Recent advances in group-based cryptography

Vladimir Shpilrain
The City College of New York
shpil@groups.sci.ccny.cuny.edu

May 24, 2013

Why?

- Efficiency (smaller key size, less computation)
- Security (?)
- Trying to do something useful

Why?

- Efficiency (smaller key size, less computation)
- Security (?)
- Trying to do something useful

Why?

- Efficiency (smaller key size, less computation)
- Security (?)
- Trying to do something useful

One-way functions

$\longrightarrow$ easy $\longleftarrow$ hard

$$f(x) = x^n$$

Trapdoor !

One-way functions

$\longrightarrow$ easy $\qquad$ $\longleftarrow$ hard

$$f(x) = x^n$$

Trapdoor !

One-way functions

$\longrightarrow$ easy $\longleftarrow$ hard

$$f(x) = x^n$$

Trapdoor !

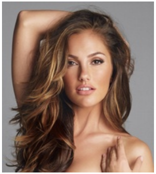
Public Key Cryptography

- Encryption
- Key agreement (a.k.a. key exchange, a.k.a. key establishment)
- Authentication

Public Key Cryptography

- Encryption
- Key agreement (a.k.a. key exchange, a.k.a. key establishment)
- Authentication

The Diffie-Hellman key establishment (1976)



1. Alice and Bob agree on a (finite) cyclic group G and a generating element g in G. We will write the group G multiplicatively.
2. Alice picks a random natural number a and sends g^a to Bob.
3. Bob picks a random natural number b and sends g^b to Alice.
4. Alice computes $K_A = (g^b)^a = g^{ba}$.
5. Bob computes $K_B = (g^a)^b = g^{ab}$.

Since $ab = ba$ (because $\mathbb{Z}$ is commutative), both Alice and Bob are now in possession of the same group element $K = K_A = K_B$ which can serve as the shared secret key.

Efficiency for legitimate parties

Exponentiation by “square-and-multiply”:

$$g^{22} = (((g^2)^2)^2)^2 \cdot (g^2)^2 \cdot g^2$$

Complexity of computing g^n is therefore $O(\log n)$, times complexity of reducing *mod p* (more generally, reducing to a “normal form” in the platform group G). In the original Diffie-Hellman protocol, G was $\mathbb{Z}_p^*$.

Efficiency for legitimate parties

Exponentiation by "square-and-multiply":

$$g^{22} = (((g^2)^2)^2)^2 \cdot (g^2)^2 \cdot g^2$$

Complexity of computing g^n is therefore $O(\log n)$, times complexity of reducing *mod p* (more generally, reducing to a "normal form" in the platform group G). In the original Diffie-Hellman protocol, G was $\mathbb{Z}_p^*$.

Efficiency for legitimate parties

Exponentiation by "square-and-multiply":

$$g^{22} = (((g^2)^2)^2)^2 \cdot (g^2)^2 \cdot g^2$$

Complexity of computing g^n is therefore $O(\log n)$, times complexity of reducing *mod p* (more generally, reducing to a "normal form" in the platform group G). In the original Diffie-Hellman protocol, G was $\mathbb{Z}_p^*$.

Variations on Diffie-Hellman: why not just multiply them?

1. Alice and Bob agree on a (finite) cyclic group G and a generating element g in G. We will write the group G multiplicatively.
2. Alice picks a random natural number a and sends g^a to Bob.
3. Bob picks a random natural number b and sends g^b to Alice.
4. Alice computes $K_A = (g^b) \cdot (g^a) = g^{b+a}$.
5. Bob computes $K_B = (g^a) \cdot (g^b) = g^{a+b}$.

Obviously, $K_A = K_B = K$, which can serve as the shared secret key.

Drawback: anybody can obtain K the same way!

Variations on Diffie-Hellman: why not just multiply them?

1. Alice and Bob agree on a (finite) cyclic group G and a generating element g in G. We will write the group G multiplicatively.
2. Alice picks a random natural number a and sends g^a to Bob.
3. Bob picks a random natural number b and sends g^b to Alice.
4. Alice computes $K_A = (g^b) \cdot (g^a) = g^{b+a}$.
5. Bob computes $K_B = (g^a) \cdot (g^b) = g^{a+b}$.

Obviously, $K_A = K_B = K$, which can serve as the shared secret key.

Drawback: anybody can obtain K the same way!

Security assumptions

To recover g^{ab} from (g, g^a, g^b) is hard.

