Group Theory and Cryptography

Simon R. Blackburn

Joint work with Carlos Cid, Ciaran Mullan

Royal Holloway University of London

10th June 2010

Overview

Discrete logarithm problem

Let G be a group (represented in some specified way). For example, let p be a prime and let $G = \mathbb{Z}_p^*$.

Discrete logarithm problem

Let G be a group (represented in some specified way). For example, let p be a prime and let $G = \mathbb{Z}_p^*$.

The Discrete Logarithm Problem (DLP): Given $g, h \in G$, find an integer x (if it exists) such that $h = g^x$.

Discrete logarithm problem

Let G be a group (represented in some specified way). For example, let p be a prime and let $G = \mathbb{Z}_p^*$.

The Discrete Logarithm Problem (DLP): Given $g, h \in G$, find an integer x (if it exists) such that $h = g^x$.

- When $G = \mathbb{Z}_p^*$, this seems to be a hard computational problem (for large p).

Discrete logarithm problem

Let G be a group (represented in some specified way). For example, let p be a prime and let $G = \mathbb{Z}_p^*$.

The Discrete Logarithm Problem (DLP): Given $g, h \in G$, find an integer x (if it exists) such that $h = g^x$.

- When $G = \mathbb{Z}_p^*$, this seems to be a hard computational problem (for large p).
- Taking G the group of points of an elliptic curve is thought to be more secure (no 'index-calculus').

Discrete logarithm problem

Let G be a group (represented in some specified way). For example, let p be a prime and let $G = \mathbb{Z}_p^*$.

The Discrete Logarithm Problem (DLP): Given $g, h \in G$, find an integer x (if it exists) such that $h = g^x$.

- When $G = \mathbb{Z}_p^*$, this seems to be a hard computational problem (for large p).
- Taking G the group of points of an elliptic curve is thought to be more secure (no 'index-calculus').
- The DLP makes sense for any group. But it's really about cyclic groups, since we can replace G by $\langle g \rangle$.

Diffie-Hellman key exchange

Suppose Alice and Bob want to agree on a random key K.
They decide upon a large prime p and some $g \in G = \mathbb{Z}_p^*$ and perform the following protocol:

Diffie-Hellman key exchange

Suppose Alice and Bob want to agree on a random key K.
They decide upon a large prime p and some $g \in G = \mathbb{Z}_p^*$ and perform the following protocol:

- Alice chooses a random integer $0 \leq a < |G|$ and sends $c_1 = g^a$ to Bob.

Diffie-Hellman key exchange

Suppose Alice and Bob want to agree on a random key K.
They decide upon a large prime p and some $g \in G = \mathbb{Z}_p^*$ and perform the following protocol:

- Alice chooses a random integer $0 \leq a < |G|$ and sends $c_1 = g^a$ to Bob.
- Bob chooses a random integer $0 \leq b < |G|$ and sends $c_2 = g^b$ to Alice.

Diffie-Hellman key exchange

Suppose Alice and Bob want to agree on a random key K.
They decide upon a large prime p and some $g \in G = \mathbb{Z}_p^*$ and perform the following protocol:

- Alice chooses a random integer $0 \leq a < |G|$ and sends $c_1 = g^a$ to Bob.
- Bob chooses a random integer $0 \leq b < |G|$ and sends $c_2 = g^b$ to Alice.

Alice and Bob both share the same key $K = g^{ab}$.

Diffie-Hellman key exchange

Suppose Alice and Bob want to agree on a random key K.
They decide upon a large prime p and some $g \in G = \mathbb{Z}_p^*$ and perform the following protocol:

- Alice chooses a random integer $0 \leq a < |G|$ and sends $c_1 = g^a$ to Bob.
- Bob chooses a random integer $0 \leq b < |G|$ and sends $c_2 = g^b$ to Alice.
- On receiving c_2 Alice computes $K = c_2^a$.

Alice and Bob both share the same key $K = g^{ab}$.

Diffie-Hellman key exchange

Suppose Alice and Bob want to agree on a random key K.
They decide upon a large prime p and some $g \in G = \mathbb{Z}_p^*$ and perform the following protocol:

- Alice chooses a random integer $0 \leq a < |G|$ and sends $c_1 = g^a$ to Bob.
- Bob chooses a random integer $0 \leq b < |G|$ and sends $c_2 = g^b$ to Alice.
- On receiving c_2 Alice computes $K = c_2^a$.
- On receiving c_1 Bob computes $K = c_1^b$.

