Challenges in Proving Hard-Core Predicates for a Diffie-Hellman Problem

N. Fazio[1,2] R. Gennaro[1,2] I.M. Perera[2] W.E. Skeith III[1,2]

[1]The City College of CUNY
{fazio,rosario,wes}@cs.ccny.cuny.edu

[2]The Graduate Center of CUNY
iperera@gc.cuny.edu

April 10, 2013

Our Results

Result 1: Bit-security of Diffie-Hellman over Elliptic Curves

If Diffie-Hellman (DH) problem over elliptic curves (EC) is hard, every bit* of the secret Diffie-Hellman value is unpredictable.

Result 2: Bit-security of (Partial) DH over Finite Fields

Extension of Result 1 to (partial) DH problem over the finite field $\mathbb{F}_{p^2}$.

Result 3: Bit-security of Finite Field-based Partial OWF

Every bit* of the input to a finite field-based partial one-way function (FFB-POWF) is unpredictable.

Our Results

Result 1: Bit-security of Diffie-Hellman over Elliptic Curves

If Diffie-Hellman (DH) problem over elliptic curves (EC) is hard, every bit* of the secret Diffie-Hellman value is unpredictable.

Result 2: Bit-security of (Partial) DH over Finite Fields

Extension of Result 1 to (partial) DH problem over the finite field $\mathbb{F}_{p^2}$.

Result 3: Bit-security of Finite Field-based Partial OWF

Every bit* of the input to a finite field-based partial one-way function (FFB-POWF) is unpredictable.

Our Results

Result 1: Bit-security of Diffie-Hellman over Elliptic Curves

If Diffie-Hellman (DH) problem over elliptic curves (EC) is hard, every bit* of the secret Diffie-Hellman value is unpredictable.

Result 2: Bit-security of (Partial) DH over Finite Fields

Extension of Result 1 to (partial) DH problem over the finite field $\mathbb{F}_{p^2}$.

Result 3: Bit-security of Finite Field-based Partial OWF

Every bit* of the input to a finite field-based partial one-way function (FFB-POWF) is unpredictable.

One-way Functions and Hard-core Predicates

One-way Function

- $f : \mathcal{X} \to \mathcal{Y}$ is a one-way function (OWF) iff
  1. It is easy to compute $f(x)$ given $x \in \mathcal{X}$
  2. It is hard to invert, i.e.,

$$\forall \text{ PPT } \mathcal{A} \qquad \Pr_{x \xleftarrow{\$} \mathcal{X}} [f(z) = y \mid y = f(x),\ z = \mathcal{A}(y)] \leq \text{negl.}$$

Hard-core Predicate for OWF f

- $P : \mathcal{X} \to \{0, 1\}$ is a hard-core predicate for f iff

$$\forall \text{ PPT } \mathcal{A} \qquad \Pr_{x \xleftarrow{\$} \mathcal{X}} [\mathcal{A}(f(x)) = P(x)] \leq \frac{1}{2} + \text{negl.}$$

One-way Functions and Hard-core Predicates

One-way Function

- $f : \mathcal{X} \to \mathcal{Y}$ is a one-way function (OWF) iff
  1. It is easy to compute $f(x)$ given $x \in \mathcal{X}$
  2. It is hard to invert, i.e.,

$$\forall \text{ PPT } \mathcal{A} \qquad \Pr_{x \xleftarrow{\$} \mathcal{X}} [f(z) = y \mid y = f(x),\ z = \mathcal{A}(y)] \leq \text{negl.}$$

Hard-core Predicate for OWF f

- $P : \mathcal{X} \to \{0, 1\}$ is a hard-core predicate for f iff

$$\forall \text{ PPT } \mathcal{A} \qquad \Pr_{x \xleftarrow{\$} \mathcal{X}} [\mathcal{A}(f(x)) = P(x)] \leq \frac{1}{2} + \text{negl.}$$

Why We Need Hard-core Predicates

- $f(x)$ could reveal a lot of partial information about x but not about its hard-core predicates
- Can use hard-core predicates for any application where *pseudo-randomness* is needed
  - Key exchange, encryption, pseudo-random generators, etc.

Why We Need Hard-core Predicates

- $f(x)$ could reveal a lot of partial information about x but not about its hard-core predicates
- Can use hard-core predicates for any application where *pseudo-randomness* is needed
  - Key exchange, encryption, pseudo-random generators, etc.

Known Hard-core Predicates for One-way Functions

Specific Hard-core Predicates

- MSB of DL over $\mathbb{F}_p$ is hard-core - *Blum and Micali (1984)*
- LSB of RSA is hard-core - *Alexi et al. (1988)*
- Each bit of DL modulo Blum integer is hard-core - *Håstad et al. (1993)*
- Every bit of RSA is hard-core - *Håstad and Näslund (1998)*

General Hard-core Predicates

- Every OWF f can be modified to obtain a OWF g having a specific hard-core bit - *Goldreich and Levin (1989)*

Known Hard-core Predicates for One-way Functions

Specific Hard-core Predicates

- MSB of DL over $\mathbb{F}_p$ is hard-core - *Blum and Micali (1984)*
- LSB of RSA is hard-core - *Alexi et al. (1988)*
- Each bit of DL modulo Blum integer is hard-core - *Håstad et al. (1993)*
- Every bit of RSA is hard-core - *Håstad and Näslund (1998)*

General Hard-core Predicates

- Every OWF f can be modified to obtain a OWF g having a specific hard-core bit - *Goldreich and Levin (1989)*

Known Hard-core Predicates for One-way Functions

Specific Hard-core Predicates

- MSB of DL over $\mathbb{F}_p$ is hard-core - *Blum and Micali (1984)*
- LSB of RSA is hard-core - *Alexi et al. (1988)*
- Each bit of DL modulo Blum integer is hard-core - *Håstad et al. (1993)*
- Every bit of RSA is hard-core - *Håstad and Näslund (1998)*

General Hard-core Predicates

- Every OWF f can be modified to obtain a OWF g having a specific hard-core bit - *Goldreich and Levin (1989)*

Known Hard-core Predicates for One-way Functions

Specific Hard-core Predicates

- MSB of DL over $\mathbb{F}_p$ is hard-core - *Blum and Micali (1984)*
- LSB of RSA is hard-core - *Alexi et al. (1988)*
- Each bit of DL modulo Blum integer is hard-core - *Håstad et al. (1993)*
- Every bit of RSA is hard-core - *Håstad and Näslund (1998)*

General Hard-core Predicates

- Every OWF f can be modified to obtain a OWF g having a specific hard-core bit - *Goldreich and Levin (1989)*

Known Hard-core Predicates for One-way Functions

Specific Hard-core Predicates

- MSB of DL over $\mathbb{F}_p$ is hard-core - *Blum and Micali (1984)*
- LSB of RSA is hard-core - *Alexi et al. (1988)*
- Each bit of DL modulo Blum integer is hard-core - *Håstad et al. (1993)*
- Every bit of RSA is hard-core - *Håstad and Näslund (1998)*

General Hard-core Predicates

- Every OWF f can be modified to obtain a OWF g having a specific hard-core bit - *Goldreich and Levin (1989)*

Diffie-Hellman Problem and its Hard-core Predicates

DH Problem

- $\mathbb{G} = \langle g \rangle$ — a group with generator g and order q. DH is hard in $\mathbb{G}$ iff

$$\forall \text{ PPT } \mathcal{A} \qquad \Pr_{a,b \xleftarrow{\$} \mathbb{Z}_q} \left[\mathcal{A}(\mathbb{G}, q, g, g^a, g^b) = g^{ab} \right] \leq \text{negl.}$$

Hard-core Predicate for DH

- $P : \mathbb{G} \to \{0,1\}$ is a hard-core predicate for DH problem iff

$$\forall \text{ PPT } \mathcal{A} \qquad \Pr_{a,b \xleftarrow{\$} \mathbb{Z}_q} \left[\mathcal{A}(\mathbb{G}, q, g, g^a, g^b) = P(g^{ab}) \right] \leq \frac{1}{2} + \text{negl.}$$

- No deterministic hard-core predicate for DH is known
  - But the generic (randomized) Goldreich-Levin result works
- In a modified model LSB of EC-based DH secret value is unpredictable – *Boneh and Shparlinski (2001)*

Diffie-Hellman Problem and its Hard-core Predicates

DH Problem

- $\mathbb{G} = \langle g \rangle$ — a group with generator g and order q. DH is hard in $\mathbb{G}$ iff

$$\forall \text{ PPT } \mathcal{A} \qquad \Pr_{a,b \xleftarrow{\$} \mathbb{Z}_q} \left[\mathcal{A}(\mathbb{G}, q, g, g^a, g^b) = g^{ab} \right] \leq \text{negl.}$$

Hard-core Predicate for DH

- $P : \mathbb{G} \rightarrow \{0,1\}$ is a hard-core predicate for DH problem iff

$$\forall \text{ PPT } \mathcal{A} \qquad \Pr_{a,b \xleftarrow{\$} \mathbb{Z}_q} \left[\mathcal{A}(\mathbb{G}, q, g, g^a, g^b) = P(g^{ab}) \right] \leq \frac{1}{2} + \text{negl.}$$

