London MathSpace - Diffie-Hellman key exchange

Merovius

December 5, 2014

Groups

Definition

A *Group* is a set G, with an operation

$$\cdot : G \times G \to G$$

$$(g, h) \mapsto g \cdot h =: gh$$

such that:

1. Associativity: $\forall g_1, g_2, g_3 \in G : (g_1 g_2) g_3 = g_1 (g_2 g_3)$
2. Existence of a neutral element:
   $\exists e \in G : \forall g \in G : eg = ge = g$
3. Existence of inverses: $\forall g \in G : \exists h \in G : gh = hg = e$

Groups

Definition

A *Group* is a set G, with an operation

$$\cdot \colon G \times G \to G$$

$$(g, h) \mapsto g \cdot h =: gh$$

such that:

1. Associativity: $\forall g_1, g_2, g_3 \in G \colon (g_1 g_2) g_3 = g_1 (g_2 g_3)$
2. Existence of a neutral element:
   $\exists e \in G \colon \forall g \in G \colon eg = ge = g$
3. Existence of inverses: $\forall g \in G \colon \exists h \in G \colon gh = hg = e$

By writing (for $n \in \mathbb{Z}$) $g^n = \underbrace{g \cdot \dots \cdot g}_{n \text{ times}}$ we get the usual identities

$g^{n+m} = g^n g^m$ and $(g^n)^m = (g^m)^n = g^{mn}$.

Examples of groups

- $\mathbb{Z}$ with the operation $+$

Examples of groups

- $\mathbb{Z}$ with the operation $+$
- $\mathbb{Q}$ with the operation $\cdot$

Examples of groups

- $\mathbb{Z}$ with the operation $+$
- $\mathbb{Q}$ with the operation $\cdot$
- Permutations of a rubik's cube with operation of composition

Examples of groups

- $\mathbb{Z}$ with the operation $+$
- $\mathbb{Q}$ with the operation $\cdot$
- Permutations of a rubik's cube with operation of composition
- Symmetries of a regular n-gon with operation of composition

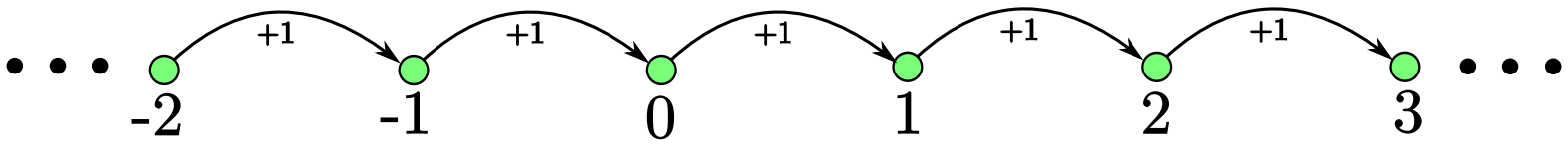
Examples of groups

- $\mathbb{Z}$ with the operation $+$
- $\mathbb{Q}$ with the operation $\cdot$
- Permutations of a rubik's cube with operation of composition
- Symmetries of a regular n-gon with operation of composition
- $\mathbb{Z}/n\mathbb{Z}$, the integers modulo n, with operation of addition

