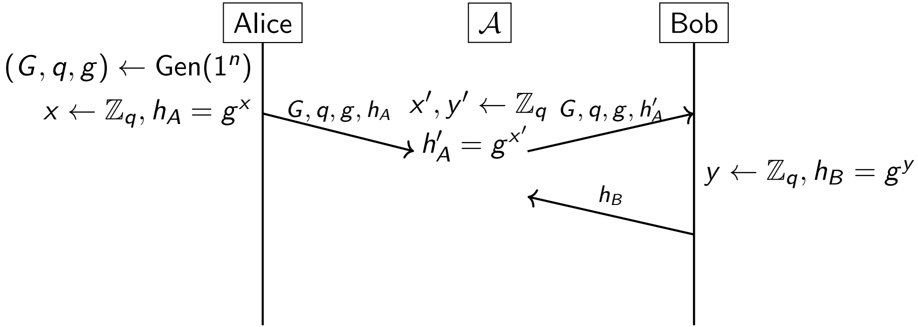
Man-in-the-middle Attack



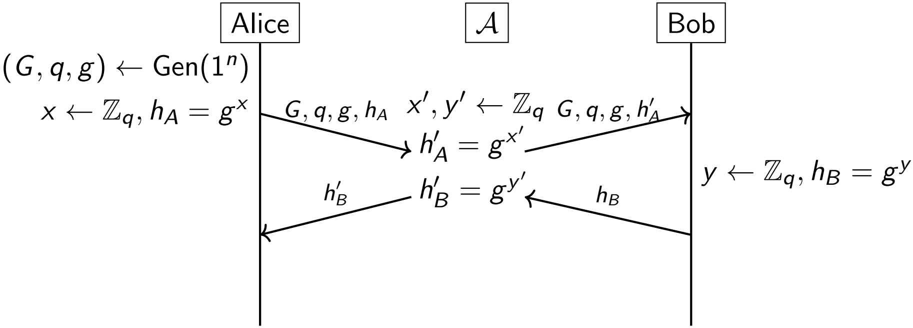
Man-in-the-middle Attack



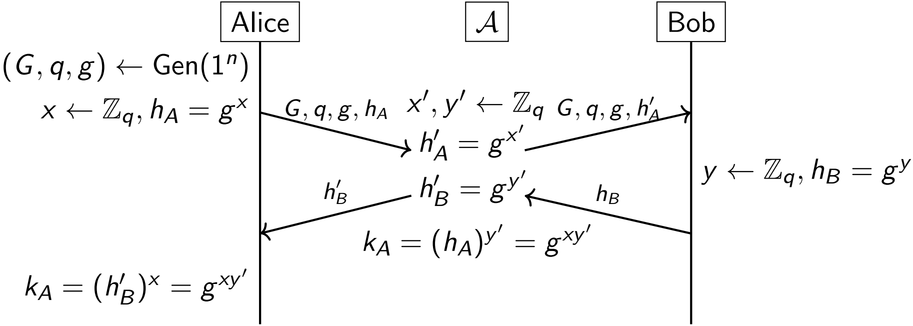
Man-in-the-middle Attack



Man-in-the-middle Attack



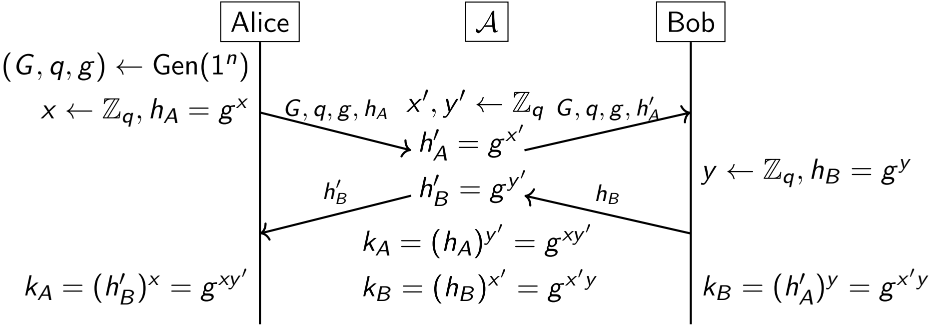
Man-in-the-middle Attack



Man-in-the-middle Attack

Alice	$\mathcal{A}$	Bob
$(G, q, g) \leftarrow \mathsf{Gen}(1^n)$		
$x \leftarrow \mathbb{Z}_q, h_A = g^x$	$x', y' \leftarrow \mathbb{Z}_q$	
$\xrightarrow{G, q, g, h_A}$	$h'_A = g^{x'}$	$\xrightarrow{G, q, g, h'_A}$
		$y \leftarrow \mathbb{Z}_q, h_B = g^y$
$\xleftarrow{h'_B}$	$h'_B = g^{y'}$	$\xleftarrow{h_B}$
	$k_A = (h_A)^{y'} = g^{xy'}$	
$k_A = (h'_B)^x = g^{xy'}$	$k_B = (h_B)^{x'} = g^{x'y}$	$k_B = (h'_A)^y = g^{x'y}$

Man-in-the-middle Attack



Result

- $k_A \neq k_B$ – A and B fail to agree on a key
- $\mathcal{A}$ has shared keys with both

Going Beyond Key Exchange

	Private-Key	Public-Key
Secrecy	Private-key encryption	Public-key encryption
Integrity	MACs	Digital signatures

Going Beyond Key Exchange

	Private-Key	Public-Key
Secrecy	Private-key encryption	Public-key encryption
Integrity	MACs	Digital signatures

Public-Key Encryption

Going Beyond Key Exchange

	Private-Key	Public-Key
Secrecy	Private-key encryption	Public-key encryption
Integrity	MACs	Digital signatures

Public-Key Encryption

- User A has keys (pk_A, sk_A)

Going Beyond Key Exchange

	Private-Key	Public-Key
Secrecy	Private-key encryption	Public-key encryption
Integrity	MACs	Digital signatures

Public-Key Encryption

- User A has keys (pk_A, sk_A)
- Public key pk_A is used to encrypt messages to A

Going Beyond Key Exchange

	Private-Key	Public-Key
Secrecy	Private-key encryption	Public-key encryption