- No deterministic hard-core predicate for DH is known
  - But the generic (randomized) Goldreich-Levin result works
- In a modified model LSB of EC-based DH secret value is unpredictable - *Boneh and Shparlinski (2001)*

Diffie-Hellman Problem and its Hard-core Predicates

DH Problem

- $\mathbb{G} = \langle g \rangle$ — a group with generator g and order q. DH is hard in $\mathbb{G}$ iff

$$\forall \text{ PPT } \mathcal{A} \qquad \Pr_{a,b \xleftarrow{\$} \mathbb{Z}_q} \left[\mathcal{A}(\mathbb{G}, q, g, g^a, g^b) = g^{ab} \right] \leq \text{negl.}$$

Hard-core Predicate for DH

- $P : \mathbb{G} \to \{0, 1\}$ is a hard-core predicate for DH problem iff

$$\forall \text{ PPT } \mathcal{A} \qquad \Pr_{a,b \xleftarrow{\$} \mathbb{Z}_q} \left[\mathcal{A}(\mathbb{G}, q, g, g^a, g^b) = P(g^{ab}) \right] \leq \frac{1}{2} + \text{negl.}$$

- No deterministic hard-core predicate for DH is known
  - But the generic (randomized) Goldreich-Levin result works
- In a modified model LSB of EC-based DH secret value is unpredictable - *Boneh and Shparlinski (2001)*

Diffie-Hellman Problem and its Hard-core Predicates

DH Problem

- $\mathbb{G} = \langle g \rangle$ — a group with generator g and order q. DH is hard in $\mathbb{G}$ iff

$$\forall \text{ PPT } \mathcal{A} \qquad \Pr_{a,b \xleftarrow{\$} \mathbb{Z}_q}\left[\mathcal{A}(\mathbb{G}, q, g, g^a, g^b) = g^{ab}\right] \leq \text{negl.}$$

Hard-core Predicate for DH

- $P : \mathbb{G} \to \{0, 1\}$ is a hard-core predicate for DH problem iff

$$\forall \text{ PPT } \mathcal{A} \qquad \Pr_{a,b \xleftarrow{\$} \mathbb{Z}_q}\left[\mathcal{A}(\mathbb{G}, q, g, g^a, g^b) = P(g^{ab})\right] \leq \frac{1}{2} + \text{negl.}$$

- No deterministic hard-core predicate for DH is known
  - But the generic (randomized) Goldreich-Levin result works
- In a modified model LSB of EC-based DH secret value is unpredictable - *Boneh and Shparlinski (2001)*

The Result of Boneh and Shparlinski (2001)

Highlights

- EC-based DH is hard $\rightarrow$ LSB of DH secret is hard-core
- Given Ω predicting LSB of DH secret over a random representation of the curve, recover the entire DH secret
- Breakthrough: Use the representation of the curve to randomize the queries to Ω

The Result of Boneh and Shparlinski (2001)

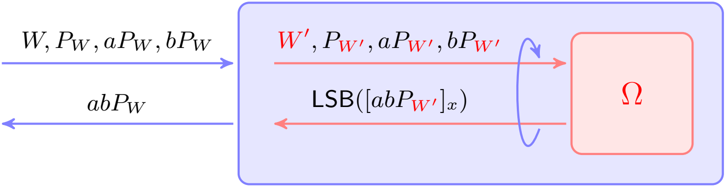
Highlights

- EC-based DH is hard $\rightarrow$ LSB of DH secret is hard-core
- Given Ω predicting LSB of DH secret over a random representation of the curve, recover the entire DH secret
- Breakthrough: Use the representation of the curve to randomize the queries to Ω

The Result of Boneh and Shparlinski (2001)

Highlights

- EC-based DH is hard $\rightarrow$ LSB of DH secret is hard-core
- Given Ω predicting LSB of DH secret over a random representation of the curve, recover the entire DH secret
- Breakthrough: Use the representation of the curve to randomize the queries to Ω



The Result of Duc and Jetchev (2012)

Highlights

- Applies to EC-based OWF (ECB-OWF)
  (i.e., f does not depend on the representation of the curve)
- f is an ECB-OWF $\rightarrow$ every bit of its input is hard-core
- Given Ω predicting any bit of the input to f, invert f
- Main Idea: Apply the Boneh-Shparlinski randomization technique together with the Akavia et al. list-decoding approach.

The Result of Duc and Jetchev (2012)

Highlights

- Applies to EC-based OWF (ECB-OWF)
  (i.e., f does not depend on the representation of the curve)
- f is an ECB-OWF $\rightarrow$ every bit of its input is hard-core
- Given Ω predicting any bit of the input to f, invert f
- Main Idea: Apply the Boneh-Shparlinski randomization technique together with the Akavia et al. list-decoding approach.

The Result of Duc and Jetchev (2012)

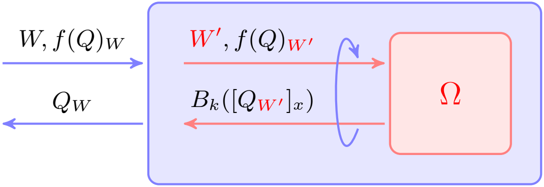
Highlights

- Applies to EC-based OWF (ECB-OWF)
  (i.e., f does not depend on the representation of the curve)
- f is an ECB-OWF $\rightarrow$ every bit of its input is hard-core
- Given Ω predicting any bit of the input to f, invert f
- Main Idea: Apply the Boneh-Shparlinski randomization technique together with the Akavia et al. list-decoding approach.

The Result of Duc and Jetchev (2012)

Highlights

- Applies to EC-based OWF (ECB-OWF) (i.e., f does not depend on the representation of the curve)
- f is an ECB-OWF $\rightarrow$ every bit of its input is hard-core
- Given Ω predicting any bit of the input to f, invert f
- Main Idea: Apply the Boneh-Shparlinski randomization technique together with the Akavia et al. list-decoding approach.



The Result of Akavia et al. (2003)

Highlights

- Let $f : \mathbb{Z}_n \to \mathcal{Y}$ be a OWF, and $\pi : \mathbb{Z}_n \to \{\pm 1\}$ a predicate
- A framework for proving that π is hard-core for f

Approach

1. Define a *multiplication* code

$$\mathcal{C} = \{C_x : \mathbb{Z}_n \to \{\pm 1\} \mid x \in \mathbb{Z}_n\} \quad \text{where} \quad C_x(\lambda) = \pi(\lambda \cdot x)$$

2. Use the oracle that predicts $\pi(x)$ from $f(x)$ to construct a noisy version of C_x
3. Use list-decoding techniques to find a small set of candidates for x

The Result of Akavia et al. (2003)

Highlights

- Let $f : \mathbb{Z}_n \to \mathcal{Y}$ be a OWF, and $\pi : \mathbb{Z}_n \to \{\pm 1\}$ a predicate
- A framework for proving that π is hard-core for f

Approach

1. Define a *multiplication* code

$$\mathcal{C} = \{C_x : \mathbb{Z}_n \to \{\pm 1\} \mid x \in \mathbb{Z}_n\} \quad \text{where} \quad C_x(\lambda) = \pi(\lambda \cdot x)$$

2. Use the oracle that predicts $\pi(x)$ from $f(x)$ to construct a noisy version of C_x
3. Use list-decoding techniques to find a small set of candidates for x

The Result of Akavia et al. (2003)

Highlights

- Let $f : \mathbb{Z}_n \to \mathcal{Y}$ be a OWF, and $\pi : \mathbb{Z}_n \to \{\pm 1\}$ a predicate
- A framework for proving that π is hard-core for f

Approach

1. Define a *multiplication* code

$$\mathcal{C} = \{C_x : \mathbb{Z}_n \to \{\pm 1\} \mid x \in \mathbb{Z}_n\} \quad \text{where} \quad C_x(\lambda) = \pi(\lambda \cdot x)$$

2. Use the oracle that predicts $\pi(x)$ from $f(x)$ to construct a noisy version of C_x
3. Use list-decoding techniques to find a small set of candidates for x

The Result of Akavia et al. (2003)

Highlights

- Let $f : \mathbb{Z}_n \to \mathcal{Y}$ be a OWF, and $\pi : \mathbb{Z}_n \to \{\pm 1\}$ a predicate
- A framework for proving that π is hard-core for f

Approach

1. Define a *multiplication* code

$$\mathcal{C} = \{C_x : \mathbb{Z}_n \to \{\pm 1\} \mid x \in \mathbb{Z}_n\} \quad \text{where} \quad C_x(\lambda) = \pi(\lambda \cdot x)$$

2. Use the oracle that predicts $\pi(x)$ from $f(x)$ to construct a noisy version of C_x
3. Use list-decoding techniques to find a small set of candidates for x

More details on Akavia et al.

$$\mathcal{C} = \{C_x : \mathbb{Z}_n \to \{\pm 1\} \mid x \in \mathbb{Z}_n\} \quad \text{where} \quad C_x(\lambda) = \pi(\lambda \cdot x)$$

It should be shown that $\mathcal{C}$ meets the following properties

Accessible Given $y = f(x)$, it is possible to get a "noisy" $\tilde{C}_x$ of C_x

- They assume f is homomorphic
  i.e., given λ and $f(x)$ it is possible to compute $f(\lambda x)$
- Noisy access to $C_x(\lambda)$ is obtained by querying the oracle on $f(\lambda x)$

Concentrated Every codeword C_x is a Fourier concentrated function

Recoverable Given a frequency (character) χ, $\exists$ a poly time algorithm that finds all values x such that χ is "heavy" for C_x

Fourier-Learnable It is possible to efficiently learn all the heavy coefficients of C_x given query access to its noisy version.