To recover a from (g, g^a) (discrete log problem) is hard.

Security assumptions

To recover g^{ab} from (g, g^a, g^b) is hard.

To recover a from (g, g^a) (discrete log problem) is hard.

The Ko-Lee et al. protocol

1. Alice and Bob agree on a group G and an element w in G. Thus, G and w are public.
2. Alice picks a private $a \in G$ and sends $w^a = a^{-1}wa$ to Bob.
3. Bob picks a private $b \in G$ and sends $w^b = b^{-1}wb$ to Alice.
4. Alice computes $K_A = (w^b)^a = w^{ba}$, and Bob computes $K_B = (w^a)^b = w^{ab}$.

If $ab = ba$, then Alice and Bob get a common private key $K_B = w^{ab} = w^{ba} = K_A$. Typically, there are two public subgroups A and B of the group G, given by their (finite) generating sets, such that $ab = ba$ for any $a \in A$, $b \in B$.

The platform group G

(P0) The group G has to be well known. More specifically, the *conjugacy search problem* (i.e., recovering a from $(w,\ a^{-1}wa)$) in the group G either has to be well studied or can be reduced to a well-known problem.

(P1) The word problem in G should have a fast (e.g. quadratic-time) solution by a deterministic algorithm. Better yet, there should be an efficiently computable "normal form" for elements of G.

(P2) The conjugacy search problem should *not* have an efficient solution by a deterministic algorithm.

(P3) There should be a way to disguise elements of G so that it would be impossible to recover x from $x^{-1}wx$ just by inspection. Example: "normal form".

(P4) G should be "large", i.e. have a "fast growth". This is necessary to have a sufficiently large key space.

The platform group G

(P0) The group G has to be well known. More specifically, the *conjugacy search problem* (i.e., recovering a from $(w,\ a^{-1}wa)$) in the group G either has to be well studied or can be reduced to a well-known problem.

(P1) The word problem in G should have a fast (e.g. quadratic-time) solution by a deterministic algorithm. Better yet, there should be an efficiently computable "normal form" for elements of G.

(P2) The conjugacy search problem should *not* have an efficient solution by a deterministic algorithm.

(P3) There should be a way to disguise elements of G so that it would be impossible to recover x from $x^{-1}wx$ just by inspection. Example: "normal form".

(P4) G should be "large", i.e. have a "fast growth". This is necessary to have a sufficiently large key space.

The platform group G

- (P0) The group G has to be well known. More specifically, the *conjugacy search problem* (i.e., recovering a from $(w,\ a^{-1}wa)$) in the group G either has to be well studied or can be reduced to a well-known problem.
- (P1) The word problem in G should have a fast (e.g. quadratic-time) solution by a deterministic algorithm. Better yet, there should be an efficiently computable "normal form" for elements of G.
- (P2) The conjugacy search problem should *not* have an efficient solution by a deterministic algorithm.
- (P3) There should be a way to disguise elements of G so that it would be impossible to recover x from $x^{-1}wx$ just by inspection. Example: "normal form".
- (P4) G should be "large", i.e. have a "fast growth". This is necessary to have a sufficiently large key space.

The platform group G

(P0) The group G has to be well known. More specifically, the *conjugacy search problem* (i.e., recovering a from $(w,\ a^{-1}wa)$) in the group G either has to be well studied or can be reduced to a well-known problem.

(P1) The word problem in G should have a fast (e.g. quadratic-time) solution by a deterministic algorithm. Better yet, there should be an efficiently computable "normal form" for elements of G.

(P2) The conjugacy search problem should *not* have an efficient solution by a deterministic algorithm.

(P3) There should be a way to disguise elements of G so that it would be impossible to recover x from $x^{-1}wx$ just by inspection. Example: "normal form".

(P4) G should be "large", i.e. have a "fast growth". This is necessary to have a sufficiently large key space.

The platform group G

(P0) The group G has to be well known. More specifically, the *conjugacy search problem* (i.e., recovering a from $(w,\ a^{-1}wa)$) in the group G either has to be well studied or can be reduced to a well-known problem.

(P1) The word problem in G should have a fast (e.g. quadratic-time) solution by a deterministic algorithm. Better yet, there should be an efficiently computable "normal form" for elements of G.

(P2) The conjugacy search problem should *not* have an efficient solution by a deterministic algorithm.