Examples of groups

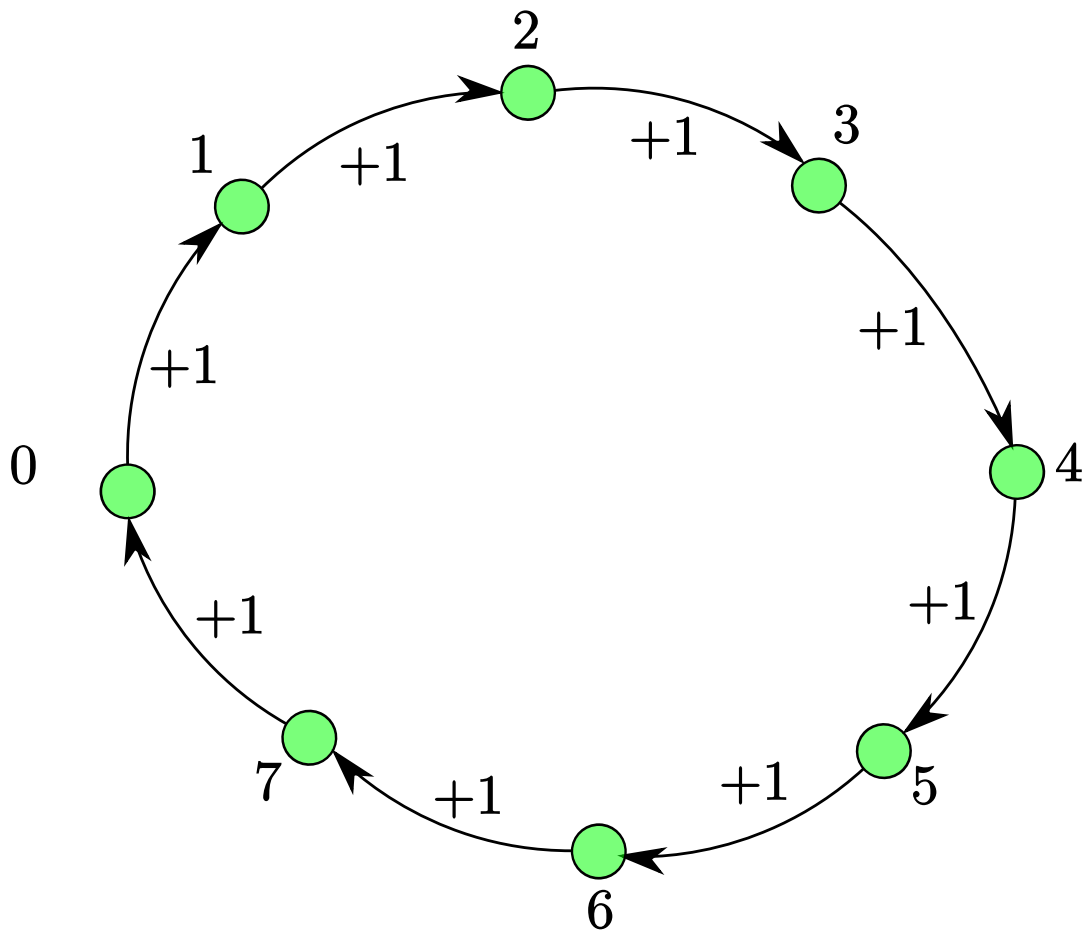
- $\mathbb{Z}$ with the operation $+$
- $\mathbb{Q}$ with the operation $\cdot$
- Permutations of a rubik's cube with operation of composition
- Symmetries of a regular n-gon with operation of composition
- $\mathbb{Z}/n\mathbb{Z}$, the integers modulo n, with operation of addition
- $(\mathbb{Z}/p\mathbb{Z})^{\times} := \mathbb{Z}/p\mathbb{Z} \setminus \{0\}$ with p a prime number, with operation of multiplication

