Isogeny-basd cryptography II

Key exchange on graphs

Tanja Lange
(with lots of slides by Lorenz Panny)

Eindhoven University of Technology

SAC – Post-quantum cryptography

Diffie–Hellman key exchange '76

Public parameters:

- a finite group G (traditionally $\mathbb{F}_p^*$, today elliptic curves)
- an element $g \in G$ of prime order q

Diffie–Hellman key exchange '76

Public parameters:

- a finite group G (traditionally $\mathbb{F}_p^*$, today elliptic curves)
- an element $g \in G$ of prime order q

Alice	public	Bob
$a \xleftarrow{\text{random}} \{0 \ldots q-1\}$		$b \xleftarrow{\text{random}} \{0 \ldots q-1\}$
g^a		g^b
$k = \text{hash}((g^b)^a)$		$k = \text{hash}((g^a)^b)$

Diffie–Hellman key exchange '76

Public parameters:

- a finite group G (traditionally $\mathbb{F}_p^*$, today elliptic curves)
- an element $g \in G$ of prime order q

Alice	public	Bob
$a \xleftarrow{\text{random}} \{0 \ldots q-1\}$		$b \xleftarrow{\text{random}} \{0 \ldots q-1\}$
g^a		g^b
$k = \text{hash}(((g^b)^a)$		$k = \text{hash}((g^a)^b)$

Fundamental reason this works: $\cdot^a$ and $\cdot^b$ commute!

Diffie–Hellman: Bob vs. Eve

<u>Bob</u>

1. Set $t \leftarrow g$.
2. Set $t \leftarrow t \cdot g$.
3. Set $t \leftarrow t \cdot g$.
4. Set $t \leftarrow t \cdot g$.

...

$b-2$. Set $t \leftarrow t \cdot g$.

$b-1$. Set $t \leftarrow t \cdot g$.

b. Publish $B \leftarrow t \cdot g$.

Diffie–Hellman: Bob vs. Eve

Bob

1. Set $t \leftarrow g$.
2. Set $t \leftarrow t \cdot g$.
3. Set $t \leftarrow t \cdot g$.
4. Set $t \leftarrow t \cdot g$.

...

$b-2$. Set $t \leftarrow t \cdot g$.

$b-1$. Set $t \leftarrow t \cdot g$.

b. Publish $B \leftarrow t \cdot g$.

Is this a good idea?

Diffie–Hellman: Bob vs. Eve

Bob

1. Set $t \leftarrow g$.
2. Set $t \leftarrow t \cdot g$.
3. Set $t \leftarrow t \cdot g$.
4. Set $t \leftarrow t \cdot g$.

...

$b-2$. Set $t \leftarrow t \cdot g$.

$b-1$. Set $t \leftarrow t \cdot g$.

b. Publish $B \leftarrow t \cdot g$.

Attacker Eve

1. Set $t \leftarrow g$. If $t = B$ return 1.
2. Set $t \leftarrow t \cdot g$. If $t = B$ return 2.
3. Set $t \leftarrow t \cdot g$. If $t = B$ return 3.
4. Set $t \leftarrow t \cdot g$. If $t = B$ return 3.

...

$b-2$. Set $t \leftarrow t \cdot g$. If $t = B$ return $b-2$.

$b-1$. Set $t \leftarrow t \cdot g$. If $t = B$ return $b-1$.

b. Set $t \leftarrow t \cdot g$. If $t = B$ return b.

$b+1$. Set $t \leftarrow t \cdot g$. If $t = B$ return $b+1$.

$b+2$. Set $t \leftarrow t \cdot g$. If $t = B$ return $b+2$.

...

Diffie–Hellman: Bob vs. Eve

Bob

1. Set $t \leftarrow g$.
2. Set $t \leftarrow t \cdot g$.
3. Set $t \leftarrow t \cdot g$.
4. Set $t \leftarrow t \cdot g$.

...

$b-2$. Set $t \leftarrow t \cdot g$.

$b-1$. Set $t \leftarrow t \cdot g$.

b. Publish $B \leftarrow t \cdot g$.

Attacker Eve

1. Set $t \leftarrow g$. If $t = B$ return 1.
2. Set $t \leftarrow t \cdot g$. If $t = B$ return 2.
3. Set $t \leftarrow t \cdot g$. If $t = B$ return 3.
4. Set $t \leftarrow t \cdot g$. If $t = B$ return 3.