More details on Akavia et al.

$$\mathcal{C} = \{C_x : \mathbb{Z}_n \to \{\pm 1\} \mid x \in \mathbb{Z}_n\} \quad \text{where} \quad C_x(\lambda) = \pi(\lambda \cdot x)$$

It should be shown that $\mathcal{C}$ meets the following properties

Accessible Given $y = f(x)$, it is possible to get a "noisy" $\tilde{C}_x$ of C_x

- They assume f is homomorphic
  i.e., given λ and $f(x)$ it is possible to compute $f(\lambda x)$
- Noisy access to $C_x(\lambda)$ is obtained by querying the oracle on $f(\lambda x)$

Concentrated Every codeword C_x is a Fourier concentrated function

Recoverable Given a frequency (character) χ, $\exists$ a poly time algorithm that finds all values x such that χ is "heavy" for C_x

Fourier-Learnable It is possible to efficiently learn all the heavy coefficients of C_x given query access to its noisy version.

More details on Akavia et al.

$$\mathcal{C} = \{C_x : \mathbb{Z}_n \to \{\pm 1\} \mid x \in \mathbb{Z}_n\} \quad \text{where} \quad C_x(\lambda) = \pi(\lambda \cdot x)$$

It should be shown that $\mathcal{C}$ meets the following properties

Accessible Given $y = f(x)$, it is possible to get a "noisy" $\tilde{C}_x$ of C_x

- They assume f is homomorphic
  i.e., given λ and $f(x)$ it is possible to compute $f(\lambda x)$
- Noisy access to $C_x(\lambda)$ is obtained by querying the oracle on $f(\lambda x)$

Concentrated Every codeword C_x is a Fourier concentrated function

- They prove it for LSB, MSB and segment predicates
  Morillo and Rafols (2008) generalize to any bit

Recoverable Given a frequency (character) χ, $\exists$ a poly time algorithm that finds all values x such that χ is "heavy" for C_x

Fourier-Learnable It is possible to efficiently learn all the heavy coefficients of C_x given query access to its noisy version.

More details on Akavia et al.

$$\mathcal{C} = \{C_x : \mathbb{Z}_n \to \{\pm 1\} \mid x \in \mathbb{Z}_n\} \quad \text{where} \quad C_x(\lambda) = \pi(\lambda \cdot x)$$

It should be shown that $\mathcal{C}$ meets the following properties

Accessible Given $y = f(x)$, it is possible to get a "noisy" $\tilde{C}_x$ of C_x

- They assume f is homomorphic
  i.e., given λ and $f(x)$ it is possible to compute $f(\lambda x)$
- Noisy access to $C_x(\lambda)$ is obtained by querying the oracle on $f(\lambda x)$

Concentrated Every codeword C_x is a Fourier concentrated function

- They prove it for LSB, MSB and segment predicates
- Morillo and Ràfols (2008) generalize to any bit

Recoverable Given a frequency (character) χ, $\exists$ a poly time algorithm that finds all values x such that χ is "heavy" for C_x

Fourier-Learnable It is possible to efficiently learn all the heavy coefficients of C_x given query access to its noisy version.

More details on Akavia et al.

$$\mathcal{C} = \{C_x : \mathbb{Z}_n \to \{\pm 1\} \mid x \in \mathbb{Z}_n\} \quad \text{where} \quad C_x(\lambda) = \pi(\lambda \cdot x)$$

It should be shown that $\mathcal{C}$ meets the following properties

Accessible Given $y = f(x)$, it is possible to get a "noisy" $\tilde{C}_x$ of C_x

- They assume f is homomorphic
  i.e., given λ and $f(x)$ it is possible to compute $f(\lambda x)$
- Noisy access to $C_x(\lambda)$ is obtained by querying the oracle on $f(\lambda x)$

Concentrated Every codeword C_x is a Fourier concentrated function

- They prove it for LSB, MSB and segment predicates
- Morillo and Ràfols (2008) generalize to any bit

Recoverable Given a frequency (character) χ, $\exists$ a poly time algorithm that finds all values x such that χ is "heavy" for C_x

Fourier-Learnable It is possible to efficiently learn all the heavy coefficients of C_x given query access to its noisy version.

More details on Akavia et al.

$$\mathcal{C} = \{C_x : \mathbb{Z}_n \to \{\pm 1\} \mid x \in \mathbb{Z}_n\} \quad \text{where} \quad C_x(\lambda) = \pi(\lambda \cdot x)$$

It should be shown that $\mathcal{C}$ meets the following properties

Accessible Given $y = f(x)$, it is possible to get a "noisy" $\tilde{C}_x$ of C_x

- They assume f is homomorphic
  i.e., given λ and $f(x)$ it is possible to compute $f(\lambda x)$
- Noisy access to $C_x(\lambda)$ is obtained by querying the oracle on $f(\lambda x)$

Concentrated Every codeword C_x is a Fourier concentrated function

- They prove it for LSB, MSB and segment predicates
- Morillo and Ràfols (2008) generalize to any bit

Recoverable Given a frequency (character) χ, $\exists$ a poly time algorithm that finds all values x such that χ is "heavy" for C_x

- Easy consequence of being a multiplication code

Fourier-Learnable It is possible to efficiently learn all the heavy coefficients of C_x given query access to its noisy version.

More details on Akavia et al.

$$\mathcal{C} = \{C_x : \mathbb{Z}_n \to \{\pm 1\} \mid x \in \mathbb{Z}_n\} \quad \text{where} \quad C_x(\lambda) = \pi(\lambda \cdot x)$$

It should be shown that $\mathcal{C}$ meets the following properties

Accessible Given $y = f(x)$, it is possible to get a "noisy" $\tilde{C}_x$ of C_x

- They assume f is homomorphic
  i.e., given λ and $f(x)$ it is possible to compute $f(\lambda x)$
- Noisy access to $C_x(\lambda)$ is obtained by querying the oracle on $f(\lambda x)$

Concentrated Every codeword C_x is a Fourier concentrated function

- They prove it for LSB, MSB and segment predicates
- Morillo and Ràfols (2008) generalize to any bit

Recoverable Given a frequency (character) χ, $\exists$ a poly time algorithm that finds all values x such that χ is "heavy" for C_x

- Easy consequence of being a multiplication code

Fourier-Learnable It is possible to efficiently learn all the heavy coefficients of C_x given query access to its noisy version.

More details on Akavia et al.

$$\mathcal{C} = \{C_x : \mathbb{Z}_n \to \{\pm 1\} \mid x \in \mathbb{Z}_n\} \quad \text{where} \quad C_x(\lambda) = \pi(\lambda \cdot x)$$

It should be shown that $\mathcal{C}$ meets the following properties

Accessible Given $y = f(x)$, it is possible to get a "noisy" $\tilde{C}_x$ of C_x

- They assume f is homomorphic
  i.e., given λ and $f(x)$ it is possible to compute $f(\lambda x)$
- Noisy access to $C_x(\lambda)$ is obtained by querying the oracle on $f(\lambda x)$

Concentrated Every codeword C_x is a Fourier concentrated function

- They prove it for LSB, MSB and segment predicates
- Morillo and Ràfols (2008) generalize to any bit

Recoverable Given a frequency (character) χ, $\exists$ a poly time algorithm that finds all values x such that χ is "heavy" for C_x

- Easy consequence of being a multiplication code

Fourier-Learnable It is possible to efficiently learn all the heavy coefficients of C_x given query access to its noisy version.

More details on Akavia et al.

$$\mathcal{C} = \{C_x : \mathbb{Z}_n \to \{\pm 1\} \mid x \in \mathbb{Z}_n\} \quad \text{where} \quad C_x(\lambda) = \pi(\lambda \cdot x)$$

It should be shown that $\mathcal{C}$ meets the following properties

Accessible Given $y = f(x)$, it is possible to get a "noisy" $\tilde{C}_x$ of C_x

- They assume f is homomorphic
  i.e., given λ and $f(x)$ it is possible to compute $f(\lambda x)$
- Noisy access to $C_x(\lambda)$ is obtained by querying the oracle on $f(\lambda x)$

Concentrated Every codeword C_x is a Fourier concentrated function

- They prove it for LSB, MSB and segment predicates
- Morillo and Ràfols (2008) generalize to any bit

Recoverable Given a frequency (character) χ, $\exists$ a poly time algorithm that finds all values x such that χ is "heavy" for C_x

- Easy consequence of being a multiplication code

Fourier-Learnable It is possible to efficiently learn all the heavy coefficients of C_x given query access to its noisy version.