Cayley-Graph



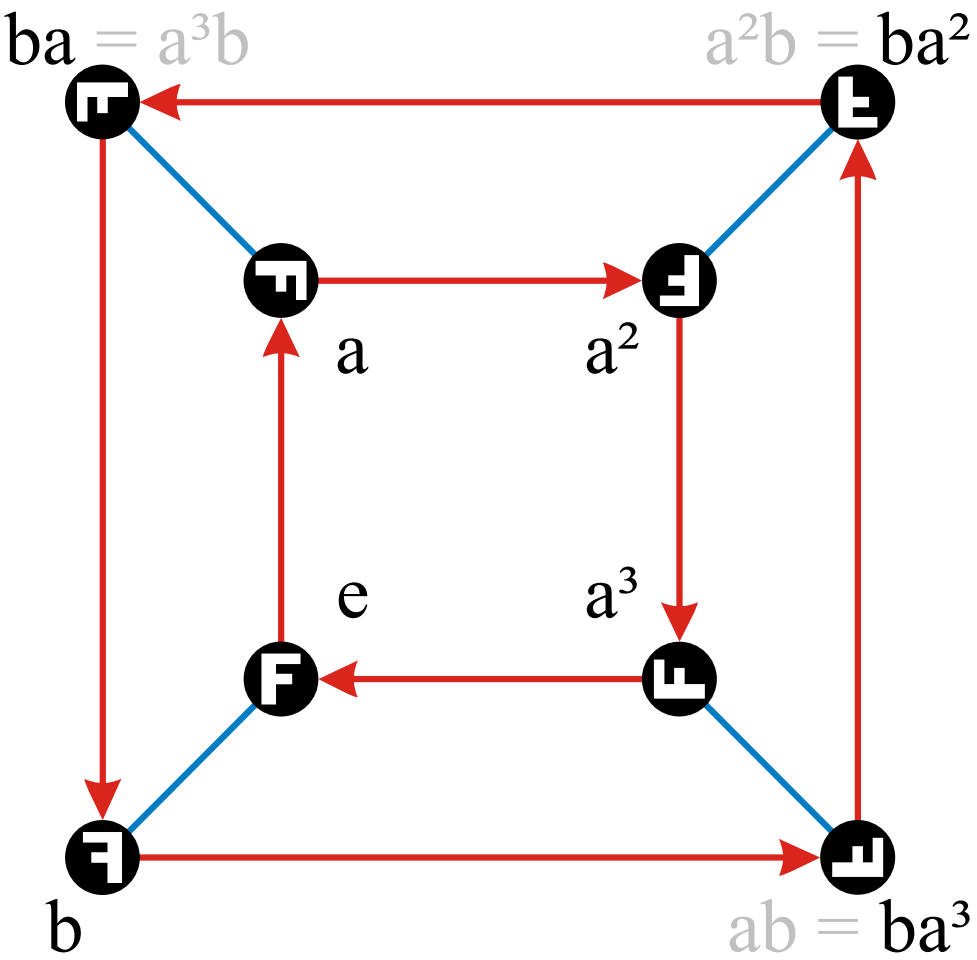
Cayley-Graph of $\mathbb{Z}$.

Cayley-Graph



Cayley-Graph of $\mathbb{Z}/8\mathbb{Z}$.

Cayley-Graph



Cayley-Graph of D_4 (Symmetrien des Quadrats).

Cyclic groups

Definition

Let G be a group, $g \in G$. Then

$$< g >:= \{g^n \mid n \in \mathbb{Z}\}$$

is called the *Subgroup generated by g*. G is called *cyclic*, if $G =< g >$, for some $g \in G$.

Cyclic groups

Definition

Let G be a group, $g \in G$. Then

$$< g >:= \{g^n \mid n \in \mathbb{Z}\}$$

is called the *Subgroup generated by g*. G is called *cyclic*, if $G =< g >$, for some $g \in G$.

Examples:

- $\mathbb{Z} =< 1 >=< -1 >$ is cyclic.

Cyclic groups

Definition

Let G be a group, $g \in G$. Then

$$< g >:= \{g^n \mid n \in \mathbb{Z}\}$$

is called the *Subgroup generated by g*. G is called *cyclic*, if $G =< g >$, for some $g \in G$.

Examples:

- $\mathbb{Z} =< 1 >=< -1 >$ is cyclic.
- $\mathbb{Q}$ is not cyclic.

Cyclic groups

Definition

Let G be a group, $g \in G$. Then

$$< g >:= \{g^n \mid n \in \mathbb{Z}\}$$

is called the *Subgroup generated by g*. G is called *cyclic*, if $G =< g >$, for some $g \in G$.

Examples:

- $\mathbb{Z} =< 1 >=< -1 >$ is cyclic.
- $\mathbb{Q}$ is not cyclic.
- The rubik's group is not cyclic either.

