On Error Correction in the Exponent

Chris Peikert

MIT Computer Science and AI Laboratory

Theory of Cryptography Conference

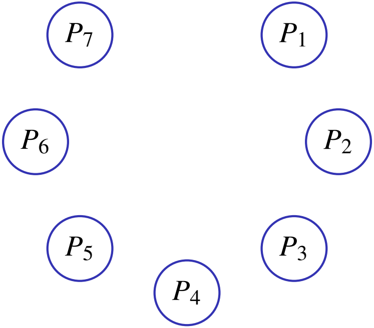
5 March 2006

Error Correction (in the Exponent)

Sharing Secrets (mod q)

- Random $\mathbf{p}(\cdot)$, $\deg(\mathbf{p}) < k$, s.t. $\mathbf{p}(0) =$ secret.
- P_i gets share $x_i = \mathbf{p}(i)$.

$(x_1, \ldots, x_n)$ is Reed-Solomon codewd.



Error Correction (in the Exponent)

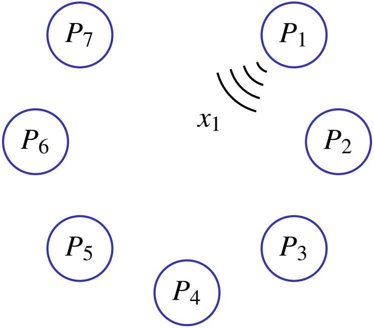
Sharing Secrets (mod q)

- Random $\mathbf{p}(\cdot)$, $\deg(\mathbf{p}) < k$, s.t. $\mathbf{p}(0) =$ secret.
- P_i gets share $x_i = \mathbf{p}(i)$.

$(x_1, \ldots, x_n)$ is Reed-Solomon codewd.

Reconstruction

- P_i announces x_i.



Error Correction (in the Exponent)

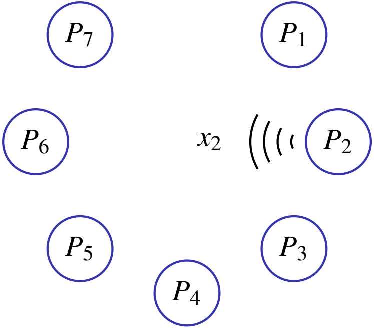
Sharing Secrets (mod q)

- Random $\mathbf{p}(\cdot)$, $\deg(\mathbf{p}) < k$, s.t. $\mathbf{p}(0) =$ secret.
- P_i gets share $x_i = \mathbf{p}(i)$.

$(x_1, \ldots, x_n)$ is Reed-Solomon codewd.

Reconstruction

- P_i announces x_i.



Sharing Secrets (mod q)

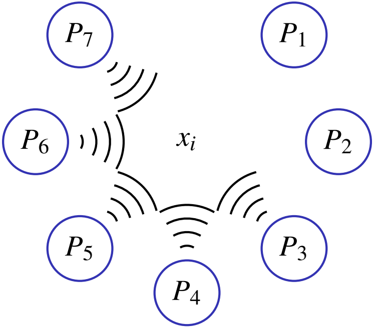
- Random $\mathbf{p}(\cdot)$, $\deg(\mathbf{p}) < k$, s.t. $\mathbf{p}(0) =$ secret.
- P_i gets share $x_i = \mathbf{p}(i)$.

$(x_1, \ldots, x_n)$ is Reed-Solomon codewd.

Reconstruction

- P_i announces x_i.

Interpolation: $\mathbf{p}(\alpha) = \sum x_i \lambda_i$ for any α.



Error Correction (in the Exponent)

Sharing Secrets (mod q)

- Random $\mathbf{p}(\cdot), \quad \deg(\mathbf{p}) < k$, s.t. $\mathbf{p}(0) =$ secret.
- P_i gets share $x_i = \mathbf{p}(i)$.

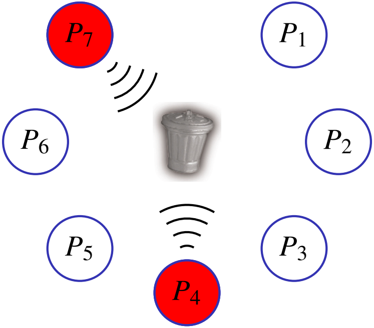
$(x_1, \ldots, x_n)$ is Reed-Solomon codewd.

Reconstruction

- P_i announces x_i.

Interpolation: $\mathbf{p}(\alpha) = \sum x_i \lambda_i$ for any α.

Error correction: [BeWe86, GuSu98]



Error Correction (in the Exponent)

Sharing Secrets (mod q)

- Random $\mathbf{p}(\cdot), \quad \deg(\mathbf{p}) < k$, s.t. $\mathbf{p}(0) = \text{secret}$.
- P_i gets share $x_i = \mathbf{p}(i)$.