...

$b-2$. Set $t \leftarrow t \cdot g$. If $t = B$ return $b-2$.

$b-1$. Set $t \leftarrow t \cdot g$. If $t = B$ return $b-1$.

b. Set $t \leftarrow t \cdot g$. If $t = B$ return b.

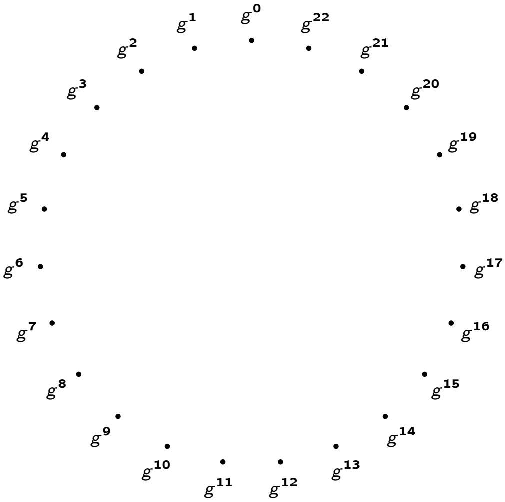
$b+1$. Set $t \leftarrow t \cdot g$. If $t = B$ return $b+1$.

$b+2$. Set $t \leftarrow t \cdot g$. If $t = B$ return $b+2$.

...

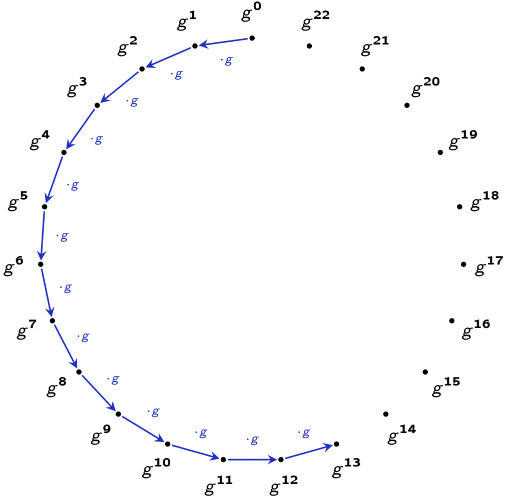
Effort for both: $O(\#G)$. Bob needs to be smarter.

(There also exist better attacks)



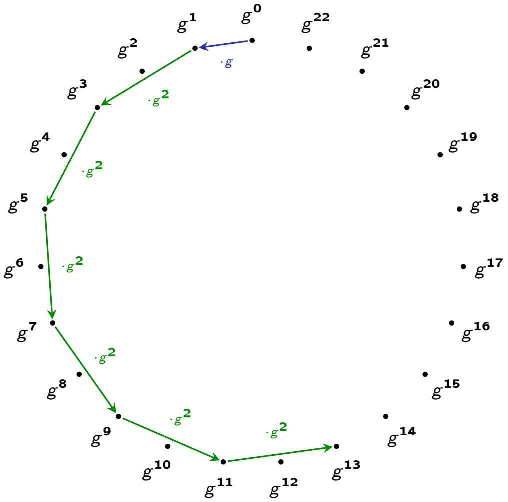
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

multiply



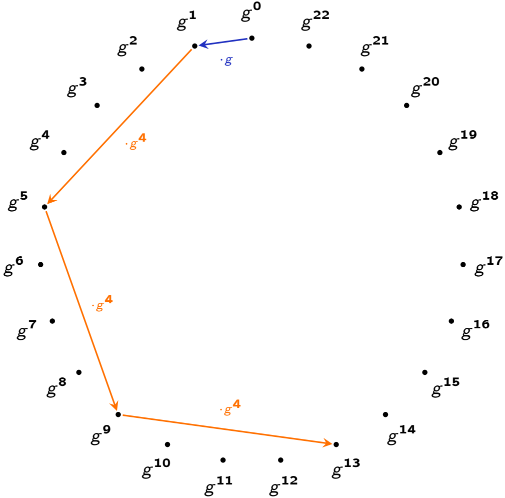
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply



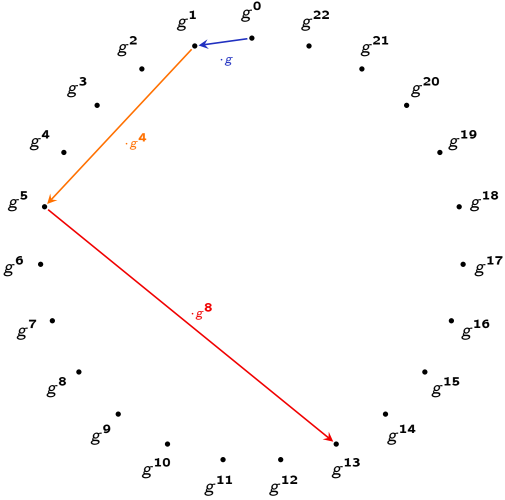
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply-and-square-and-multiply



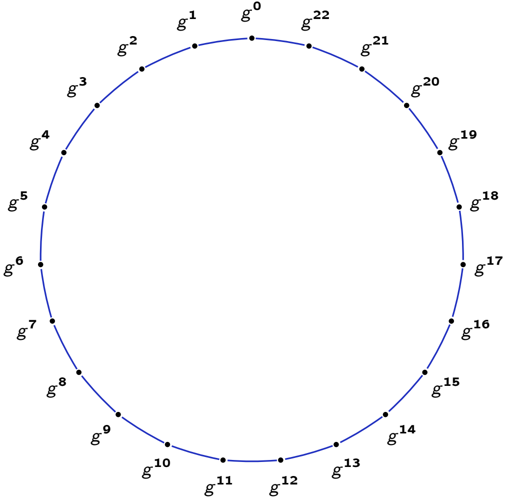
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply-and-square-and-multiply-and-square-and-r



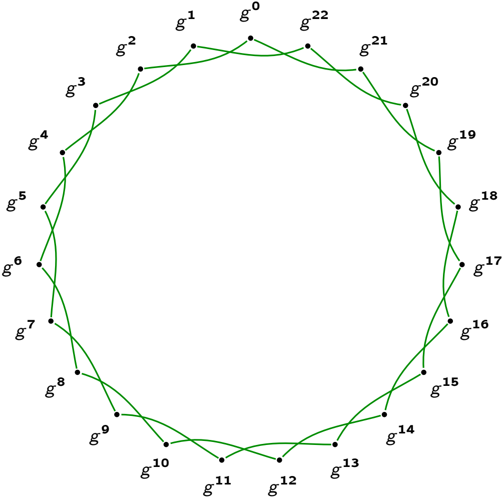
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply as graphs



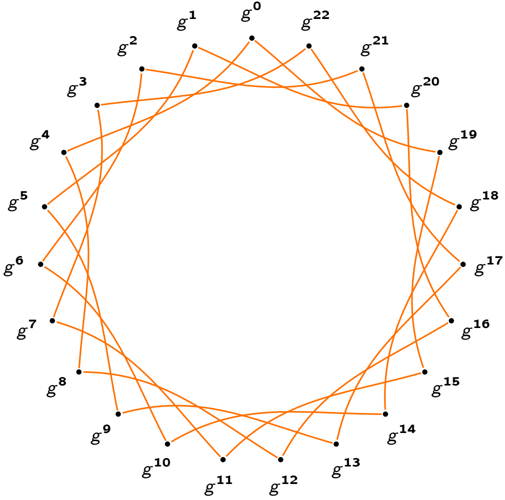
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply as graphs



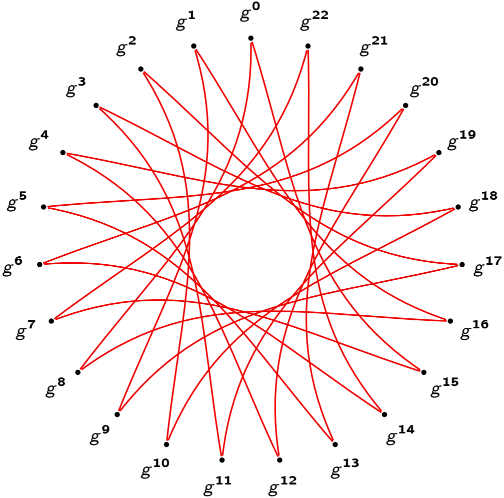
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply as graphs