Cyclic groups

Definition

Let G be a group, $g \in G$. Then

$$< g >:= \{g^n \mid n \in \mathbb{Z}\}$$

is called the *Subgroup generated by g*. G is called *cyclic*, if $G =< g >$, for some $g \in G$.

Examples:

- $\mathbb{Z} =< 1 >=< -1 >$ is cyclic.
- $\mathbb{Q}$ is not cyclic.
- The rubik's group is not cyclic either.
- $\mathbb{Z}/n\mathbb{Z}$ is cyclic with generator 1.

Cyclic groups

Definition

Let G be a group, $g \in G$. Then

$$< g >:= \{g^n \mid n \in \mathbb{Z}\}$$

is called the *Subgroup generated by g*. G is called *cyclic*, if $G =< g >$, for some $g \in G$.

Examples:

- $\mathbb{Z} =< 1 >=< -1 >$ is cyclic.
- $\mathbb{Q}$ is not cyclic.
- The rubik's group is not cyclic either.
- $\mathbb{Z}/n\mathbb{Z}$ is cyclic with generator 1.
- $(\mathbb{Z}/p\mathbb{Z})^\times$ is cyclic.

Turing machines

- A *Turing* machine is an abstraction of a computer that let's us *mathematically precise* define what it's runtime, memory usage... etc. are.

Turing machines

- A *Turing* machine is an abstraction of a computer that let's us *mathematically precise* define what it's runtime, memory usage... etc. are.
- (Sorry, too lazy for a complete rundown on theoretical computer science)

Turing machines

- A *Turing* machine is an abstraction of a computer that let's us *mathematically precise* define what it's runtime, memory usage... etc. are.
- (Sorry, too lazy for a complete rundown on theoretical computer science)
- A Turing machince takes an input (for example in $\{0,1\}^*$) and produces an output.

Turing machines

- A *Turing* machine is an abstraction of a computer that let's us *mathematically precise* define what it's runtime, memory usage... etc. are.
- (Sorry, too lazy for a complete rundown on theoretical computer science)
- A Turing machince takes an input (for example in $\{0,1\}^*$) and produces an output.
- It runs in *polynomial time*, if it's runtime can be represented as a polynomial of the size (= number of bits) of the input.

Turing machines

- A *Turing* machine is an abstraction of a computer that let's us *mathematically precise* define what it's runtime, memory usage... etc. are.
- (Sorry, too lazy for a complete rundown on theoretical computer science)
- A Turing machince takes an input (for example in $\{0,1\}^*$) and produces an output.
- It runs in *polynomial time*, if it's runtime can be represented as a polynomial of the size (= number of bits) of the input.
- A *Probabalistic* Turing machine takes an additional input in $\{0,1\}^n$, for some n.

Turing machines

- A *Turing* machine is an abstraction of a computer that let's us *mathematically precise* define what it's runtime, memory usage... etc. are.
- (Sorry, too lazy for a complete rundown on theoretical computer science)
- A Turing machince takes an input (for example in $\{0,1\}^*$) and produces an output.
- It runs in *polynomial time*, if it's runtime can be represented as a polynomial of the size (= number of bits) of the input.
- A *Probabalistic* Turing machine takes an additional input in $\{0,1\}^n$, for some n.
- A *Probabalistic Polynomial-Time* (PPT) Turing machine is a probabalistic turing machine with polynomial running-time.

Negligible functions

A function $\epsilon\colon \mathbb{N} \to \mathbb{N}$ is called *negligible*, if for every polynomial p, we have

$$\exists N \in \mathbb{N}\colon \forall n > N\colon \epsilon(n) < \frac{1}{p(n)}$$

Negligible functions

A function $\epsilon\colon \mathbb{N} \to \mathbb{N}$ is called *negligible*, if for every polynomial p, we have

$$\exists N \in \mathbb{N}\colon \forall n > N\colon \epsilon(n) < \frac{1}{p(n)}$$

Lemma

If p is a polynomial, and ϵ is negligible, then $p(\epsilon(n))$ is negligible.