Alice and Bob both share the same key $K = g^{ab}$.

Diffie-Hellman key exchange

Suppose Alice and Bob want to agree on a random key K.
They decide upon a large prime p and some $g \in G = \mathbb{Z}_p^*$ and perform the following protocol:

- Alice chooses a random integer $0 \leq a < |G|$ and sends $c_1 = g^a$ to Bob.
- Bob chooses a random integer $0 \leq b < |G|$ and sends $c_2 = g^b$ to Alice.
- On receiving c_2 Alice computes $K = c_2^a$.
- On receiving c_1 Bob computes $K = c_1^b$.

Alice and Bob both share the same key $K = g^{ab}$.

Alice and Bob have the same key: $(g^a)^b = (g^b)^a$.

Finding hard instances of the DLP?

- The DLP makes sense for any cyclic group.

Finding hard instances of the DLP?

- The DLP makes sense for any cyclic group.
- The DLP seems hard sometimes: $G = \mathbb{Z}_p^*$, or $G = E(\mathbb{F}_q)$.

Finding hard instances of the DLP?

- The DLP makes sense for any cyclic group.
- The DLP seems hard sometimes: $G = \mathbb{Z}_p^*$, or $G = E(\mathbb{F}_q)$.
- But not always hard: $G = (\mathbb{Z}_p, +)$.

Finding hard instances of the DLP?

- The DLP makes sense for any cyclic group.
- The DLP seems hard sometimes: $G = \mathbb{Z}_p^*$, or $G = E(\mathbb{F}_q)$.
- But not always hard: $G = (\mathbb{Z}_p, +)$.
- Another bad idea: $G \leq \mathrm{GL}(n, q)$.

Finding hard instances of the DLP?

- The DLP makes sense for any cyclic group.
- The DLP seems hard sometimes: $G = \mathbb{Z}_p^*$, or $G = E(\mathbb{F}_q)$.
- But not always hard: $G = (\mathbb{Z}_p, +)$.
- Another bad idea: $G \leq \mathrm{GL}(n, q)$.
- But what about non-abelian groups?

Ko Lee Cheon Han Kang Park

- Let G be a (non-abelian) group. For $a, g \in G$ define

$$g^a = a^{-1}ga.$$

Ko Lee Cheon Han Kang Park

- Let G be a (non-abelian) group. For $a, g \in G$ define
$$g^a = a^{-1}ga.$$
- Problem: $(g^a)^b \neq (g^b)^a$, in general.

Ko Lee Cheon Han Kang Park

- Let G be a (non-abelian) group. For $a, g \in G$ define

$$g^a = a^{-1}ga.$$

- Problem: $(g^a)^b \neq (g^b)^a$, in general.
- Solution: Choose $a \in A \leq G$ and $b \in B \leq G$ where $[A, B] = \{1\}$.

Ko Lee Cheon Han Kang Park

- Let G be a (non-abelian) group. For $a, g \in G$ define
$$g^a = a^{-1}ga.$$

- Problem: $(g^a)^b \neq (g^b)^a$, in general.
- Solution: Choose $a \in A \leq G$ and $b \in B \leq G$ where $[A, B] = \{1\}$.
- How do you choose a group G and subgroups A and B?

Ko Lee Cheon Han Kang Park

- Let G be a (non-abelian) group. For $a, g \in G$ define

$$g^a = a^{-1}ga.$$

- Problem: $(g^a)^b \neq (g^b)^a$, in general.
- Solution: Choose $a \in A \leq G$ and $b \in B \leq G$ where $[A, B] = \{1\}$.
- How do you choose a group G and subgroups A and B?
- Ko et al. suggest using a braid group.

Braid group cryptography

- How difficult is the *conjugacy search problem*?
- There's a nice survey: 'Braid based cryptography' by Patrick Dehornoy.

Braid group cryptography

- How difficult is the *conjugacy search problem*?
- There's a nice survey: 'Braid based cryptography' by Patrick Dehornoy.
- Length based attacks work for many instances.

Braid group cryptography

- How difficult is the *conjugacy search problem*?
- There's a nice survey: 'Braid based cryptography' by Patrick Dehornoy.
- Length based attacks work for many instances.
- How can we generate hard instances?

Braid group cryptography

- How difficult is the *conjugacy search problem*?
- There's a nice survey: 'Braid based cryptography' by Patrick Dehornoy.
- Length based attacks work for many instances.
- How can we generate hard instances?
- There are no braid based schemes that are competitive with (say) elliptic curve DLP-based schemes.