More details on Akavia et al.

$$\mathcal{C} = \{C_x : \mathbb{Z}_n \to \{\pm 1\} \mid x \in \mathbb{Z}_n\} \quad \text{where} \quad C_x(\lambda) = \pi(\lambda \cdot x)$$

It should be shown that $\mathcal{C}$ meets the following properties

Accessible Given $y = f(x)$, it is possible to get a "noisy" $\tilde{C}_x$ of C_x

- They assume f is homomorphic
  i.e., given λ and $f(x)$ it is possible to compute $f(\lambda x)$
- Noisy access to $C_x(\lambda)$ is obtained by querying the oracle on $f(\lambda x)$

Concentrated Every codeword C_x is a Fourier concentrated function

- They prove it for LSB, MSB and segment predicates
- Morillo and Ràfols (2008) generalize to any bit

Recoverable Given a frequency (character) χ, $\exists$ a poly time algorithm that finds all values x such that χ is "heavy" for C_x

- Easy consequence of being a multiplication code

Fourier-Learnable It is possible to efficiently learn all the heavy coefficients of C_x given query access to its noisy version.

Elliptic Curves, Short Weierstrass Equations, Isomorphisms

Short Weierstrass Equations

- E — an elliptic curve
- $W_{a,b}$ — a short Weierstrass equation representing E

$$W_{a,b} : y^2 = x^3 + ax + b \quad \text{for} \quad a, b \in \mathbb{F}_p,\ 4a^3 + 27b^2 \neq 0$$

Isomorphism Classes

- $W_{a,b}$ is isomorphic to $W'_{a',b'}$ iff $a' = \lambda'^{-4}a$, $b' = \lambda'^{-6}b$ for $\lambda' \in \mathbb{F}_p^\times$
- $\mathcal{W}(E)$ — the isomorphism class of E

$$\mathcal{W}(E) = \{y^2 = x^3 + \lambda^4 ax + \lambda^6 b \mid \lambda \in \mathbb{F}_p^\times\}$$

- $\Phi_\lambda : E \to E$ — the easily computable isomorphism

$$\Phi_\lambda((x, y)) = (\lambda^2 x, \lambda^3 y)$$

Elliptic Curves, Short Weierstrass Equations, Isomorphisms

Short Weierstrass Equations

- E — an elliptic curve
- $W_{a,b}$ — a short Weierstrass equation representing E

$$W_{a,b} : y^2 = x^3 + ax + b \quad \text{for} \quad a, b \in \mathbb{F}_p,\ 4a^3 + 27b^2 \neq 0$$

Isomorphism Classes

- $W_{a,b}$ is isomorphic to $W'_{a',b'}$ iff $a' = \lambda'^{-4}a$, $b' = \lambda'^{-6}b$ for $\lambda' \in \mathbb{F}_p^\times$
- $\mathcal{W}(E)$ — the isomorphism class of E

$$\mathcal{W}(E) = \{y^2 = x^3 + \lambda^4 ax + \lambda^6 b \mid \lambda \in \mathbb{F}_p^\times\}$$

- $\Phi_\lambda : E \to E$ — the easily computable isomorphism

$$\Phi_\lambda((x, y)) = (\lambda^2 x, \lambda^3 y)$$

Elliptic Curves, Short Weierstrass Equations, Isomorphisms

Short Weierstrass Equations

- E — an elliptic curve
- $W_{a,b}$ — a short Weierstrass equation representing E

$$W_{a,b} : y^2 = x^3 + ax + b \quad \text{for} \quad a, b \in \mathbb{F}_p,\ 4a^3 + 27b^2 \neq 0$$

Isomorphism Classes

- $W_{a,b}$ is isomorphic to $W'_{a',b'}$ iff $a' = \lambda'^{-4}a$, $b' = \lambda'^{-6}b$ for $\lambda' \in \mathbb{F}_p^\times$
- $\mathcal{W}(E)$ — the isomorphism class of E

$$\mathcal{W}(E) = \{y^2 = x^3 + \lambda^4 ax + \lambda^6 b \mid \lambda \in \mathbb{F}_p^\times\}$$

- $\Phi_\lambda : E \to E$ — the easily computable isomorphism

$$\Phi_\lambda((x, y)) = (\lambda^2 x, \lambda^3 y)$$

Elliptic Curves, Short Weierstrass Equations, Isomorphisms

Short Weierstrass Equations

- E — an elliptic curve
- $W_{a,b}$ — a short Weierstrass equation representing E

$$W_{a,b} : y^2 = x^3 + ax + b \quad \text{for} \quad a, b \in \mathbb{F}_p,\ 4a^3 + 27b^2 \neq 0$$

Isomorphism Classes

- $W_{a,b}$ is isomorphic to $W'_{a',b'}$ iff $a' = \lambda'^{-4}a$, $b' = \lambda'^{-6}b$ for $\lambda' \in \mathbb{F}_p^\times$
- $\mathcal{W}(E)$ — the isomorphism class of E

$$\mathcal{W}(E) = \left\{ y^2 = x^3 + \lambda^4 ax + \lambda^6 b \mid \lambda \in \mathbb{F}_p^\times \right\}$$

- $\Phi_\lambda : E \to E$ — the easily computable isomorphism

$$\Phi_\lambda((x, y)) = (\lambda^2 x, \lambda^3 y)$$

Our Result 1: Bit-security of Diffie-Hellman over Elliptic Curves

Assumption

- $\mathcal{E}$ — elliptic curve instance generator, E — an elliptic curve generated by $\mathcal{E}$, $\mathbb{G} = \langle P \rangle$ — a cyclic subgroup of E
- DH problem over $\mathcal{E}$ is hard iff

$$\forall \text{ PPT } \mathcal{A} \qquad \Pr_{a,b}\left[\mathcal{A}(E, P, aP, bP) = abP \mid E \leftarrow \mathcal{E}(1^\ell)\right] \leq \text{negl}(\ell)$$

Theorem

- If DH over $\mathcal{E}$ is hard, then

$$\forall \text{ PPT } \Omega \qquad \left| \Pr_{a,b,\lambda}\left[\Omega(\lambda, P, aP, bP) = B_k([\Phi_\lambda(abP)]_x)\right] - \beta_k \right| \leq \text{negl}(\ell)$$

Our Result 1: Bit-security of Diffie-Hellman over Elliptic Curves

Assumption

- $\mathcal{E}$ — elliptic curve instance generator, E — an elliptic curve generated by $\mathcal{E}$, $\mathbb{G} = \langle P \rangle$ — a cyclic subgroup of E
- DH problem over $\mathcal{E}$ is hard iff

$$\forall \text{ PPT } \mathcal{A} \qquad \Pr_{a,b}\left[\mathcal{A}(E, P, aP, bP) = abP \mid E \leftarrow \mathcal{E}(1^{\ell})\right] \leq \mathsf{negl}(\ell)$$

Theorem

- If DH over $\mathcal{E}$ is hard, then

$$\forall \text{ PPT } \Omega \qquad \left|\Pr_{a,b,\lambda}\left[\Omega(\lambda, P, aP, bP) = B_k([\Phi_{\lambda}(abP)]_x)\right] - \beta_k\right| \leq \mathsf{negl}(\ell)$$

Our Result 1: Proof Sketch

What we are given

1. $E,\ P,\ aP,\ bP$
2. Ω predicting $B_k([\Phi_\lambda(abP)]_x) = B_k(\lambda^2[abP]_x)$ with non-negl adv

How we do it

1. Define the multiplication code

$$\mathcal{C} = \{C_Q : \mathbb{F}_p^\times \to \{\pm 1\} \mid Q \in \mathbb{F}_p\} \quad \text{where} \quad C_Q(\lambda) = B_k(\lambda \cdot Q_x)$$

2. But $\Phi_\lambda(\cdot)$ squares λ. So define

$$\Omega'(\lambda, P, aP, bP) = \begin{cases} \Omega(\sqrt{\lambda}, P, aP, bP) & \text{if } \lambda \text{ is a square} \\ \beta_k\text{-biased coin} & \text{otherwise} \end{cases}$$

3. $\mathcal{C}$ meets three properties required for the framework of Akavia et. al.

4. This process gives us a poly-size list of candidates; just output one at random or use Shoup's self-corrector (which outputs the correct one with high probability)

Our Result 1: Proof Sketch

What we are given

1. E, P, aP, bP
2. Ω predicting $B_k([\Phi_\lambda(abP)]_x) = B_k(\lambda^2[abP]_x)$ with non-negl adv

How we do it

1. Define the multiplication code

$$\mathcal{C} = \left\{ C_Q : \mathbb{F}_p^\times \to \{\pm 1\} \mid Q \in \mathbb{F}_p \right\} \quad \text{where} \quad C_Q(\lambda) = B_k(\lambda \cdot Q_x)$$

2. But $\Phi_\lambda(\cdot)$ squares λ. So define

$$\Omega'(\lambda, P, aP, bP) = \begin{cases} \Omega(\sqrt{\lambda}, P, aP, bP) & \text{if } \lambda \text{ is a square} \\ \beta_k\text{-biased coin} & \text{otherwise} \end{cases}$$

3. $\mathcal{C}$ meets three properties required for the framework of Akavia et al.
   - Accessible: Ω' gives us access to a noisy $\tilde{C}_Q = \Omega'(\lambda, P, aP, bP)$
   - Concentrated: Codewords are Fourier concentrated
   - The recovery algorithm of Akavia et al. also works
4. This process gives us a poly-size list of candidates: just output one at random or use Shoup's self-corrector (which outputs the correct one with high probability)