Negligible functions

A function $\epsilon\colon \mathbb{N} \to \mathbb{N}$ is called *negligible*, if for every polynomial p, we have

$$\exists N \in \mathbb{N}\colon \forall n > N\colon \epsilon(n) < \frac{1}{p(n)}$$

Lemma

If p is a polynomial, and ϵ is negligible, then $p(\epsilon(n))$ is negligible.

Proof.

Assume $p(\epsilon(n))$ is not neglible. Let w.l.o.g. $p(n) = n^m$. Let q be a polynomial, such that

$$\forall N \in \mathbb{N}\colon \exists n > N\colon p(\epsilon(n)) = \epsilon(n)^m \geq \frac{1}{q(n)}$$

Negligible functions

A function $\epsilon\colon \mathbb{N} \to \mathbb{N}$ is called *negligible*, if for every polynomial p, we have

$$\exists N \in \mathbb{N}\colon \forall n > N\colon \epsilon(n) < \frac{1}{p(n)}$$

Lemma

If p is a polynomial, and ϵ is negligible, then $p(\epsilon(n))$ is negligible.

Proof.

Assume $p(\epsilon(n))$ is not neglible. Let w.l.o.g. $p(n) = n^m$. Let q be a polynomial, such that

$$\forall N \in \mathbb{N}\colon \exists n > N\colon p(\epsilon(n)) = \epsilon(n)^m \geq \frac{1}{q(n)} \geq \frac{1}{q(n)^m}$$

Negligible functions

A function $\epsilon\colon \mathbb{N} \to \mathbb{N}$ is called *negligible*, if for every polynomial p, we have

$$\exists N \in \mathbb{N}\colon \forall n > N\colon \epsilon(n) < \frac{1}{p(n)}$$

Lemma

If p is a polynomial, and ϵ is negligible, then $p(\epsilon(n))$ is negligible.

Proof.

Assume $p(\epsilon(n))$ is not neglible. Let w.l.o.g. $p(n) = n^m$. Let q be a polynomial, such that

$$\forall N \in \mathbb{N}\colon \exists n > N\colon p(\epsilon(n)) = \epsilon(n)^m \geq \frac{1}{q(n)} \geq \frac{1}{q(n)^m}$$

$$\Rightarrow \epsilon(n) \geq \frac{1}{q(n)}$$

Negligible functions

A function $\epsilon\colon \mathbb{N} \to \mathbb{N}$ is called *negligible*, if for every polynomial p, we have

$$\exists N \in \mathbb{N}\colon \forall n > N\colon \epsilon(n) < \frac{1}{p(n)}$$

Lemma

If p is a polynomial, and ϵ is negligible, then $p(\epsilon(n))$ is negligible.

Proof.

Assume $p(\epsilon(n))$ is not neglible. Let w.l.o.g. $p(n) = n^m$. Let q be a polynomial, such that

$$\forall N \in \mathbb{N}\colon \exists n > N\colon p(\epsilon(n)) = \epsilon(n)^m \geq \frac{1}{q(n)} \geq \frac{1}{q(n)^m}$$

$$\Rightarrow \epsilon(n) \geq \frac{1}{q(n)} \qquad \lightning$$

□

Diffie-Hellman key exchange

Problem: Alice and Bob have a verified, but non-confidential channel. They want to negiotiate (using public data) some common secret K.

Diffie-Hellman key exchange

Problem: Alice and Bob have a verified, but non-confidential channel. They want to negiotiate (using public data) some common secret K.
Diffie-Hellman key exchange:

1. Alice and Bob negotiate a finite cyclic group G and a generator g, so $G =< g >$ (as part of the protocol specification).