Anshel, Anshel, Goldfeld

- Let G be a group generated by $x_1, x_2, \ldots, x_n$.

Anshel, Anshel, Goldfeld

- Let G be a group generated by $x_1, x_2, \ldots, x_n$.
- Alice chooses a product $a \in G$ of generators and their inverses:

$$a = x_{i_1}^{\epsilon_1} x_{i_2}^{\epsilon_2} \cdots x_{i_k}^{\epsilon_k}$$

with $i_j \in \{1, 2, \ldots, n\}$ and $\epsilon_j = \pm 1$.

Anshel, Anshel, Goldfeld

- Let G be a group generated by $x_1, x_2, \ldots, x_n$.
- Alice chooses a product $a \in G$ of generators and their inverses:

$$a = x_{i_1}^{\epsilon_1} x_{i_2}^{\epsilon_2} \cdots x_{i_k}^{\epsilon_k}$$

  with $i_j \in \{1, 2, \ldots, n\}$ and $\epsilon_j = \pm 1$.
- Alice sends $x_1^a, x_2^a, \ldots, x_n^a$ to Bob.

Anshel, Anshel, Goldfeld

- Let G be a group generated by $x_1, x_2, \ldots, x_n$.
- Alice chooses a product $a \in G$ of generators and their inverses:

$$a = x_{i_1}^{\epsilon_1} x_{i_2}^{\epsilon_2} \cdots x_{i_k}^{\epsilon_k}$$

  with $i_j \in \{1, 2, \ldots, n\}$ and $\epsilon_j = \pm 1$.
- Alice sends $x_1^a, x_2^a, \ldots, x_n^a$ to Bob.
- Bob chooses $b \in G$ and sends $x_1^b, x_2^b, \ldots, x_n^b$ to Alice.

Anshel, Anshel, Goldfeld

- Let G be a group generated by $x_1, x_2, \ldots, x_n$.
- Alice chooses a product $a \in G$ of generators and their inverses:

$$a = x_{i_1}^{\epsilon_1} x_{i_2}^{\epsilon_2} \cdots x_{i_k}^{\epsilon_k}$$

  with $i_j \in \{1, 2, \ldots, n\}$ and $\epsilon_j = \pm 1$.
- Alice sends $x_1^a, x_2^a, \ldots, x_n^a$ to Bob.
- Bob chooses $b \in G$ and sends $x_1^b, x_2^b, \ldots, x_n^b$ to Alice.
- Common key:

$$[a, b] = a^{-1} b^{-1} ab.$$

Anshel, Anshel, Goldfeld

- Let G be a group generated by $x_1, x_2, \ldots, x_n$.
- Alice chooses a product $a \in G$ of generators and their inverses:

$$a = x_{i_1}^{\epsilon_1} x_{i_2}^{\epsilon_2} \cdots x_{i_k}^{\epsilon_k}$$

  with $i_j \in \{1, 2, \ldots, n\}$ and $\epsilon_j = \pm 1$.
- Alice sends $x_1^a, x_2^a, \ldots, x_n^a$ to Bob.
- Bob chooses $b \in G$ and sends $x_1^b, x_2^b, \ldots, x_n^b$ to Alice.
- Common key:

$$[a, b] = a^{-1} b^{-1} ab.$$

Anshel, Anshel, Goldfeld

- Why can Alice calculate $[a, b]$?

Anshel, Anshel, Goldfeld

- Why can Alice calculate $[a, b]$?
- She knows

$$b^{-1}ab = (b^{-1}x_{i_1}b)^{\epsilon_1}(b^{-1}x_{i_2}b)^{\epsilon_2}\cdots(b^{-1}x_{i_k}b)^{\epsilon_k}.$$

Anshel, Anshel, Goldfeld

- Why can Alice calculate $[a, b]$?
- She knows

$$b^{-1}ab = (b^{-1}x_{i_1}b)^{\epsilon_1}(b^{-1}x_{i_2}b)^{\epsilon_2}\cdots(b^{-1}x_{i_k}b)^{\epsilon_k}.$$

- So she knows $a^{-1}(b^{-1}ab) = [a, b]$.