Our Result 1: Proof Sketch

What we are given

1. E, P, aP, bP
2. Ω predicting $B_k([\Phi_\lambda(abP)]_x) = B_k(\lambda^2[abP]_x)$ with non-negl adv

How we do it

1. Define the multiplication code

$$\mathcal{C} = \{C_Q : \mathbb{F}_p^\times \to \{\pm 1\} \mid Q \in \mathbb{F}_p\} \quad \text{where} \quad C_Q(\lambda) = B_k(\lambda \cdot Q_x)$$

2. But $\Phi_\lambda(\cdot)$ squares λ. So define

$$\Omega'(\lambda, P, aP, bP) = \begin{cases} \Omega(\sqrt{\lambda}, P, aP, bP) & \text{if } \lambda \text{ is a square} \\ \beta_k\text{-biased coin} & \text{otherwise} \end{cases}$$

3. $\mathcal{C}$ meets three properties required for the framework of Akavia et al.
   - Accessible Ω' gives us access to a noisy $\tilde{C}_Q = \Omega'(\lambda, P, aP, bP)$
   - Concentrated Codewords are Fourier concentrated
   - Recoverable The recovery algorithm of Akavia et al. also works
4. This process gives us a poly-size list of candidates: just output one at random or use Shoup's self-corrector (which outputs the correct one with high probability)

Our Result 1: Proof Sketch

What we are given

1. E, P, aP, bP
2. Ω predicting $B_k([\Phi_\lambda(abP)]_x) = B_k(\lambda^2[abP]_x)$ with non-negl adv

How we do it

1. Define the multiplication code

$$\mathcal{C} = \{C_Q : \mathbb{F}_p^\times \to \{\pm 1\} \mid Q \in \mathbb{F}_p\} \quad \text{where} \quad C_Q(\lambda) = B_k(\lambda \cdot Q_x)$$

2. But $\Phi_\lambda(\cdot)$ squares λ. So define

$$\Omega'(\lambda, P, aP, bP) = \begin{cases} \Omega(\sqrt{\lambda}, P, aP, bP) & \text{if } \lambda \text{ is a square} \\ \beta_k\text{-biased coin} & \text{otherwise} \end{cases}$$

3. $\mathcal{C}$ meets three properties required for the framework of Akavia et al.
   - **Accessible** Ω' gives us access to a noisy $\tilde{C}_Q = \Omega'(\lambda, P, aP, bP)$
   - **Concentrated** Codewords are Fourier concentrated
   - **Recoverable** The recovery algorithm of Akavia et al. also works
4. This process gives us a poly-size list of candidates: just output one at random or use Shoup's self-corrector (which outputs the correct one with high probability)

Our Result 1: Proof Sketch

What we are given

1. E, P, aP, bP
2. Ω predicting $B_k([\Phi_\lambda(abP)]_x) = B_k(\lambda^2[abP]_x)$ with non-negl adv

How we do it

1. Define the multiplication code

$$\mathcal{C} = \{C_Q : \mathbb{F}_p^\times \to \{\pm 1\} \mid Q \in \mathbb{F}_p\} \quad \text{where} \quad C_Q(\lambda) = B_k(\lambda \cdot Q_x)$$

2. But $\Phi_\lambda(\cdot)$ squares λ. So define

$$\Omega'(\lambda, P, aP, bP) = \begin{cases} \Omega(\sqrt{\lambda}, P, aP, bP) & \text{if } \lambda \text{ is a square} \\ \beta_k\text{-biased coin} & \text{otherwise} \end{cases}$$

3. $\mathcal{C}$ meets three properties required for the framework of Akavia et al.
   - **Accessible** Ω' gives us access to a noisy $\tilde{C}_Q = \Omega'(\lambda, P, aP, bP)$
   - **Concentrated** Codewords are Fourier concentrated
   - **Recoverable** The recovery algorithm of Akavia et al. also works
4. This process gives us a poly-size list of candidates: just output one at random or use Shoup's self-corrector (which outputs the correct one with high probability)

Our Result 1: Proof Sketch

What we are given

1. $E,\ P,\ aP,\ bP$
2. Ω predicting $B_k([\Phi_\lambda(abP)]_x) = B_k(\lambda^2[abP]_x)$ with non-negl adv

How we do it

1. Define the multiplication code

$$\mathcal{C} = \{C_Q : \mathbb{F}_p^\times \to \{\pm 1\} \mid Q \in \mathbb{F}_p\} \quad \text{where} \quad C_Q(\lambda) = B_k(\lambda \cdot Q_x)$$

2. But $\Phi_\lambda(\cdot)$ squares λ. So define

$$\Omega'(\lambda, P, aP, bP) = \begin{cases} \Omega(\sqrt{\lambda}, P, aP, bP) & \text{if } \lambda \text{ is a square} \\ \beta_k\text{-biased coin} & \text{otherwise} \end{cases}$$

3. $\mathcal{C}$ meets three properties required for the framework of Akavia et al.
   - **Accessible** Ω' gives us access to a noisy $\tilde{C}_Q = \Omega'(\lambda, P, aP, bP)$
   - **Concentrated** Codewords are Fourier concentrated
   - **Recoverable** The recovery algorithm of Akavia et al. also works
4. This process gives us a poly-size list of candidates: just output one at random or use Shoup's self-corrector (which outputs the correct one with high probability)

Our Result 1: Proof Sketch

What we are given

1. E, P, aP, bP
2. Ω predicting $B_k([\Phi_\lambda(abP)]_x) = B_k(\lambda^2[abP]_x)$ with non-negl adv

How we do it

1. Define the multiplication code

$$\mathcal{C} = \{C_Q : \mathbb{F}_p^\times \to \{\pm 1\} \mid Q \in \mathbb{F}_p\} \quad \text{where} \quad C_Q(\lambda) = B_k(\lambda \cdot Q_x)$$

2. But $\Phi_\lambda(\cdot)$ squares λ. So define

$$\Omega'(\lambda, P, aP, bP) = \begin{cases} \Omega(\sqrt{\lambda}, P, aP, bP) & \text{if } \lambda \text{ is a square} \\ \beta_k\text{-biased coin} & \text{otherwise} \end{cases}$$

3. $\mathcal{C}$ meets three properties required for the framework of Akavia et al.
   - **Accessible** Ω' gives us access to a noisy $\tilde{C}_Q = \Omega'(\lambda, P, aP, bP)$
   - **Concentrated** Codewords are Fourier concentrated
   - **Recoverable** The recovery algorithm of Akavia et al. also works
4. This process gives us a poly-size list of candidates: just output one at random or use Shoup's self-corrector (which outputs the correct one with high probability)

Other Candidate Settings?

The Finite Field $\mathbb{F}_{p^2}$

- For a given prime p, there are many ($\approx p^2/2$) fields $\mathbb{F}_{p^2}$, and they are all isomorphic to each other
- $h(x) = x^2 + h_1 x + h_0$ — a monic irreducible polynomial of degree 2, $I_2(p)$ — the set of all such polynomials
- $\mathbb{F}_{p^2}$ is isomorphic to $\mathbb{F}_p[x]/(h)$
- We can write elements in $\mathbb{F}_{p^2}$ as linear polynomials
- So for $g \in \mathbb{F}_{p^2}$, denote $g = g_0 + g_1 x$, $[g]_i = g_i$
- Also for $h, \hat{h} \in I_2(p)$ there exists an easily computable isomorphism

$$\phi_{\hat{h}} : \mathbb{F}_p[x]/(h) \to \mathbb{F}_p[x]/(\hat{h}) \quad \text{and} \quad \phi_{\hat{h}} = \begin{bmatrix} 1 & \mu \\ 0 & \lambda \end{bmatrix}$$

- Also given an isomorphism $\phi_{\hat{h}}$ defined by a 2×2 matrix as above, it is easy to find $\hat{h}$
- Note that $[\phi_{\hat{h}}(g)]_1 = \lambda [g]_1$

Other Candidate Settings?

The Finite Field $\mathbb{F}_{p^2}$

- For a given prime p, there are many ($\approx p^2/2$) fields $\mathbb{F}_{p^2}$, and they are all isomorphic to each other
- $h(x) = x^2 + h_1 x + h_0$ — a monic irreducible polynomial of degree 2, $I_2(p)$ — the set of all such polynomials
- $\mathbb{F}_{p^2}$ is isomorphic to $\mathbb{F}_p[x]/(h)$
- We can write elements in $\mathbb{F}_{p^2}$ as linear polynomials
- So for $g \in \mathbb{F}_{p^2}$, denote $g = g_0 + g_1 x$, $[g]_i = g_i$
- Also for $h, \hat{h} \in I_2(p)$ there exists an easily computable isomorphism

$$\phi_{\hat{h}} : \mathbb{F}_p[x]/(h) \to \mathbb{F}_p[x]/(\hat{h}) \quad \text{and} \quad \phi_{\hat{h}} = \begin{bmatrix} 1 & \mu \\ 0 & \lambda \end{bmatrix}$$

- Also given an isomorphism $\phi_{\hat{h}}$ defined by a 2×2 matrix as above, it is easy to find $\hat{h}$
- Note that $[\phi_{\hat{h}}(g)]_1 = \lambda [g]_1$

Other Candidate Settings?