Diffie-Hellman key exchange

Problem: Alice and Bob have a verified, but non-confidential channel. They want to negiotiate (using public data) some common secret K.
Diffie-Hellman key exchange:

1. Alice and Bob negotiate a finite cyclic group G and a generator g, so $G =< g >$ (as part of the protocol specification).
2. Alice chooses (uniformly at random) an integer a and Bob chooses an integer b.

Diffie-Hellman key exchange

Problem: Alice and Bob have a verified, but non-confidential channel. They want to negiotiate (using public data) some common secret K.
Diffie-Hellman key exchange:

1. Alice and Bob negotiate a finite cyclic group G and a generator g, so $G =< g >$ (as part of the protocol specification).
2. Alice chooses (uniformly at random) an integer a and Bob chooses an integer b.
3. Alice sends $x = g^a$ over the channel, Bob sends $y = g^b$.

Diffie-Hellman key exchange

Problem: Alice and Bob have a verified, but non-confidential channel. They want to negiotiate (using public data) some common secret K.
Diffie-Hellman key exchange:

1. Alice and Bob negotiate a finite cyclic group G and a generator g, so $G =< g >$ (as part of the protocol specification).
2. Alice chooses (uniformly at random) an integer a and Bob chooses an integer b.
3. Alice sends $x = g^a$ over the channel, Bob sends $y = g^b$.
4. Alice computes $k_A := y^a$ and Bob computes $k_B := x^b$.

Diffie-Hellman key exchange

Problem: Alice and Bob have a verified, but non-confidential channel. They want to negiotiate (using public data) some common secret K.
Diffie-Hellman key exchange:

1. Alice and Bob negotiate a finite cyclic group G and a generator g, so $G =< g >$ (as part of the protocol specification).
2. Alice chooses (uniformly at random) an integer a and Bob chooses an integer b.
3. Alice sends $x = g^a$ over the channel, Bob sends $y = g^b$.
4. Alice computes $k_A := y^a$ and Bob computes $k_B := x^b$.

In the end, we have

$$k_A = y^a = (g^b)^a = g^{ba} = g^{ab} = (g^a)^b = x^b = k_B$$

Example

1. Let $G = \mathbb{Z}/256\mathbb{Z}$, $g = 1$.

Example

1. Let $G = \mathbb{Z}/256\mathbb{Z}$, $g = 1$.
2. Alice chooses (say) $a = 92$. Bob chooses $b = 214$.

Example

1. Let $G = \mathbb{Z}/256\mathbb{Z}$, $g = 1$.
2. Alice chooses (say) $a = 92$. Bob chooses $b = 214$.
3. Alice computes $x = g^{92} = 92 \cdot 1 = 92$, Bob computes $y = g^{214} = 214 \cdot 1 = 214$. They exchange them.

Example

1. Let $G = \mathbb{Z}/256\mathbb{Z}$, $g = 1$.
2. Alice chooses (say) $a = 92$. Bob chooses $b = 214$.
3. Alice computes $x = g^{92} = 92 \cdot 1 = 92$, Bob computes $y = g^{214} = 214 \cdot 1 = 214$. They exchange them.
4. Alice computes $k_A = y^{92} = 92 \cdot 214 = 19688 \equiv 232$. Bob computes $k_B = x^{214} = 214 \cdot 92 = 19688 \equiv 232$.

Example

1. Let $G = \mathbb{Z}/256\mathbb{Z}$, $g = 1$.
2. Alice chooses (say) $a = 92$. Bob chooses $b = 214$.
3. Alice computes $x = g^{92} = 92 \cdot 1 = 92$, Bob computes $y = g^{214} = 214 \cdot 1 = 214$. They exchange them.
4. Alice computes $k_A = y^{92} = 92 \cdot 214 = 19688 \equiv 232$. Bob computes $k_B = x^{214} = 214 \cdot 92 = 19688 \equiv 232$.

This is obviously stupid

Hardness assumptions

Let G be a finite cyclic group with generator g

Definition

The *computational discrete log problem* for the group G is the following: Find a PPT Turing machine A, that, given an element $h \in G$, outputs an $n \in \mathbb{Z}$, such that $\Pr[g^n = h]$ is non-negligible.