Anshel, Anshel, Goldfeld

- Why can Alice calculate $[a, b]$?
- She knows

$$b^{-1}ab = (b^{-1}x_{i_1}b)^{\epsilon_1}(b^{-1}x_{i_2}b)^{\epsilon_2}\cdots(b^{-1}x_{i_k}b)^{\epsilon_k}.$$

- So she knows $a^{-1}(b^{-1}ab) = [a, b]$.
- Similarly: Bob knows $a^{-1}ba$,

Anshel, Anshel, Goldfeld

- Why can Alice calculate $[a, b]$?
- She knows

$$b^{-1}ab = (b^{-1}x_{i_1}b)^{\epsilon_1}(b^{-1}x_{i_2}b)^{\epsilon_2}\cdots(b^{-1}x_{i_k}b)^{\epsilon_k}.$$

- So she knows $a^{-1}(b^{-1}ab) = [a, b]$.
- Similarly: Bob knows $a^{-1}ba$, so knows $a^{-1}b^{-1}a$,

Anshel, Anshel, Goldfeld

- Why can Alice calculate $[a, b]$?
- She knows

$$b^{-1}ab = (b^{-1}x_{i_1}b)^{\epsilon_1}(b^{-1}x_{i_2}b)^{\epsilon_2}\cdots(b^{-1}x_{i_k}b)^{\epsilon_k}.$$

- So she knows $a^{-1}(b^{-1}ab) = [a, b]$.
- Similarly: Bob knows $a^{-1}ba$, so knows $a^{-1}b^{-1}a$, so knows $(a^{-1}b^{-1}a)b = [a, b]$.

Covers

- A cover is an (ordered) collection of sets

$$A_1, A_2, \ldots, A_s \subseteq G$$

such that for all $h \in G$

$$h = a_1 a_2 \cdots a_s$$

in at least one way, where $a_i \in A_i$.

Covers

- A cover is an (ordered) collection of sets

$$A_1, A_2, \ldots, A_s \subseteq G$$

such that for all $h \in G$

$$h = a_1 a_2 \cdots a_s$$

in at least one way, where $a_i \in A_i$.

- Example in $G = \mathbb{Z}_2^3$:

$$A_1 = \{000, 001, 100, 101\} \; A_2 = \{000, 111, 011\}.$$

Covers

- A cover is an (ordered) collection of sets

$$A_1, A_2, \ldots, A_s \subseteq G$$

such that for all $h \in G$

$$h = a_1 a_2 \cdots a_s$$

in at least one way, where $a_i \in A_i$.

- Example in $G = \mathbb{Z}_2^3$:

$$A_1 = \{000, 001, 100, 101\} \; A_2 = \{000, 111, 011\}.$$

- This generalises the DLP:

$$A_i = \{1, g^{2^i}\} \text{ for } 1 \leq i \leq \log_2 |\langle g \rangle|.$$

is a cover for $\langle g \rangle$.

Logarithmic signatures

- A logarithmic signature or factorisation is an (ordered) collection of sets
$$A_1, A_2, \ldots, A_s \subseteq G$$
such that for all $h \in G$
$$h = a_1 a_2 \cdots a_s$$
in exactly one way, where $a_i \in A_i$.

Logarithmic signatures

- A logarithmic signature or factorisation is an (ordered) collection of sets
$$A_1, A_2, \ldots, A_s \subseteq G$$
such that for all $h \in G$
$$h = a_1 a_2 \cdots a_s$$
in exactly one way, where $a_i \in A_i$.
- Example in $G = \mathbb{Z}_2^3$:
$$A_1 = \{000, 001, 100, 101\} \; A_2 = \{000, 111\}.$$

Logarithmic signatures

- A logarithmic signature or factorisation is an (ordered) collection of sets
$$A_1, A_2, \ldots, A_s \subseteq G$$
such that for all $h \in G$
$$h = a_1 a_2 \cdots a_s$$
in exactly one way, where $a_i \in A_i$.
- Example in $G = \mathbb{Z}_2^3$:
$$A_1 = \{000, 001, 100, 101\} \; A_2 = \{000, 111\}.$$
- More general example (transversal logarithmic signatures) : Given a chain
$$1 = H_0 < H_1 < \cdots < H_s = G$$
of subgroups, set A_i to be a complete set of coset representatives for H_i in H_{i-1}.