The Finite Field $\mathbb{F}_{p^2}$

- For a given prime p, there are many ($\approx p^2/2$) fields $\mathbb{F}_{p^2}$, and they are all isomorphic to each other
- $h(x) = x^2 + h_1 x + h_0$ — a monic irreducible polynomial of degree 2, $I_2(p)$ — the set of all such polynomials
- $\mathbb{F}_{p^2}$ is isomorphic to $\mathbb{F}_p[x]/(h)$
- We can write elements in $\mathbb{F}_{p^2}$ as linear polynomials
- So for $g \in \mathbb{F}_{p^2}$, denote $g = g_0 + g_1 x$, $[g]_i = g_i$
- Also for $h, \hat{h} \in I_2(p)$ there exists an easily computable isomorphism

$$\phi_{\hat{h}} : \mathbb{F}_p[x]/(h) \to \mathbb{F}_p[x]/(\hat{h}) \quad \text{and} \quad \phi_{\hat{h}} = \begin{bmatrix} 1 & \mu \\ 0 & \lambda \end{bmatrix}$$

- Also given an isomorphism $\phi_{\hat{h}}$ defined by a 2×2 matrix as above, it is easy to find $\hat{h}$
- Note that $[\phi_{\hat{h}}(g)]_1 = \lambda [g]_1$

Other Candidate Settings?

The Finite Field $\mathbb{F}_{p^2}$

- For a given prime p, there are many ($\approx p^2/2$) fields $\mathbb{F}_{p^2}$, and they are all isomorphic to each other
- $h(x) = x^2 + h_1 x + h_0$ — a monic irreducible polynomial of degree 2, $I_2(p)$ — the set of all such polynomials
- $\mathbb{F}_{p^2}$ is isomorphic to $\mathbb{F}_p[x]/(h)$
- We can write elements in $\mathbb{F}_{p^2}$ as linear polynomials
- So for $g \in \mathbb{F}_{p^2}$, denote $g = g_0 + g_1 x$, $[g]_i = g_i$
- Also for $h, \hat{h} \in I_2(p)$ there exists an easily computable isomorphism

$$\phi_{\hat{h}} : \mathbb{F}_p[x]/(h) \to \mathbb{F}_p[x]/(\hat{h}) \quad \text{and} \quad \phi_{\hat{h}} = \begin{bmatrix} 1 & \mu \\ 0 & \lambda \end{bmatrix}$$

- Also given an isomorphism $\phi_{\hat{h}}$ defined by a 2×2 matrix as above, it is easy to find $\hat{h}$
- Note that $[\phi_{\hat{h}}(g)]_1 = \lambda [g]_1$

Other Candidate Settings?

The Finite Field $\mathbb{F}_{p^2}$

- For a given prime p, there are many ($\approx p^2/2$) fields $\mathbb{F}_{p^2}$, and they are all isomorphic to each other
- $h(x) = x^2 + h_1 x + h_0$ — a monic irreducible polynomial of degree 2, $I_2(p)$ — the set of all such polynomials
- $\mathbb{F}_{p^2}$ is isomorphic to $\mathbb{F}_p[x]/(h)$
- We can write elements in $\mathbb{F}_{p^2}$ as linear polynomials
- So for $g \in \mathbb{F}_{p^2}$, denote $g = g_0 + g_1 x$, $[g]_i = g_i$
- Also for $h, \hat{h} \in I_2(p)$ there exists an easily computable isomorphism

$$\phi_{\hat{h}} : \mathbb{F}_p[x]/(h) \to \mathbb{F}_p[x]/(\hat{h}) \quad \text{and} \quad \phi_{\hat{h}} = \begin{bmatrix} 1 & \mu \\ 0 & \lambda \end{bmatrix}$$

- Also given an isomorphism $\phi_{\hat{h}}$ defined by a 2×2 matrix as above, it is easy to find $\hat{h}$
- Note that $[\phi_{\hat{h}}(g)]_1 = \lambda [g]_1$

Other Candidate Settings?

The Finite Field $\mathbb{F}_{p^2}$

- For a given prime p, there are many ($\approx p^2/2$) fields $\mathbb{F}_{p^2}$, and they are all isomorphic to each other
- $h(x) = x^2 + h_1 x + h_0$ — a monic irreducible polynomial of degree 2, $I_2(p)$ — the set of all such polynomials
- $\mathbb{F}_{p^2}$ is isomorphic to $\mathbb{F}_p[x]/(h)$
- We can write elements in $\mathbb{F}_{p^2}$ as linear polynomials
- So for $g \in \mathbb{F}_{p^2}$, denote $g = g_0 + g_1 x$, $[g]_i = g_i$
- Also for $h, \hat{h} \in I_2(p)$ there exists an easily computable isomorphism

$$\phi_{\hat{h}} : \mathbb{F}_p[x]/(h) \to \mathbb{F}_p[x]/(\hat{h}) \quad \text{and} \quad \phi_{\hat{h}} = \begin{bmatrix} 1 & \mu \\ 0 & \lambda \end{bmatrix}$$

- Also given an isomorphism $\phi_{\hat{h}}$ defined by a 2×2 matrix as above, it is easy to find $\hat{h}$
- Note that $[\phi_{\hat{h}}(g)]_1 = \lambda [g]_1$

Other Candidate Settings?

The Finite Field $\mathbb{F}_{p^2}$

- For a given prime p, there are many ($\approx p^2/2$) fields $\mathbb{F}_{p^2}$, and they are all isomorphic to each other
- $h(x) = x^2 + h_1 x + h_0$ — a monic irreducible polynomial of degree 2, $I_2(p)$ — the set of all such polynomials
- $\mathbb{F}_{p^2}$ is isomorphic to $\mathbb{F}_p[x]/(h)$
- We can write elements in $\mathbb{F}_{p^2}$ as linear polynomials
- So for $g \in \mathbb{F}_{p^2}$, denote $g = g_0 + g_1 x$, $[g]_i = g_i$
- Also for $h, \hat{h} \in I_2(p)$ there exists an easily computable isomorphism

$$\phi_{\hat{h}} : \mathbb{F}_p[x]/(h) \to \mathbb{F}_p[x]/(\hat{h}) \quad \text{and} \quad \phi_{\hat{h}} = \begin{bmatrix} 1 & \mu \\ 0 & \lambda \end{bmatrix}$$

- Also given an isomorphism $\phi_{\hat{h}}$ defined by a 2×2 matrix as above, it is easy to find $\hat{h}$
- Note that $[\phi_{\hat{h}}(g)]_1 = \lambda [g]_1$

Other Candidate Settings?

The Finite Field $\mathbb{F}_{p^2}$

- For a given prime p, there are many ($\approx p^2/2$) fields $\mathbb{F}_{p^2}$, and they are all isomorphic to each other
- $h(x) = x^2 + h_1 x + h_0$ — a monic irreducible polynomial of degree 2, $I_2(p)$ — the set of all such polynomials
- $\mathbb{F}_{p^2}$ is isomorphic to $\mathbb{F}_p[x]/(h)$
- We can write elements in $\mathbb{F}_{p^2}$ as linear polynomials
- So for $g \in \mathbb{F}_{p^2}$, denote $g = g_0 + g_1 x$, $[g]_i = g_i$
- Also for $h, \hat{h} \in I_2(p)$ there exists an easily computable isomorphism

$$\phi_{\hat{h}} : \mathbb{F}_p[x]/(h) \to \mathbb{F}_p[x]/(\hat{h}) \quad \text{and} \quad \phi_{\hat{h}} = \begin{bmatrix} 1 & \mu \\ 0 & \lambda \end{bmatrix}$$

- Also given an isomorphism $\phi_{\hat{h}}$ defined by a 2×2 matrix as above, it is easy to find $\hat{h}$
- Note that $[\phi_{\hat{h}}(g)]_1 = \lambda [g]_1$

Our Result 2: Bit-security of (Partial) DH over $\mathbb{F}_{p^2}$

Assumption

- $\mathcal{F}$ — finite field instance generator, F — a finite field generated by $\mathcal{F}$, g — a generator of F
- DH problem over $\mathcal{F}$ is hard iff

$$\forall \text{ PPT } \mathcal{A} \qquad \Pr_{a,b}\left[\mathcal{A}(F, g, g^a, g^b) = \quad g^{ab} \quad \mid F \leftarrow \mathcal{F}(1^\ell)\right] \leq \text{negl}(\ell)$$

Theorem

- If (Partial) DH over $\mathcal{F}$ is hard, then

$$\forall \text{ PPT } \Omega \qquad \left|\Pr_{a,b,\hat{h}}\left[\Omega(\hat{h}, g, g^a, g^b) = B_k\left(\left[\phi_{\hat{h}}\left(g^{ab}\right)\right]_1\right)\right] - \beta_k\right| \leq \text{negl}(\ell)$$

Proof Idea

- Apply the framework of Akavia et al.