(P3) There should be a way to disguise elements of G so that it would be impossible to recover x from $x^{-1}wx$ just by inspection. Example: "normal form".

(P4) G should be "large", i.e. have a "fast growth". This is necessary to have a sufficiently large key space.

Platform groups

- Braid groups
- Thompson's group
- Small cancellation groups
- Polycyclic groups
- Groups of matrices over various rings

Semidirect product

Let G, H be two groups, let $Aut(G)$ be the group of automorphisms of G, and let $\rho : H \to Aut(G)$ be a homomorphism. Then the semidirect product of G and H is the set

$$\Gamma = G \rtimes_\rho H = \{(g, h) : g \in G,\ h \in H\}$$

with the group operation given by

$$(g, h)(g', h') = (g^{\rho(h)} \cdot g',\ h \cdot h').$$

Here $g^{\rho(h)}$ denotes the image of g under the automorphism $\rho(h)$.

Holomorph

If $H = Aut(G)$, then the corresponding semidirect product is called the *holomorph* of the group G. Thus, the holomorph of G, usually denoted by $Hol(G)$, is the set of all pairs $(g,\ \phi)$, where $g \in G,\ \phi \in Aut(G)$, with the group operation given by

$$(g,\ \phi) \cdot (g',\ \phi') = (\phi'(g) \cdot g',\ \phi \cdot \phi').$$

It is often more practical to use a subgroup of $Aut(G)$ in this construction.

Also, if we want the result to be just a semigroup, not necessarily a group, we can consider the semigroup $End(G)$ instead of the group $Aut(G)$ in this construction.

Holomorph

If $H = Aut(G)$, then the corresponding semidirect product is called the *holomorph* of the group G. Thus, the holomorph of G, usually denoted by $Hol(G)$, is the set of all pairs $(g, \ \phi)$, where $g \in G, \ \phi \in Aut(G)$, with the group operation given by

$$(g, \ \phi) \cdot (g', \ \phi') = (\phi'(g) \cdot g', \ \phi \cdot \phi').$$

It is often more practical to use a subgroup of $Aut(G)$ in this construction.

Also, if we want the result to be just a semigroup, not necessarily a group, we can consider the semigroup $End(G)$ instead of the group $Aut(G)$ in this construction.

Holomorph

If $H = Aut(G)$, then the corresponding semidirect product is called the *holomorph* of the group G. Thus, the holomorph of G, usually denoted by $Hol(G)$, is the set of all pairs $(g, \ \phi)$, where $g \in G, \ \phi \in Aut(G)$, with the group operation given by

$$(g, \ \phi) \cdot (g', \ \phi') = (\phi'(g) \cdot g', \ \phi \cdot \phi').$$

It is often more practical to use a subgroup of $Aut(G)$ in this construction.

Also, if we want the result to be just a semigroup, not necessarily a group, we can consider the semigroup $End(G)$ instead of the group $Aut(G)$ in this construction.

Using semidirect product (Habeeb-Kahrobaei-Koupparis-Shpilrain)

Let G be a group (or a semigroup). An element $g \in G$ is chosen and made public as well as an arbitrary automorphism (or an endomorphism) ϕ of G. Bob chooses a private $n \in \mathbb{N}$, while Alice chooses a private $m \in \mathbb{N}$. Both Alice and Bob are going to work with elements of the form (g, ϕ^k), where $g \in G,\ k \in \mathbb{N}$.

1. Alice computes $(g, \phi)^m = (\phi^{m-1}(g) \cdots \phi^2(g) \cdot \phi(g) \cdot g,\ \phi^m)$ and sends **only the first component** of this pair to Bob. Thus, she sends to Bob **only** the element $a = \phi^{m-1}(g) \cdots \phi^2(g) \cdot \phi(g) \cdot g$ of the group G.

2. Bob computes $(g, \phi)^n = (\phi^{n-1}(g) \cdots \phi^2(g) \cdot \phi(g) \cdot g,\ \phi^n)$ and sends **only the first component** of this pair to Alice: $b = \phi^{n-1}(g) \cdots \phi^2(g) \cdot \phi(g) \cdot g$.

3. Alice computes $(b, x) \cdot (a,\ \phi^m) = (\phi^m(b) \cdot a,\ x \cdot \phi^m)$. Her key is now $K_A = \phi^m(b) \cdot a$. Note that she does not actually "compute" $x \cdot \phi^m$ because she does not know the automorphism x; recall that it was not transmitted to her. But she does not need it to compute K_A.