Logarithmic signatures

- Let $A_1, A_2, \ldots, A_s$ be a logarithmic signature, with $|A_i| = r_i$. Then

$$|G| = r_1 r_2 \cdots r_s.$$

Logarithmic signatures

- Let $A_1, A_2, \ldots, A_s$ be a logarithmic signature, with $|A_i| = r_i$. Then
$$|G| = r_1 r_2 \cdots r_s.$$
- The logarithmic signature induces a bijective function
$$\breve{\alpha} : \mathbb{Z}_{r_1} \times \mathbb{Z}_{r_2} \times \cdots \times \mathbb{Z}_{r_s} \to G$$
defined by
$$\breve{\alpha}(k_1, k_2, \ldots, k_r) = a_1 a_2 \ldots a_r$$
where a_i is the k_ith element of A_i.

Logarithmic signatures

- Let $A_1, A_2, \ldots, A_s$ be a logarithmic signature, with $|A_i| = r_i$. Then
$$|G| = r_1 r_2 \cdots r_s.$$
- The logarithmic signature induces a bijective function
$$\breve{\alpha} : \mathbb{Z}_{r_1} \times \mathbb{Z}_{r_2} \times \cdots \times \mathbb{Z}_{r_s} \to G$$
defined by
$$\breve{\alpha}(k_1, k_2, \ldots, k_r) = a_1 a_2 \ldots a_r$$
where a_i is the k_ith element of A_i.
- If $\breve{\alpha}$ is easy to invert, we say that α is tame.

Logarithmic signatures

- Let $A_1, A_2, \dots, A_s$ be a logarithmic signature, with $|A_i| = r_i$. Then
$$|G| = r_1 r_2 \cdots r_s.$$
- The logarithmic signature induces a bijective function
$$\breve{\alpha} : \mathbb{Z}_{r_1} \times \mathbb{Z}_{r_2} \times \cdots \times \mathbb{Z}_{r_s} \to G$$
defined by
$$\breve{\alpha}(k_1, k_2, \dots, k_r) = a_1 a_2 \dots a_r$$
where a_i is the k_ith element of A_i.
- If $\breve{\alpha}$ is easy to invert, we say that α is tame.
- If we have a cover, the function $\breve{\alpha}$ makes sense and is onto, but is not necessarily injective.

(Simplified) MST_3

- Let G be a group with centre Z. Let random $A_1, A_2, \ldots, A_s \subseteq G$ have $\prod |A_i| = |Z|$. These are public.

(Simplified) MST_3

- Let G be a group with centre Z. Let random $A_1, A_2, \ldots, A_s \subseteq G$ have $\prod |A_i| = |Z|$. These are public.
- Let $B_1, B_2, \ldots, B_s$ be a secret tame logarithmic signature for Z with $|B_i| = |A_i|$. Let $t \in G$ be secret.

(Simplified) MST_3

- Let G be a group with centre Z. Let random $A_1, A_2, \ldots, A_s \subseteq G$ have $\prod |A_i| = |Z|$. These are public.
- Let $B_1, B_2, \ldots, B_s$ be a secret tame logarithmic signature for Z with $|B_i| = |A_i|$. Let $t \in G$ be secret.
- Idea: use $t^{-1}A_i t$ to disguise B_i.

(Simplified) MST_3

- Let G be a group with centre Z. Let random $A_1, A_2, \ldots, A_s \subseteq G$ have $\prod |A_i| = |Z|$. These are public.
- Let $B_1, B_2, \ldots, B_s$ be a secret tame logarithmic signature for Z with $|B_i| = |A_i|$. Let $t \in G$ be secret.
- Idea: use $t^{-1}A_i t$ to disguise B_i.
- If $A_i = \{a_{i,1}, \ldots, a_{i,r_i}\}$ and $B_i = \{b_{i,1}, \ldots, b_{i,r_i}\}$ define $G_i = B_i \cdot (t^{-1}A_i t) = \{b_{i,j}t^{-1}a_{i,j}t\}$.

(Simplified) MST_3

- Let G be a group with centre Z. Let random $A_1, A_2, \dots, A_s \subseteq G$ have $\prod |A_i| = |Z|$. These are public.
- Let $B_1, B_2, \dots, B_s$ be a secret tame logarithmic signature for Z with $|B_i| = |A_i|$. Let $t \in G$ be secret.
- Idea: use $t^{-1}A_i t$ to disguise B_i.
- If $A_i = \{a_{i,1}, \dots, a_{i,r_i}\}$ and $B_i = \{b_{i,1}, \dots, b_{i,r_i}\}$ define $G_i = B_i \cdot (t^{-1}A_i t) = \{b_{i,j} t^{-1} a_{i,j} t\}$.
- Public key: the sets G_i. Secret key: t and the sets B_i.