Our Result 2: Bit-security of (Partial) DH over $\mathbb{F}_{p^2}$

Assumption

- $\mathcal{F}$ — finite field instance generator, F — a finite field generated by $\mathcal{F}$, g — a generator of F
- DH problem over $\mathcal{F}$ is hard iff

a linear polynomial

$$\forall \text{ PPT } \mathcal{A} \qquad \Pr_{a,b}\left[\mathcal{A}(F, g, g^a, g^b) = \boxed{g^{ab}} \mid F \leftarrow \mathcal{F}(1^\ell)\right] \leq \mathsf{negl}(\ell)$$

Theorem

- If (Partial) DH over $\mathcal{F}$ is hard, then

$$\forall \text{ PPT } \Omega \qquad \left|\Pr_{a,b,\hat{h}}\left[\Omega(\hat{h}, g, g^a, g^b) = B_k\left(\left[\phi_{\hat{h}}\left(g^{ab}\right)\right]_1\right)\right] - \beta_k\right| \leq \mathsf{negl}(\ell)$$

Proof Idea

- Apply the framework of Akavia et al.

Our Result 2: Bit-security of (Partial) DH over $\mathbb{F}_{p^2}$

New Assumption

- $\mathcal{F}$ — finite field instance generator, F — a finite field generated by $\mathcal{F}$, g — a generator of F
- (Partial) DH problem over $\mathcal{F}$ is hard iff

the degree-1 coefficient

$$\forall \text{ PPT } \mathcal{A} \qquad \Pr_{a,b}\left[\mathcal{A}(F, g, g^a, g^b) = \boxed{[g^{ab}]_1} \mid F \leftarrow \mathcal{F}(1^\ell)\right] \leq \mathsf{negl}(\ell)$$

Theorem

- If (Partial) DH over $\mathcal{F}$ is hard, then

$$\forall \text{ PPT } \Omega \qquad \left|\Pr_{a,b,\hat{h}}\left[\Omega(\hat{h}, g, g^a, g^b) = B_k\left(\left[\phi_{\hat{h}}\left(g^{ab}\right)\right]_1\right)\right] - \beta_k\right| \leq \mathsf{negl}(\ell)$$

Proof Idea

- Apply the framework of Akavia et al.

Our Result 2: Bit-security of (Partial) DH over $\mathbb{F}_{p^2}$

New Assumption

- $\mathcal{F}$ — finite field instance generator, F — a finite field generated by $\mathcal{F}$, g — a generator of F
- (Partial) DH problem over $\mathcal{F}$ is hard iff

the degree-1 coefficient

$$\forall \text{ PPT } \mathcal{A} \qquad \Pr_{a,b}\left[\mathcal{A}(F, g, g^a, g^b) = \left[g^{ab}\right]_1 \mid F \leftarrow \mathcal{F}(1^\ell)\right] \leq \mathsf{negl}(\ell)$$

Theorem

- If (Partial) DH over $\mathcal{F}$ is hard, then

$$\forall \text{ PPT } \Omega \qquad \left|\Pr_{a,b,\hat{h}}\left[\Omega(\hat{h}, g, g^a, g^b) = B_k\left(\left[\phi_{\hat{h}}\left(g^{ab}\right)\right]_1\right)\right] - \beta_k\right| \leq \mathsf{negl}(\ell)$$

Proof Idea

- Apply the framework of Akavia et al.

Our Result 2: Bit-security of (Partial) DH over $\mathbb{F}_{p^2}$

New Assumption

- $\mathcal{F}$ — finite field instance generator, F — a finite field generated by $\mathcal{F}$, g — a generator of F
- (Partial) DH problem over $\mathcal{F}$ is hard iff

the degree-1 coefficient

$$\forall \text{ PPT } \mathcal{A} \qquad \Pr_{a,b}\left[\mathcal{A}(F, g, g^a, g^b) = \boxed{[g^{ab}]_1} \mid F \leftarrow \mathcal{F}(1^\ell)\right] \leq \mathsf{negl}(\ell)$$

Theorem

- If (Partial) DH over $\mathcal{F}$ is hard, then

$$\forall \text{ PPT } \Omega \qquad \left|\Pr_{a,b,\hat{h}}\left[\Omega(\hat{h}, g, g^a, g^b) = B_k\left(\left[\phi_{\hat{h}}\left(g^{ab}\right)\right]_1\right)\right] - \beta_k\right| \leq \mathsf{negl}(\ell)$$

Proof Idea

- Apply the framework of Akavia et al.

Our Result 2: Proof Sketch

What we are given

1. $F,\ g,\ g^a,\ g^b$
2. Ω predicting $B_k([\phi_{\hat{h}}(g^{ab})]_1) = B_k(\lambda[g^{ab}]_1)$ with non-negl adv

How we do it

1. Define the multiplication code

$$\mathcal{C} = \{C_\alpha : \mathbb{F}_p^\times \to \{\pm 1\} \mid \alpha \in \mathbb{F}_{p^2}\} \quad \text{where} \quad C_\alpha(\lambda) = B_k(\lambda \cdot [\alpha]_1)$$

2. $\mathcal{C}$ meets three properties required for the framework of Akavia et al.

Our Result 2: Proof Sketch

What we are given

1. F, g, g^a, g^b
2. Ω predicting $B_k([\phi_{\hat{h}}(g^{ab})]_1) = B_k(\lambda[g^{ab}]_1)$ with non-negl adv

How we do it

1. Define the multiplication code

$$\mathcal{C} = \left\{ C_\alpha : \mathbb{F}_p^\times \to \{\pm 1\} \mid \alpha \in \mathbb{F}_{p^2} \right\} \quad \text{where} \quad C_\alpha(\lambda) = B_k(\lambda \cdot [\alpha]_1)$$

2. $\mathcal{C}$ meets three properties required for the framework of Akavia et al.

   Accessible: Ω gives us access to a noisy $\tilde{C}_\alpha = \Omega(\lambda, g, g^a, g^b)$

   Concentrated: Codewords are Fourier concentrated

   The recovery algorithm of Akavia et al. also works

Our Result 2: Proof Sketch

What we are given

1. F, g, g^a, g^b
2. Ω predicting $B_k([\phi_{\hat{h}}(g^{ab})]_1) = B_k(\lambda[g^{ab}]_1)$ with non-negl adv

How we do it

1. Define the multiplication code — a linear polynomial

$$\mathcal{C} = \left\{ C_\alpha : \mathbb{F}_p^\times \to \{\pm 1\} \mid \boxed{\alpha} \in \mathbb{F}_{p^2} \right\} \quad \text{where} \quad C_\alpha(\lambda) = B_k(\lambda \cdot [\alpha]_1)$$

2. $\mathcal{C}$ meets three properties required for the framework of Akavia et al.
   - Accessible: Ω gives us access to a noisy $\tilde{C}_\alpha = \Omega(\lambda, g, g^a, g^b)$
   - Concentrated: Codewords are Fourier concentrated
   - Recoverable: The recovery algorithm of Akavia et al. also works

Our Result 2: Proof Sketch

What we are given

1. $F,\ g,\ g^a,\ g^b$
2. Ω predicting $B_k([\phi_{\hat{h}}(g^{ab})]_1) = B_k(\lambda[g^{ab}]_1)$ with non-negl adv

How we do it

1. Define the multiplication code

   a linear polynomial

   $$\mathcal{C} = \left\{ C_\alpha : \mathbb{F}_p^\times \to \{\pm 1\} \mid \boxed{\alpha} \in \mathbb{F}_{p^2} \right\} \quad \text{where} \quad C_\alpha(\lambda) = B_k(\lambda \cdot [\alpha]_1)$$

2. $\mathcal{C}$ meets three properties required for the framework of Akavia et al.

   - **Accessible** Ω gives us access to a noisy $\tilde{C}_\alpha = \Omega(\lambda, g, g^a, g^b)$
   - **Concentrated** Codewords are Fourier concentrated
   - **Recoverable** The recovery algorithm of Akavia et al. also works

     But, we only get a poly-list of degree-1 coefficients

     So, we pick one coefficient at random (Shoup's self-corrector does not work in this "partial" case)

Our Result 2: Proof Sketch

What we are given

1. $F,\ g,\ g^a,\ g^b$
2. Ω predicting $B_k([\phi_{\hat{h}}(g^{ab})]_1) = B_k(\lambda[g^{ab}]_1)$ with non-negl adv

How we do it

1. Define the multiplication code (a linear polynomial: α)

$$\mathcal{C} = \left\{ C_\alpha : \mathbb{F}_p^\times \to \{\pm 1\} \mid \boxed{\alpha} \in \mathbb{F}_{p^2} \right\} \quad \text{where} \quad C_\alpha(\lambda) = B_k(\lambda \cdot [\alpha]_1)$$

2. $\mathcal{C}$ meets three properties required for the framework of Akavia et al.
   - **Accessible**: Ω gives us access to a noisy $\tilde{C}_\alpha = \Omega(\lambda, g, g^a, g^b)$
   - **Concentrated**: Codewords are Fourier concentrated
   - **Recoverable**: The recovery algorithm of Akavia et al. also works
     But, we only get a poly-list of degree-1 coefficients
     So, we pick one coefficient at random (Shoup's self-corrector does not work in this "partial" case)

Our Result 2: Proof Sketch

What we are given

1. F, g, g^a, g^b
2. Ω predicting $B_k([\phi_{\hat{h}}(g^{ab})]_1) = B_k(\lambda[g^{ab}]_1)$ with non-negl adv

How we do it

1. Define the multiplication code — a linear polynomial

$$\mathcal{C} = \left\{ C_\alpha : \mathbb{F}_p^\times \to \{\pm 1\} \mid \boxed{\alpha} \in \mathbb{F}_{p^2} \right\} \quad \text{where} \quad C_\alpha(\lambda) = B_k(\lambda \cdot [\alpha]_1)$$

2. $\mathcal{C}$ meets three properties required for the framework of Akavia et al.