Using semidirect product (Habeeb-Kahrobaei-Koupparis-Shpilrain)

Let G be a group (or a semigroup). An element $g \in G$ is chosen and made public as well as an arbitrary automorphism (or an endomorphism) ϕ of G. Bob chooses a private $n \in \mathbb{N}$, while Alice chooses a private $m \in \mathbb{N}$. Both Alice and Bob are going to work with elements of the form (g, ϕ^k), where $g \in G,\ k \in \mathbb{N}$.

1. Alice computes $(g, \phi)^m = (\phi^{m-1}(g) \cdots \phi^2(g) \cdot \phi(g) \cdot g,\ \phi^m)$ and sends **only the first component** of this pair to Bob. Thus, she sends to Bob **only** the element $a = \phi^{m-1}(g) \cdots \phi^2(g) \cdot \phi(g) \cdot g$ of the group G.
2. Bob computes $(g, \phi)^n = (\phi^{n-1}(g) \cdots \phi^2(g) \cdot \phi(g) \cdot g,\ \phi^n)$ and sends **only the first component** of this pair to Alice: $b = \phi^{n-1}(g) \cdots \phi^2(g) \cdot \phi(g) \cdot g$.
3. Alice computes $(b, x) \cdot (a,\ \phi^m) = (\phi^m(b) \cdot a,\ x \cdot \phi^m)$. Her key is now $K_A = \phi^m(b) \cdot a$. Note that she does not actually "compute" $x \cdot \phi^m$ because she does not know the automorphism x; recall that it was not transmitted to her. But she does not need it to compute K_A.

Using semidirect product (Habeeb-Kahrobaei-Koupparis-Shpilrain)

Let G be a group (or a semigroup). An element $g \in G$ is chosen and made public as well as an arbitrary automorphism (or an endomorphism) ϕ of G. Bob chooses a private $n \in \mathbb{N}$, while Alice chooses a private $m \in \mathbb{N}$. Both Alice and Bob are going to work with elements of the form (g, ϕ^k), where $g \in G,\ k \in \mathbb{N}$.

1. Alice computes $(g, \phi)^m = (\phi^{m-1}(g) \cdots \phi^2(g) \cdot \phi(g) \cdot g,\ \phi^m)$ and sends **only the first component** of this pair to Bob. Thus, she sends to Bob **only** the element $a = \phi^{m-1}(g) \cdots \phi^2(g) \cdot \phi(g) \cdot g$ of the group G.
2. Bob computes $(g, \phi)^n = (\phi^{n-1}(g) \cdots \phi^2(g) \cdot \phi(g) \cdot g,\ \phi^n)$ and sends **only the first component** of this pair to Alice: $b = \phi^{n-1}(g) \cdots \phi^2(g) \cdot \phi(g) \cdot g$.
3. Alice computes $(b, x) \cdot (a,\ \phi^m) = (\phi^m(b) \cdot a,\ x \cdot \phi^m)$. Her key is now $K_A = \phi^m(b) \cdot a$. Note that she does not actually "compute" $x \cdot \phi^m$ because she does not know the automorphism x; recall that it was not transmitted to her. But she does not need it to compute K_A.

Using semidirect product (cont.)

4. Bob computes $(a, y) \cdot (b,\ \phi^n) = (\phi^n(a) \cdot b,\ y \cdot \phi^n)$. His key is now $K_B = \phi^n(a) \cdot b$. Again, Bob does not actually "compute" $y \cdot \phi^n$ because he does not know the automorphism y.

5. Since $(b, x) \cdot (a,\ \phi^m) = (a,\ y) \cdot (b,\ \phi^n) = (g,\ \phi)^{m+n}$, we should have $K_A = K_B = K$, the shared secret key.

Special case: Diffie-Hellman

$G = \mathbb{Z}_p^*$
$\phi(g) = g^k$ for all $g \in G$ and a fixed k, $1 < k < p-1$.

Then $(g, \phi)^m = (\phi^{m-1}(g) \cdots \phi(g) \cdot \phi^2(g) \cdot g,\ \phi^m)$.
The first component is equal to $g^{k^{m-1}+\ldots+k+1} = g^{\frac{k^m-1}{k-1}}$.
The shared key $K = g^{\frac{k^{m+n}-1}{k-1}}$.