(Simplified) MST_3

- Let G be a group with centre Z. Let random $A_1, A_2, \ldots, A_s \subseteq G$ have $\prod |A_i| = |Z|$. These are public.
- Let $B_1, B_2, \ldots, B_s$ be a secret tame logarithmic signature for Z with $|B_i| = |A_i|$. Let $t \in G$ be secret.
- Idea: use $t^{-1}A_i t$ to disguise B_i.
- If $A_i = \{a_{i,1}, \ldots, a_{i,r_i}\}$ and $B_i = \{b_{i,1}, \ldots, b_{i,r_i}\}$ define $G_i = B_i \cdot (t^{-1}A_i t) = \{b_{i,j}t^{-1}a_{i,j}t\}$.
- Public key: the sets G_i. Secret key: t and the sets B_i.
- Encrypt x as the pair $(\breve{\alpha}(x), \breve{\gamma}(x))$.

(Simplified) MST_3

- Let G be a group with centre Z. Let random $A_1, A_2, \ldots, A_s \subseteq G$ have $\prod |A_i| = |Z|$. These are public.
- Let $B_1, B_2, \ldots, B_s$ be a secret tame logarithmic signature for Z with $|B_i| = |A_i|$. Let $t \in G$ be secret.
- Idea: use $t^{-1}A_i t$ to disguise B_i.
- If $A_i = \{a_{i,1}, \ldots, a_{i,r_i}\}$ and $B_i = \{b_{i,1}, \ldots, b_{i,r_i}\}$ define $G_i = B_i \cdot (t^{-1}A_i t) = \{b_{i,j} t^{-1} a_{i,j} t\}$.
- Public key: the sets G_i. Secret key: t and the sets B_i.
- Encrypt x as the pair $(\breve{\alpha}(x), \breve{\gamma}(x))$.
- To Decrypt (y_1, y_2): use $y_2 = \breve{\beta}(x)\, t^{-1}\breve{\alpha}(x)t$.

Platform groups and practicalities

- Lempken, Magliveras, van Trung, Wei suggest G should be a Suzuki 2-group. Let q be a power of 2, and $\theta \in \mathrm{Aut}(\mathbb{F}_q)$. Then:

$$G = \left\{ \begin{pmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ b & a^\theta & 1 \end{pmatrix} \text{ where } a, b \in \mathbb{F}_q \right\}.$$

Platform groups and practicalities

- Lempken, Magliveras, van Trung, Wei suggest G should be a Suzuki 2-group. Let q be a power of 2, and $\theta \in \mathrm{Aut}(\mathbb{F}_q)$. Then:

$$G = \left\{ \begin{pmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ b & a^\theta & 1 \end{pmatrix} \text{ where } a, b \in \mathbb{F}_q \right\}.$$

- Z consists of the matrices with $a = 0$.

Platform groups and practicalities

- Lempken, Magliveras, van Trung, Wei suggest G should be a Suzuki 2-group. Let q be a power of 2, and $\theta \in \mathrm{Aut}(\mathbb{F}_q)$. Then:

$$G = \left\{ \begin{pmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ b & a^\theta & 1 \end{pmatrix} \text{ where } a, b \in \mathbb{F}_q \right\}.$$

- Z consists of the matrices with $a = 0$.
- G/Z is abelian of order q, and Z also has order q.

Platform groups and practicalities

- Lempken, Magliveras, van Trung, Wei suggest G should be a Suzuki 2-group. Let q be a power of 2, and $\theta \in \mathrm{Aut}(\mathbb{F}_q)$. Then:

$$G = \left\{ \begin{pmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ b & a^\theta & 1 \end{pmatrix} \text{ where } a, b \in \mathbb{F}_q \right\}.$$

- Z consists of the matrices with $a = 0$.
- G/Z is abelian of order q, and Z also has order q.
- Unanswered question: How to generate $B_1, B_2, \ldots, B_s$.

A simple attack

- Lempken et al. gave several $O(q^2)$ attacks. Magliveras, Svaba, van Trung and Zajac give the following attack.

A simple attack

- Lempken et al. gave several $O(q^2)$ attacks. Magliveras, Svaba, van Trung and Zajac give the following attack.
- We are given covers A_i and G_i, where

$$G_i = B_i \cdot (t^{-1} A_i t).$$

The decryption process uses t and B_i.

A simple attack

- Lempken et al. gave several $O(q^2)$ attacks. Magliveras, Svaba, van Trung and Zajac give the following attack.
- We are given covers A_i and G_i, where

$$G_i = B_i \cdot (t^{-1} A_i t).$$

  The decryption process uses t and B_i.
- The attack is: Guess t. Find B_i from the equation above.