Integrity	MACs	Digital signatures

Public-Key Encryption

- User A has keys (pk_A, sk_A)
- Public key pk_A is used to encrypt messages to A
- Secret key sk_A is used by A to decrypt

Going Beyond Key Exchange

	Private-Key	Public-Key
Secrecy	Private-key encryption	Public-key encryption
Integrity	MACs	Digital signatures

Public-Key Encryption

- User A has keys (pk_A, sk_A)
- Public key pk_A is used to encrypt messages to A
- Secret key sk_A is used by A to decrypt
- A publishes pk_A while keeping sk_A secret

Going Beyond Key Exchange

	Private-Key	Public-Key
Secrecy	Private-key encryption	Public-key encryption
Integrity	MACs	Digital signatures

Public-Key Encryption

- User A has keys (pk_A, sk_A)
- Public key pk_A is used to encrypt messages to A
- Secret key sk_A is used by A to decrypt
- A publishes pk_A while keeping sk_A secret
- Anybody can encrypt, only A can decrypt

Going Beyond Key Exchange

	Private-Key	Public-Key
Secrecy	Private-key encryption	Public-key encryption
Integrity	MACs	Digital signatures

Public-Key Encryption

- User A has keys (pk_A, sk_A)
- Public key pk_A is used to encrypt messages to A
- Secret key sk_A is used by A to decrypt
- A publishes pk_A while keeping sk_A secret
- Anybody can encrypt, only A can decrypt

Digital signatures

Going Beyond Key Exchange

	Private-Key	Public-Key
Secrecy	Private-key encryption	Public-key encryption
Integrity	MACs	Digital signatures

Public-Key Encryption

- User A has keys (pk_A, sk_A)
- Public key pk_A is used to encrypt messages to A
- Secret key sk_A is used by A to decrypt
- A publishes pk_A while keeping sk_A secret
- Anybody can encrypt, only A can decrypt

Digital signatures

- A has keys (pk_A, sk_A)