Hardness assumptions

Let G be a finite cyclic group with generator g

Definition

The *computational discrete log problem* for the group G is the following: Find a PPT Turing machine A, that, given an element $h \in G$, outputs an $n \in \mathbb{Z}$, such that $\Pr[g^n = h]$ is non-negligible.

The discrete log problem on $\mathbb{Z}/n\mathbb{Z}$ is very easy: Just output h.

Hardness assumptions

Let G be a finite cyclic group with generator g

Definition

The *computational discrete log problem* for the group G is the following: Find a PPT Turing machine A, that, given an element $h \in G$, outputs an $n \in \mathbb{Z}$, such that $\Pr[g^n = h]$ is non-negligible.

The discrete log problem on $\mathbb{Z}/n\mathbb{Z}$ is very easy: Just output h.

Definition

The *computational Diffie-Hellman problem* for the group G is the following: Find a PPT Turing machine B, that, given two elements $h = g^a$ and $h' = g^b$ (without either a or b) outputs an element $n \in \mathbb{Z}$, such that $\Pr[g^n = g^{ab}]$ is non-negligible.

Hardness assumptions

Let G be a finite cyclic group with generator g

Definition

The *computational discrete log problem* for the group G is the following: Find a PPT Turing machine A, that, given an element $h \in G$, outputs an $n \in \mathbb{Z}$, such that $\Pr[g^n = h]$ is non-negligible.

The discrete log problem on $\mathbb{Z}/n\mathbb{Z}$ is very easy: Just output h.

Definition

The *computational Diffie-Hellman problem* for the group G is the following: Find a PPT Turing machine B, that, given two elements $h = g^a$ and $h' = g^b$ (without either a or b) outputs an element $n \in \mathbb{Z}$, such that $\Pr[g^n = g^{ab}]$ is non-negligible.

If we can solve discrete log for G, we can also solve the DH-Problem for G.

Security of DH

It is not yet known, whether there is a group G, such that either the DH-problem or the discrete log problem is unsolvable!

Security of DH

It is not yet known, whether there is a group G, such that either the DH-problem or the discrete log problem is unsolvable!
Iff the DH-Problem is unsolvable over G, the Diffie-Hellman Key exchange is secure:

- As the channel is verified, Eve can not temper with the transmission of x, y and they arrive unmodified.

Security of DH

It is not yet known, whether there is a group G, such that either the DH-problem or the discrete log problem is unsolvable!
Iff the DH-Problem is unsolvable over G, the Diffie-Hellman Key exchange is secure:

- As the channel is verified, Eve can not temper with the transmission of x, y and they arrive unmodified.
- Eve *only* gets to know g^a, g^b and wants to know g^{ab}. Thus she has to solve the DH-Problem for G.

Security of DH

It is not yet known, whether there is a group G, such that either the DH-problem or the discrete log problem is unsolvable!
Iff the DH-Problem is unsolvable over G, the Diffie-Hellman Key exchange is secure:

- As the channel is verified, Eve can not temper with the transmission of x, y and they arrive unmodified.
- Eve *only* gets to know g^a, g^b and wants to know g^{ab}. Thus she has to solve the DH-Problem for G.

Candidate groups for having hard DH-Problems: $(\mathbb{Z}/p\mathbb{Z})^\times$ for large p, certain elliptic curve groups (ECC-DH).

(Better) Example

1. Let $G = (\mathbb{Z}/257\mathbb{Z})^{\times}$, $g = 2$.

(Better) Example

1. Let $G = (\mathbb{Z}/257\mathbb{Z})^{\times}$, $g = 2$.
2. Alice chooses (say) $a = 92$. Bob chooses $b = 214$.

(Better) Example

1. Let $G = (\mathbb{Z}/257\mathbb{Z})^{\times}$, $g = 2$.
2. Alice chooses (say) $a = 92$. Bob chooses $b = 214$.
3. Alice computes $x = g^a = 2^{92} \equiv 241$. Bob computes $y = g^b = 2^{214} \equiv 64$. They exchange x, y.