"The Diffie-Hellman type problem" would be to recover the shared key $K = g^{\frac{k^{m+n}-1}{k-1}}$ from the triple $(g,\ g^{\frac{k^m-1}{k-1}},\ g^{\frac{k^n-1}{k-1}})$. Since g and k are public, this is equivalent to recovering $g^{k^{m+n}}$ from the triple $(g,\ g^{k^m},\ g^{k^n})$, i.e., this is exactly the standard Diffie-Hellman problem.

Special case: Diffie-Hellman

$G = \mathbb{Z}_p^*$
$\phi(g) = g^k$ for all $g \in G$ and a fixed k, $1 < k < p-1$.

Then $(g, \phi)^m = (\phi^{m-1}(g) \cdots \phi(g) \cdot \phi^2(g) \cdot g,\ \phi^m)$.
The first component is equal to $g^{k^{m-1}+\ldots+k+1} = g^{\frac{k^m-1}{k-1}}$.
The shared key $K = g^{\frac{k^{m+n}-1}{k-1}}$.

"The Diffie-Hellman type problem" would be to recover the shared key $K = g^{\frac{k^{m+n}-1}{k-1}}$ from the triple $(g,\ g^{\frac{k^m-1}{k-1}},\ g^{\frac{k^n-1}{k-1}})$. Since g and k are public, this is equivalent to recovering $g^{k^{m+n}}$ from the triple $(g,\ g^{k^m},\ g^{k^n})$, i.e., this is exactly the standard Diffie-Hellman problem.

Special case: Diffie-Hellman

$G = \mathbb{Z}_p^*$
$\phi(g) = g^k$ for all $g \in G$ and a fixed k, $1 < k < p-1$.

Then $(g, \phi)^m = (\phi^{m-1}(g) \cdots \phi(g) \cdot \phi^2(g) \cdot g,\ \phi^m)$.
The first component is equal to $g^{k^{m-1}+\ldots+k+1} = g^{\frac{k^m-1}{k-1}}$.
The shared key $K = g^{\frac{k^{m+n}-1}{k-1}}$.

"The Diffie-Hellman type problem" would be to recover the shared key $K = g^{\frac{k^{m+n}-1}{k-1}}$ from the triple $(g,\ g^{\frac{k^m-1}{k-1}},\ g^{\frac{k^n-1}{k-1}})$. Since g and k are public, this is equivalent to recovering $g^{k^{m+n}}$ from the triple $(g,\ g^{k^m},\ g^{k^n})$, i.e., this is exactly the standard Diffie-Hellman problem.

Platform: matrices over group rings

Our general protocol can be used with *any* non-commutative group G if ϕ is selected to be an inner automorphism. Furthermore, it can be used with any non-commutative *semigroup* G as well, as long as G has some invertible elements; these can be used to produce inner automorphisms. A typical example of such a semigroup would be a semigroup of matrices over some ring.

We use the semigroup of 3×3 matrices over the group ring $\mathbb{Z}_7[A_5]$, where A_5 is the alternating group on 5 elements.
Then the public key consists of two matrices: the (invertible) conjugating matrix H and a (non-invertible) matrix M. The shared secret key then is:
$K = H^{-(m+n)}(HM)^{m+n}$.

Platform: matrices over group rings

Our general protocol can be used with *any* non-commutative group G if ϕ is selected to be an inner automorphism. Furthermore, it can be used with any non-commutative *semigroup* G as well, as long as G has some invertible elements; these can be used to produce inner automorphisms. A typical example of such a semigroup would be a semigroup of matrices over some ring.

We use the semigroup of 3×3 matrices over the group ring $\mathbb{Z}_7[A_5]$, where A_5 is the alternating group on 5 elements.
Then the public key consists of two matrices: the (invertible) conjugating matrix H and a (non-invertible) matrix M. The shared secret key then is:
$K = H^{-(m+n)}(HM)^{m+n}$.

Security assumptions

To recover $H^{-(m+n)}(HM)^{m+n}$ from $(M,\ H,\ H^{-m}(HM)^m,\ H^{-n}(HM)^n)$ is hard.

To recover m from $(M,\ H,\ H^{-m}(HM)^m)$ is hard.

Security assumptions

To recover $H^{-(m+n)}(HM)^{m+n}$ from $(M,\ H,\ H^{-m}(HM)^m,\ H^{-n}(HM)^n)$ is hard.

To recover m from $(M,\ H,\ H^{-m}(HM)^m)$ is hard.

Thank you