Going Beyond Key Exchange

	Private-Key	Public-Key
Secrecy	Private-key encryption	Public-key encryption
Integrity	MACs	Digital signatures

Public-Key Encryption

- User A has keys (pk_A, sk_A)
- Public key pk_A is used to encrypt messages to A
- Secret key sk_A is used by A to decrypt
- A publishes pk_A while keeping sk_A secret
- Anybody can encrypt, only A can decrypt

Digital signatures

- A has keys (pk_A, sk_A)
- Secret key sk_A is used by A to sign messages

Going Beyond Key Exchange

	Private-Key	Public-Key
Secrecy	Private-key encryption	Public-key encryption
Integrity	MACs	Digital signatures

Public-Key Encryption

- User A has keys (pk_A, sk_A)
- Public key pk_A is used to encrypt messages to A
- Secret key sk_A is used by A to decrypt
- A publishes pk_A while keeping sk_A secret
- Anybody can encrypt, only A can decrypt

Digital signatures

- A has keys (pk_A, sk_A)
- Secret key sk_A is used by A to sign messages
- Public key pk_A is used to verify A's signatures

Going Beyond Key Exchange

	Private-Key	Public-Key
Secrecy	Private-key encryption	Public-key encryption
Integrity	MACs	Digital signatures

Public-Key Encryption

- User A has keys (pk_A, sk_A)
- Public key pk_A is used to encrypt messages to A
- Secret key sk_A is used by A to decrypt
- A publishes pk_A while keeping sk_A secret
- Anybody can encrypt, only A can decrypt

Digital signatures

- A has keys (pk_A, sk_A)
- Secret key sk_A is used by A to sign messages
- Public key pk_A is used to verify A's signatures
- A publishes pk_A while keeping sk_A secret

Going Beyond Key Exchange

	Private-Key	Public-Key
Secrecy	Private-key encryption	Public-key encryption
Integrity	MACs	Digital signatures

Public-Key Encryption

- User A has keys (pk_A, sk_A)
- Public key pk_A is used to encrypt messages to A
- Secret key sk_A is used by A to decrypt
- A publishes pk_A while keeping sk_A secret
- Anybody can encrypt, only A can decrypt

Digital signatures

- A has keys (pk_A, sk_A)
- Secret key sk_A is used by A to sign messages
- Public key pk_A is used to verify A's signatures
- A publishes pk_A while keeping sk_A secret
- Only A can sign, anybody can verify

Public-Key Revolution

The development of key-exchange in 1976 revolutionized what we could do with cryptography:

Public-Key Revolution

The development of key-exchange in 1976 revolutionized what we could do with cryptography:

- Can do key distribution over public channels, no need for couriers

Public-Key Revolution

The development of key-exchange in 1976 revolutionized what we could do with cryptography:

- Can do key distribution over public channels, no need for couriers
- Can update keys whenever necessary

Public-Key Revolution

The development of key-exchange in 1976 revolutionized what we could do with cryptography:

- Can do key distribution over public channels, no need for couriers
- Can update keys whenever necessary
- User only needs to store one secret key

Public-Key Revolution

The development of key-exchange in 1976 revolutionized what we could do with cryptography:

- Can do key distribution over public channels, no need for couriers
- Can update keys whenever necessary
- User only needs to store one secret key
- Public keys can be posted online for open systems

Public-Key Revolution

The development of key-exchange in 1976 revolutionized what we could do with cryptography:

- Can do key distribution over public channels, no need for couriers
- Can update keys whenever necessary
- User only needs to store one secret key
- Public keys can be posted online for open systems

Drawbacks:

- Public-key crypto is 2-3 orders of magnitude slower than secret-key operations

Public-Key Revolution

The development of key-exchange in 1976 revolutionized what we could do with cryptography:

- Can do key distribution over public channels, no need for couriers
- Can update keys whenever necessary
- User only needs to store one secret key
- Public keys can be posted online for open systems

Drawbacks:

- Public-key crypto is 2-3 orders of magnitude slower than secret-key operations

Public-key crypto today

Public-key cryptography enables today's Internet and more:

- When you buy something on Amazon
- When you surf the web
- . . .