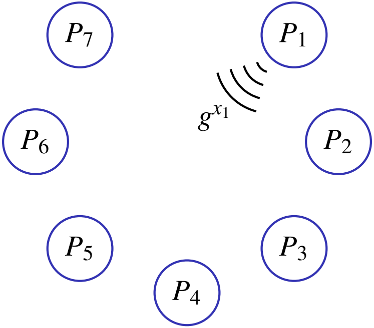
$(x_1, \ldots, x_n)$ is Reed-Solomon codewd.

Placing Shares "in the Exponent"

[CJKR96, PK96, RG03, NPR99, D03, CD04, CG99, BF99,...]

Cyclic group $G = \langle g \rangle$, order q

- P_i announces g^{x_i}.



Error Correction (in the Exponent)

Sharing Secrets (mod q)

- Random $\mathbf{p}(\cdot)$, $\deg(\mathbf{p}) < k$, s.t. $\mathbf{p}(0) =$ secret.
- P_i gets share $x_i = \mathbf{p}(i)$.

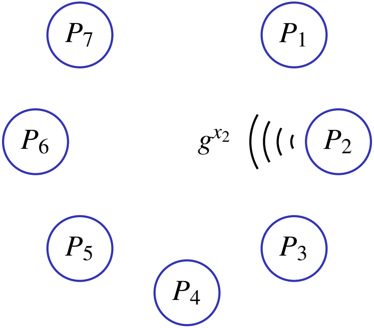
$(x_1, \ldots, x_n)$ is Reed-Solomon codewd.

Placing Shares "in the Exponent"

[CJKR96, PK96, RG03, NPR99, D03, CD04, CG99, BF99,...]

Cyclic group $G = \langle g \rangle$, order q

- P_i announces g^{x_i}.



Error Correction (in the Exponent)

Sharing Secrets (mod q)

- Random $\mathbf{p}(\cdot)$, $\deg(\mathbf{p}) < k$, s.t. $\mathbf{p}(0) =$ secret.
- P_i gets share $x_i = \mathbf{p}(i)$.

$(x_1, \ldots, x_n)$ is Reed-Solomon codewd.

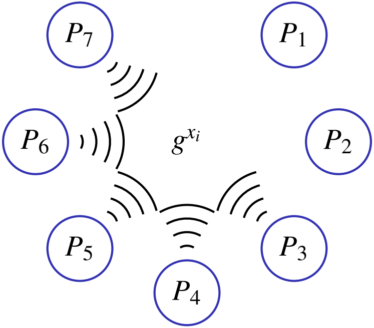
Placing Shares "in the Exponent"

[CJKR96, PK96, RG03, NPR99, D03, CD04, CG99, BF99,. . .]

Cyclic group $G = \langle g \rangle$, order q

- P_i announces g^{x_i}.

Interpolation: $g^{\mathbf{p}(\alpha)} = \prod (g^{x_i})^{\lambda_i}$



Error Correction (in the Exponent)

Sharing Secrets (mod q)

- Random $\mathbf{p}(\cdot), \quad \deg(\mathbf{p}) < k$, s.t. $\mathbf{p}(0) =$ secret.
- P_i gets share $x_i = \mathbf{p}(i)$.

$(x_1, \ldots, x_n)$ is Reed-Solomon codewd.

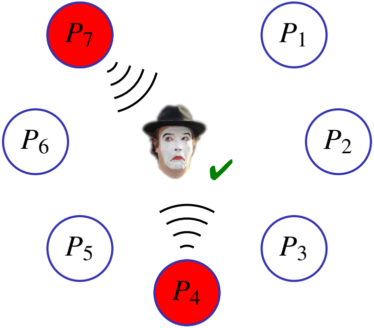
Placing Shares "in the Exponent"

[CJKR96, PK96, RG03, NPR99, D03, CD04, CG99, BF99,. . .]

Cyclic group $G = \langle g \rangle$, order q

- P_i announces g^{x_i}.

Interpolation: $\quad g^{\mathbf{p}(\alpha)} = \prod (g^{x_i})^{\lambda_i}$



Error Correction (in the Exponent)

Sharing Secrets (mod q)

- Random $\mathbf{p}(\cdot)$, $\deg(\mathbf{p}) < k$, s.t. $\mathbf{p}(0) =$ secret.
- P_i gets share $x_i = \mathbf{p}(i)$.

$(x_1, \ldots, x_n)$ is Reed-Solomon codewd.

Placing Shares "in the Exponent"

[CJKR96, PK96, RG03, NPR99, D03, CD04, CG99, BF99,...]