- **Accessible**: Ω gives us access to a noisy $\tilde{C}_\alpha = \Omega(\lambda, g, g^a, g^b)$
- **Concentrated**: Codewords are Fourier concentrated
- **Recoverable**: The recovery algorithm of Akavia et al. also works
  - But, we only get a poly-list of degree-1 coefficients
  - So, we pick one coefficient at random (Shoup's self-corrector does not work in this "partial" case)

Our Result 2: Proof Sketch

What we are given

1. F, g, g^a, g^b
2. Ω predicting $B_k([\phi_{\hat{h}}(g^{ab})]_1) = B_k(\lambda[g^{ab}]_1)$ with non-negl adv

How we do it

1. Define the multiplication code   a linear polynomial

$$\mathcal{C} = \left\{ C_\alpha : \mathbb{F}_p^\times \to \{\pm 1\} \mid \boxed{\alpha} \in \mathbb{F}_{p^2} \right\} \quad \text{where} \quad C_\alpha(\lambda) = B_k(\lambda \cdot [\alpha]_1)$$

2. $\mathcal{C}$ meets three properties required for the framework of Akavia et al.

Accessible	Ω gives us access to a noisy $\tilde{C}_\alpha = \Omega(\lambda, g, g^a, g^b)$
Concentrated	Codewords are Fourier concentrated
Recoverable	The recovery algorithm of Akavia et al. also works But, we only get a poly-list of degree-1 coefficients So, we pick one coefficient at random (Shoup's self-corrector does not work in this "partial" case)

Our Result 2: Proof Sketch

What we are given

1. F, g, g^a, g^b
2. Ω predicting $B_k([\phi_{\hat{h}}(g^{ab})]_1) = B_k(\lambda[g^{ab}]_1)$ with non-negl adv

How we do it

1. Define the multiplication code   a linear polynomial

$$\mathcal{C} = \left\{ C_\alpha : \mathbb{F}_p^\times \to \{\pm 1\} \mid \boxed{\alpha} \in \mathbb{F}_{p^2} \right\} \quad \text{where} \quad C_\alpha(\lambda) = B_k(\lambda \cdot [\alpha]_1)$$

2. $\mathcal{C}$ meets three properties required for the framework of Akavia et al.

Accessible	Ω gives us access to a noisy $\tilde{C}_\alpha = \Omega(\lambda, g, g^a, g^b)$
Concentrated	Codewords are Fourier concentrated
Recoverable	The recovery algorithm of Akavia et al. also works But, we only get a poly-list of degree-1 coefficients So, we pick one coefficient at random (Shoup's self-corrector does not work in this "partial" case)

Our Result 3: Bit-security of FFB-POWFs

- Our Result 2 also applies to finite field-based partial one-way functions
- f is a FFB-POWF iff
  1. f does not depend on a particular isomorphism class of $\mathbb{F}_{p^2}$
  2. f is easy to compute given α
  3. It is hard to compute $[\alpha]_1$ from $f(\alpha)$

Our Result 3: Bit-security of FFB-POWFs

- Our Result 2 also applies to finite field-based partial one-way functions
- f is a FFB-POWF iff
  1. f does not depend on a particular isomorphism class of $\mathbb{F}_{p^2}$
  2. f is easy to compute given α
  3. It is hard to compute $[\alpha]_1$ from $f(\alpha)$

Our Result 3: Bit-security of FFB-POWFs

- Our Result 2 also applies to finite field-based partial one-way functions
- f is a FFB-POWF iff
  1. f does not depend on a particular isomorphism class of $\mathbb{F}_{p^2}$
  2. f is easy to compute given α
  3. It is hard to compute $[\alpha]_1$ from $f(\alpha)$

Our Result 3: Bit-security of FFB-POWFs

- Our Result 2 also applies to finite field-based partial one-way functions
- f is a FFB-POWF iff
  1. f does not depend on a particular isomorphism class of $\mathbb{F}_{p^2}$
  2. f is easy to compute given α
  3. It is hard to compute $[\alpha]_1$ from $f(\alpha)$

Summary & Open Problems

Summary

1. We proved the unpredictability of every bit of the secret DH value of the of EC DH problem over a random representation of the curve
2. We also extended the above result to (partial) DH problem over finite fields $\mathbb{F}_{p^2}$
3. Our second result also applies to FFB-POWFs over $\mathbb{F}_{p^2}$
4. Our approach "augments" the input to the computationally hard problem with a random description of the underlying group

Open Problems

Summary & Open Problems

Summary

1. We proved the unpredictability of every bit of the secret DH value of the of EC DH problem over a random representation of the curve
2. We also extended the above result to (partial) DH problem over finite fields $\mathbb{F}_{p^2}$
3. Our second result also applies to FFB-POWFs over $\mathbb{F}_{p^2}$
4. Our approach "augments" the input to the computationally hard problem with a random description of the underlying group

Open Problems

1. Extend our results to $\mathbb{F}_{p^k}$ for $k > 2$

Summary & Open Problems

Summary

1. We proved the unpredictability of every bit of the secret DH value of the of EC DH problem over a random representation of the curve
2. We also extended the above result to (partial) DH problem over finite fields $\mathbb{F}_{p^2}$
3. Our second result also applies to FFB-POWFs over $\mathbb{F}_{p^2}$
4. Our approach "augments" the input to the computationally hard problem with a random description of the underlying group

Open Problems

1. Extend our results to $\mathbb{F}_{p^t}$ for $t > 2$
2. Show that DH problem over $\mathbb{F}_{p^2}$ $\rightarrow$ (Partial) DH problem over $\mathbb{F}_{p^2}$

   Show that DH problem over $\mathbb{F}_p$ $\rightarrow$ (Partial) DH problem over $\mathbb{F}_{p^2}$

Summary & Open Problems

Summary

1. We proved the unpredictability of every bit of the secret DH value of the of EC DH problem over a random representation of the curve
2. We also extended the above result to (partial) DH problem over finite fields $\mathbb{F}_{p^2}$
3. Our second result also applies to FFB-POWFs over $\mathbb{F}_{p^2}$
4. Our approach "augments" the input to the computationally hard problem with a random description of the underlying group

Open Problems

1. Extend our results to $\mathbb{F}_{p^t}$ for $t > 2$
2. Show that DH problem over $\mathbb{F}_{p^2} \to$ (Partial) DH problem over $\mathbb{F}_{p^2}$
3. Show that DH problem over $\mathbb{F}_{p} \to$ (Partial) DH problem over $\mathbb{F}_{p^2}$

Summary & Open Problems

Summary

1. We proved the unpredictability of every bit of the secret DH value of the of EC DH problem over a random representation of the curve
2. We also extended the above result to (partial) DH problem over finite fields $\mathbb{F}_{p^2}$
3. Our second result also applies to FFB-POWFs over $\mathbb{F}_{p^2}$
4. Our approach "augments" the input to the computationally hard problem with a random description of the underlying group

Open Problems

1. Extend our results to $\mathbb{F}_{p^t}$ for $t > 2$
2. Show that DH problem over $\mathbb{F}_{p^2} \rightarrow$ (Partial) DH problem over $\mathbb{F}_{p^2}$
3. Show that DH problem over $\mathbb{F}_{p} \rightarrow$ (Partial) DH problem over $\mathbb{F}_{p^2}$

Summary & Open Problems

Summary

1. We proved the unpredictability of every bit of the secret DH value of the of EC DH problem over a random representation of the curve
2. We also extended the above result to (partial) DH problem over finite fields $\mathbb{F}_{p^2}$
3. Our second result also applies to FFB-POWFs over $\mathbb{F}_{p^2}$
4. Our approach "augments" the input to the computationally hard problem with a random description of the underlying group

Open Problems

1. Extend our results to $\mathbb{F}_{p^t}$ for $t > 2$
2. Show that DH problem over $\mathbb{F}_{p^2}$ $\rightarrow$ (Partial) DH problem over $\mathbb{F}_{p^2}$
3. Show that DH problem over $\mathbb{F}_{p}$ $\rightarrow$ (Partial) DH problem over $\mathbb{F}_{p^2}$

Summary & Open Problems

Summary

1. We proved the unpredictability of every bit of the secret DH value of the of EC DH problem over a random representation of the curve
2. We also extended the above result to (partial) DH problem over finite fields $\mathbb{F}_{p^2}$
3. Our second result also applies to FFB-POWFs over $\mathbb{F}_{p^2}$
4. Our approach "augments" the input to the computationally hard problem with a random description of the underlying group

Open Problems

1. Extend our results to $\mathbb{F}_{p^t}$ for $t > 2$
2. Show that DH problem over $\mathbb{F}_{p^2} \rightarrow$ (Partial) DH problem over $\mathbb{F}_{p^2}$
3. Show that DH problem over $\mathbb{F}_p \rightarrow$ (Partial) DH problem over $\mathbb{F}_{p^2}$

Thank You!