A simple attack

- Lempken et al. gave several $O(q^2)$ attacks. Magliveras, Svaba, van Trung and Zajac give the following attack.
- We are given covers A_i and G_i, where

$$G_i = B_i \cdot (t^{-1} A_i t).$$

  The decryption process uses t and B_i.
- The attack is: Guess t. Find B_i from the equation above.
- There are $|G| = q^2$ possibilities for t. But we only need to check $|G/Z| = q$ possibilities: it's only t modulo Z that matters.

A simple attack

- Lempken et al. gave several $O(q^2)$ attacks. Magliveras, Svaba, van Trung and Zajac give the following attack.
- We are given covers A_i and G_i, where

$$G_i = B_i \cdot (t^{-1} A_i t).$$

  The decryption process uses t and B_i.
- The attack is: Guess t. Find B_i from the equation above.
- There are $|G| = q^2$ possibilities for t. But we only need to check $|G/Z| = q$ possibilities: it's only t modulo Z that matters.
- This is an exponential attack. It works for any group G.

A simple attack

- Lempken et al. gave several $O(q^2)$ attacks. Magliveras, Svaba, van Trung and Zajac give the following attack.
- We are given covers A_i and G_i, where

$$G_i = B_i \cdot (t^{-1} A_i t).$$

  The decryption process uses t and B_i.
- The attack is: Guess t. Find B_i from the equation above.
- There are $|G| = q^2$ possibilities for t. But we only need to check $|G/Z| = q$ possibilities: it's only t modulo Z that matters.
- This is an exponential attack. It works for any group G.
- More practical attacks use knowledge of how the B_i are generated.

Generating a logarithmic signature

- We want to generate a tame logarithmic signature for

$$Z = \mathbb{Z}_2 \times \mathbb{Z}_2 \times \cdots \times \mathbb{Z}_2.$$

Generating a logarithmic signature

- We want to generate a tame logarithmic signature for

$$Z = \mathbb{Z}_2 \times \mathbb{Z}_2 \times \cdots \times \mathbb{Z}_2.$$

- Amalgamated Transversal Logarithmic Signatures (ATLS)

Generating a logarithmic signature

- We want to generate a tame logarithmic signature for
$$Z = \mathbb{Z}_2 \times \mathbb{Z}_2 \times \cdots \times \mathbb{Z}_2.$$
- Amalgamated Transversal Logarithmic Signatures (ATLS)
- Start with a chain $1 = H_0 < H_1 < H_2 < \cdots < H_k = Z$ of subgroups.

Generating a logarithmic signature

- We want to generate a tame logarithmic signature for

$$Z = \mathbb{Z}_2 \times \mathbb{Z}_2 \times \cdots \times \mathbb{Z}_2.$$

- Amalgamated Transversal Logarithmic Signatures (ATLS)
- Start with a chain $1 = H_0 < H_1 < H_2 < \cdots < H_k = Z$ of subgroups.
- Set X_i to be a set of coset representatives for H_i in H_{i-1}.

Generating a logarithmic signature

- We want to generate a tame logarithmic signature for

$$Z = \mathbb{Z}_2 \times \mathbb{Z}_2 \times \cdots \times \mathbb{Z}_2.$$

- Amalgamated Transversal Logarithmic Signatures (ATLS)
- Start with a chain $1 = H_0 < H_1 < H_2 < \cdots < H_k = Z$ of subgroups.
- Set X_i to be a set of coset representatives for H_i in H_{i-1}.
- Do some of the following operations:
  - Shift: replace X_i by $X_i z$ for some $z \in Z$.
  - Permute: swap X_i and X_j; permute the elements in X_i.
  - Amalgamate: replace X_i and X_j by

$$X_i X_j = \{x_i x_j \mid x_i \in X_i \text{ and } x_j \in X_j\}.$$

The security of ATLS-MST$_3$

- Suppose we generate an ATLS $B_1, B_2, \ldots, B_s$ to use in MST$_3$.

The security of ATLS-MST_3

- Suppose we generate an ATLS $B_1, B_2, \ldots, B_s$ to use in MST_3.
- Without loss of generality, we can assume the ATLS is normalised: $1 \in B_i$ for all i.