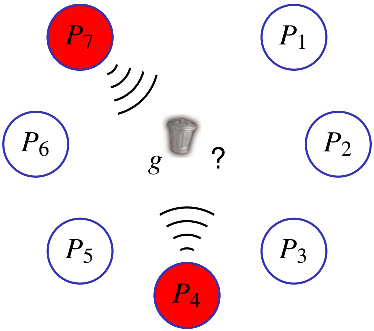
Cyclic group $G = \langle g \rangle$, order q

- P_i announces g^{x_i}.

Interpolation: $g^{\mathbf{p}(\alpha)} = \prod (g^{x_i})^{\lambda_i}$

ERROR CORRECTION: ???

- Guess-and-check: $\frac{n \log n}{k}$ errors



Our Contributions

☞ The first detailed study of the complexity of ECE.

Our Contributions

☞ The first detailed study of the complexity of ECE.

Unconditional Results

Errors	Complexity
$n - \sqrt{nk}$	EASY AS DH
$n - k - k^{1-\epsilon}$	HARD AS DLOG

Our Contributions

☞ The first detailed study of the complexity of ECE.

Unconditional Results

Gap $\approx \delta \cdot k$	Errors	Complexity
→	$n - \sqrt{nk}$	EASY AS DH
→	$n - k - k^{1-\epsilon}$	HARD AS DLOG

Our Contributions

☞ The first detailed study of the complexity of ECE.

Unconditional Results

Gap	Errors	Complexity	
$\approx \delta \cdot k$	$n - \sqrt{nk}$	EASY AS DH	link DH to DLOG?
	$n - k - k^{1-\epsilon}$	HARD AS DLOG	

Our Contributions

☞ The first detailed study of the complexity of ECE.

Unconditional Results

	Errors	Complexity	
Gap $\approx \delta \cdot k$	$n - \sqrt{nk}$	EASY AS DH	link DH to DLOG?
	$n - k - k^{1-\epsilon}$	HARD AS DLOG	

Results for Generic Algorithms

- Guess-and-check is optimal — even if DDH is easy.

Our Contributions

☞ The first detailed study of the complexity of ECE.

Unconditional Results

	Errors	Complexity	
Gap $\approx \delta \cdot k$	$n - \sqrt{nk}$	EASY AS DH	link DH to DLOG?
	$n - k - k^{1-\epsilon}$	HARD AS DLOG	

Results for Generic Algorithms

- Guess-and-check is optimal — even if DDH is easy.

Evidence for:

$$\text{DDH} < \text{ECE} \leq \text{DH}$$

A new approach for:

$$\text{DDH} < \text{ECE} = \text{DLOG} = \text{DH}$$

Relation to Discrete Log

Theorem

Decoding (in the exponent) to distance $n - k - k^{1-\epsilon}$ is as hard as computing discrete logs in G.

Relation to Discrete Log

Theorem

Decoding (in the exponent) to distance $n - k - k^{1-\epsilon}$ is as hard as computing discrete logs in G.

Proof Sketch

1. Finding a representation on uniform $\mathbf{w} \in G^n$ is as hard as dlog.
2. Uniform $\mathbf{w}$ is close (in the exponent) to some codeword.
3. Decoding $\mathbf{w}$ yields a representation on $\mathbf{w}$.

Relation to Discrete Log

Theorem

Decoding (in the exponent) to distance $n - k - k^{1-\epsilon}$
is as hard as computing discrete logs in G.

Proof Sketch

1. Finding a representation on uniform $\mathbf{w} \in G^n$ is as hard as dlog.
2. Uniform $\mathbf{w}$ is close (in the exponent) to some codeword.
3. Decoding $\mathbf{w}$ yields a representation on $\mathbf{w}$.

- Representation on $\mathbf{w}$: nonzero $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{Z}_q^n$ s.t.

$$\prod_i w_i^{a_i} = 1.$$

- [Bra93] showed hardness.

Relation to Discrete Log

Theorem

Decoding (in the exponent) to distance $n - k - k^{1-\epsilon}$ is as hard as computing discrete logs in G.

Proof Sketch

1. Finding a representation on uniform $\mathbf{w} \in G^n$ is as hard as dlog.
2. Uniform $\mathbf{w}$ is close (in the exponent) to some codeword.
3. Decoding $\mathbf{w}$ yields a representation on $\mathbf{w}$.

We show $\exists\ \ell = k + k^{1-\epsilon}$ points $w_i = g^{x_i}$, with x_i on poly of deg $< k$.

- There are $\binom{n}{\ell}$ distinct events (each very rare).
- These events have limited dependence.