(Better) Example

1. Let $G = (\mathbb{Z}/257\mathbb{Z})^{\times}$, $g = 2$.
2. Alice chooses (say) $a = 92$. Bob chooses $b = 214$.
3. Alice computes $x = g^a = 2^{92} \equiv 241$. Bob computes $y = g^b = 2^{214} \equiv 64$. They exchange x, y.
4. Alice computes $k_A = y^a = 64^{92} \equiv 256$. Bob computes $k_B = x^b = 241^{214} \equiv 256$.

(Perfect) Forward Secrecy

- Let's say Alice and Bob use public-key crypto to create a verified and confidential communication channel.

(Perfect) Forward Secrecy

- Let's say Alice and Bob use public-key crypto to create a verified and confidential communication channel.
- Eve intercepts the (encrypted) communication and records it.

(Perfect) Forward Secrecy

- Let's say Alice and Bob use public-key crypto to create a verified and confidential communication channel.
- Eve intercepts the (encrypted) communication and records it.
- At a future date, Eve gains access to Alice's secret key. She can then decrypt the recorded communication.

(Perfect) Forward Secrecy

- Let's say Alice and Bob use public-key crypto to create a verified and confidential communication channel.
- Eve intercepts the (encrypted) communication and records it.
- At a future date, Eve gains access to Alice's secret key. She can then decrypt the recorded communication.

Alternative Scenario:

- Alice and Bob use public-key crypto to create a *verified* communication channel.

(Perfect) Forward Secrecy

- Let's say Alice and Bob use public-key crypto to create a verified and confidential communication channel.
- Eve intercepts the (encrypted) communication and records it.
- At a future date, Eve gains access to Alice's secret key. She can then decrypt the recorded communication.

Alternative Scenario:

- Alice and Bob use public-key crypto to create a *verified* communication channel.
- They employ DH Key exchange to generate a common secret, which they use for a *confidential* channel.

(Perfect) Forward Secrecy

- Let's say Alice and Bob use public-key crypto to create a verified and confidential communication channel.
- Eve intercepts the (encrypted) communication and records it.
- At a future date, Eve gains access to Alice's secret key. She can then decrypt the recorded communication.

Alternative Scenario:

- Alice and Bob use public-key crypto to create a *verified* communication channel.
- They employ DH Key exchange to generate a common secret, which they use for a *confidential* channel.
- Eve intercepts the (encrypted) communication and records it.

(Perfect) Forward Secrecy

- Let's say Alice and Bob use public-key crypto to create a verified and confidential communication channel.
- Eve intercepts the (encrypted) communication and records it.
- At a future date, Eve gains access to Alice's secret key. She can then decrypt the recorded communication.

Alternative Scenario:

- Alice and Bob use public-key crypto to create a *verified* communication channel.
- They employ DH Key exchange to generate a common secret, which they use for a *confidential* channel.
- Eve intercepts the (encrypted) communication and records it.
- At a future date, Eve gains acces to Alice's secret key. Since the communication is in the past, Eve can not compromise the verification. Thus confidentiality remains.

(Perfect) Forward Secrecy

- Let's say Alice and Bob use public-key crypto to create a verified and confidential communication channel.
- Eve intercepts the (encrypted) communication and records it.
- At a future date, Eve gains access to Alice's secret key. She can then decrypt the recorded communication.

Alternative Scenario:

- Alice and Bob use public-key crypto to create a *verified* communication channel.
- They employ DH Key exchange to generate a common secret, which they use for a *confidential* channel.
- Eve intercepts the (encrypted) communication and records it.
- At a future date, Eve gains acces to Alice's secret key. Since the communication is in the past, Eve can not compromise the verification. Thus confidentiality remains.
- Every *future* communication between Alice and Bob can be decyphered by Eve.

Questions and Discussion