The security of ATLS-MST$_3$

- Suppose we generate an ATLS $B_1, B_2, \ldots, B_s$ to use in MST$_3$.
- Without loss of generality, we can assume the ATLS is normalised: $1 \in B_i$ for all i.
- For a normalised ATLS, there exists $b = b_{j,\ell} \in B_j$ (for some j) such that $B_j b = B_j$.

The security of ATLS-MST_3

- Suppose we generate an ATLS $B_1, B_2, \ldots, B_s$ to use in MST_3.
- Without loss of generality, we can assume the ATLS is normalised: $1 \in B_i$ for all i.
- For a normalised ATLS, there exists $b = b_{j,\ell} \in B_j$ (for some j) such that $B_j b = B_j$.
- Recall that we are given G_i, A_i and want to find B_i, t so that

$$G_i = B_i \cdot (t^{-1} A_i t).$$

The security of ATLS-MST_3

- Suppose we generate an ATLS $B_1, B_2, \ldots, B_s$ to use in MST_3.
- Without loss of generality, we can assume the ATLS is normalised: $1 \in B_i$ for all i.
- For a normalised ATLS, there exists $b = b_{j,\ell} \in B_j$ (for some j) such that $B_j b = B_j$.
- Recall that we are given G_i, A_i and want to find B_i, t so that

$$G_i = B_i \cdot (t^{-1} A_i t).$$

- If we guess j and ℓ (few choices) the condition $B_j b = B_j$ usually imposes very strong restrictions on t (so exhaustive search is possible).

The security of ATLS-MST_3

- Suppose we generate an ATLS $B_1, B_2, \ldots, B_s$ to use in MST_3.
- Without loss of generality, we can assume the ATLS is normalised: $1 \in B_i$ for all i.
- For a normalised ATLS, there exists $b = b_{j,\ell} \in B_j$ (for some j) such that $B_j b = B_j$.
- Recall that we are given G_i, A_i and want to find B_i, t so that

$$G_i = B_i \cdot (t^{-1} A_i t).$$

- If we guess j and ℓ (few choices) the condition $B_j b = B_j$ usually imposes very strong restrictions on t (so exhaustive search is possible).
- This attack works for practical parameter sizes.

Can the security of MST_3 be saved?

- Can you avoid this attack by generating β in some other way?

Can the security of MST_3 be saved?

- Can you avoid this attack by generating β in some other way?
- β cannot be periodic (cannot have $bB_i = B_i$ for some $b \in B_i$).
- β should have full rank (must have $\langle \bigcup_{j \neq i} B_j \rangle = G$ for all i).

Can the security of MST_3 be saved?

- Can you avoid this attack by generating β in some other way?
- β cannot be periodic (cannot have $bB_i = B_i$ for some $b \in B_i$).
- β should have full rank (must have $\langle \bigcup_{j \neq i} B_j \rangle = G$ for all i).
- What can be said about logarithmic signatures of finite elementary abelian 2-groups?

Can the security of MST_3 be saved?

- Can you avoid this attack by generating β in some other way?
- β cannot be periodic (cannot have $bB_i = B_i$ for some $b \in B_i$).
- β should have full rank (must have $\langle \bigcup_{j \neq i} B_j \rangle = G$ for all i).
- What can be said about logarithmic signatures of finite elementary abelian 2-groups?
- Related to perfect codes: Cohen, Litsyn, Vardy, Zémor 1996.

Summing up

- We've reviewed some of the cryptographic primitives based on groups (finite or infinite).
- We've sketched some problems with the security of MST_3.

Summing up

- We've reviewed some of the cryptographic primitives based on groups (finite or infinite).
- We've sketched some problems with the security of MST_3.
- Logarithmic signatures have applications to tiling problems; to perfect Gray codes and to other combinatorial problems in computer science.

Summing up

- We've reviewed some of the cryptographic primitives based on groups (finite or infinite).
- We've sketched some problems with the security of MST_3.
- Logarithmic signatures have applications to tiling problems; to perfect Gray codes and to other combinatorial problems in computer science.
- Can MST_3, or any other group-theoretic cryptosystem, be made both secure and practical?

Some Links

This talk will appear soon on my home page:

http://www.ma.rhul.ac.uk/sblackburn

S.R. Blackburn, C. Cid, C. Mullan, 'Group theory in cryptography' (*Proc. Groups St Andrews at Bath 2009*, to appear) is available at:

http://arxiv.org/abs/0906.5545

The paper 'Cryptanalysis of the MST_3 public key cryptosystem' (*J. Math Cryptology*, 2010) is also available at:

http://eprint.iacr.org/2009/248