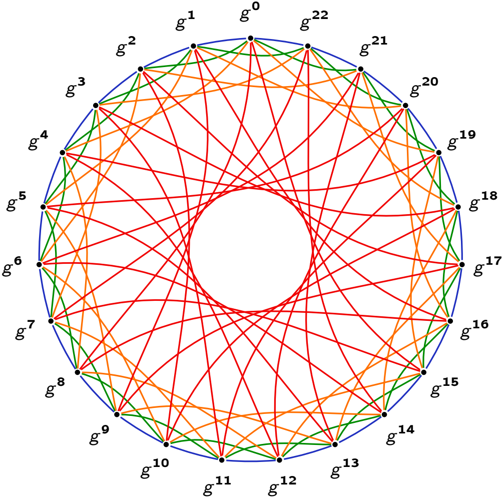
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply as graphs



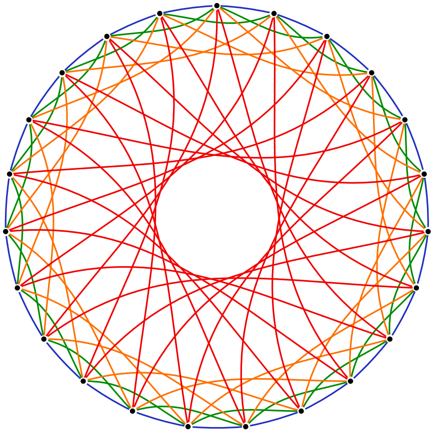
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply as a graph



Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply as a graph



Fast mixing: paths of length log(# nodes) to everywhere.

Exponential separation

Constructive computation:
With square-and-multiply, applying b takes $\Theta(\log_2 \#G)$.

Attack costs:
For well-chosen groups, recovering b takes $\Theta(\sqrt{\#G})$.

(For less-well chosen groups the attacks are faster.)

As

$$\sqrt{\#G} = 2^{0.5 \log_2 \#G}$$

attacks are exponentially harder.

Exponential separation until quantum computers come

Constructive computation:
With square-and-multiply, applying b takes $\Theta(\log_2 \#G)$.

Attack costs:
For well-chosen groups, recovering b takes $\Theta(\sqrt{\#G})$.

(For less-well chosen groups the attacks are faster.)

As

$$\sqrt{\#G} = 2^{0.5 \log_2 \#G}$$

attacks are exponentially harder.

On a sufficiently large quantum computer, Shor's algorithm quantumly computes b from g^b in any group in polynomial time.

Exponential separation until quantum computers come

Constructive computation:
With square-and-multiply, applying b takes $\Theta(\log_2 \#G)$.

Attack costs:
For well-chosen groups, recovering b takes $\Theta(\sqrt{\#G})$.

(For less-well chosen groups the attacks are faster.)

As

$$\sqrt{\#G} = 2^{0.5 \log_2 \#G}$$

attacks are exponentially harder.

On a sufficiently large quantum computer, Shor's algorithm quantumly computes b from g^b in any group in polynomial time.
Isogeny graphs to the rescue!

Big picture

- Isogenies are a source of exponentially-sized graphs.

Big picture

- Isogenies are a source of exponentially-sized graphs.
- We can walk efficiently on these graphs.

Big picture

- Isogenies are a source of exponentially-sized graphs.
- We can walk efficiently on these graphs.
- Fast mixing: short paths to (almost) all nodes.

Big picture

- Isogenies are a source of exponentially-sized graphs.
- We can walk efficiently on these graphs.
- Fast mixing: short paths to (almost) all nodes.
- No efficient* algorithms to recover paths from endpoints.
  (*Both* classical and quantum!)

Big picture

- Isogenies are a source of exponentially-sized graphs.
- We can walk efficiently on these graphs.
- Fast mixing: short paths to (almost) all nodes.
- No efficient* algorithms to recover paths from endpoints.
  (*Both* classical and quantum!)
- Enough structure to navigate the graph meaningfully.
  That is: some *well-behaved* "directions" to describe paths. More later.

Big picture

- Isogenies are a source of exponentially-sized graphs.
- We can walk efficiently on these graphs.
- Fast mixing: short paths to (almost) all nodes.
- No efficient* algorithms to recover paths from endpoints.
  (*Both* classical and quantum!)
- Enough structure to navigate the graph meaningfully.
  That is: some *well-behaved* "directions" to describe paths. More later.

It is easy to construct graphs that satisfy *almost* all of these — not enough for crypto!