Relation to Discrete Log

Theorem

Decoding (in the exponent) to distance $n - k - k^{1-\epsilon}$ is as hard as computing discrete logs in G.

Proof Sketch

1. Finding a representation on uniform $\mathbf{w} \in G^n$ is as hard as dlog.
2. Uniform $\mathbf{w}$ is close (in the exponent) to some codeword.
3. Decoding $\mathbf{w}$ yields a representation on $\mathbf{w}$.

We show $\exists\ \ell = k + k^{1-\epsilon}$ points $w_i = g^{x_i}$, with x_i on poly of deg $< k$.

- There are $\binom{n}{\ell}$ distinct events (each very rare).
- These events have limited dependence.

Relation to Discrete Log

Theorem

Decoding (in the exponent) to distance $n - k - k^{1-\epsilon}$ is as hard as computing discrete logs in G.

Proof Sketch

1. Finding a representation on uniform $\mathbf{w} \in G^n$ is as hard as dlog.
2. Uniform $\mathbf{w}$ is close (in the exponent) to some codeword.
3. Decoding $\mathbf{w}$ yields a representation on $\mathbf{w}$.
   - Decode $\mathbf{w}$ to $(g^{x_1}, \ldots, g^{x_n})$, where x_i lie on poly of deg $< k$.
   - There are $\gg k$ points $w_i = g^{x_i}$. wlog: $w_1, \ldots, w_{k+1}$.
   - Interpolate in the exponent:

$$w_{k+1} = \prod_{i=1}^{k} w_i^{\lambda_i} \quad \Rightarrow \quad \text{representation!}$$

Generic Algorithms [Sho97]

Intuition

Treat group as "black-box" — don't use element *representations*

Generic Algorithms [Sho97]

Intuition

Treat group as "black-box" — don't use element *representations*

Formalization

- Random encoding $\sigma : G \to \{0,1\}^*$
- Oracle for group operation [wlog $G = (\mathbb{Z}_q, +)$]

Intuition

Treat group as “black-box” — don’t use element *representations*

Formalization

- Random encoding $\sigma : G \to \{0,1\}^*$
- Oracle for group operation [wlog $G = (\mathbb{Z}_q, +)$]

$\sigma(x_0)$	$\sigma(x_1)$				

$\cdots$

Alg

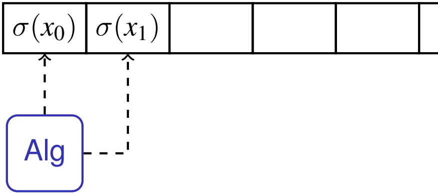
Generic Algorithms [Sho97]

Intuition

Treat group as "black-box" — don't use element *representations*

Formalization

- Random encoding $\sigma : G \to \{0,1\}^*$
- Oracle for group operation [wlog $G = (\mathbb{Z}_q, +)$]



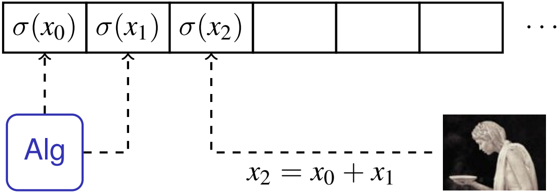
Generic Algorithms [Sho97]

Intuition

Treat group as "black-box" — don't use element *representations*

Formalization

- Random encoding $\sigma : G \to \{0,1\}^*$
- Oracle for group operation [wlog $G = (\mathbb{Z}_q, +)$]



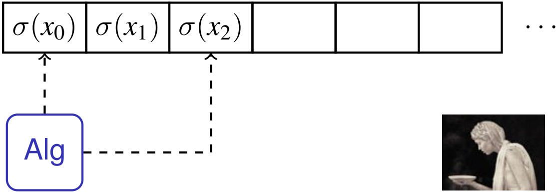
Generic Algorithms [Sho97]

Intuition

Treat group as "black-box" — don't use element *representations*

Formalization

- Random encoding $\sigma : G \to \{0,1\}^*$
- Oracle for group operation   [wlog $G = (\mathbb{Z}_q, +)$]



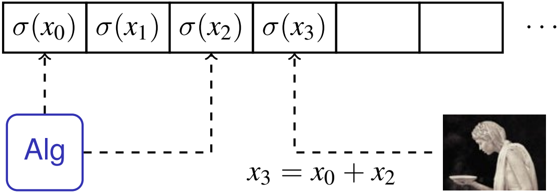
Generic Algorithms [Sho97]

Intuition

Treat group as “black-box” — don’t use element *representations*

Formalization

- Random encoding $\sigma : G \to \{0,1\}^*$
- Oracle for group operation [wlog $G = (\mathbb{Z}_q, +)$]



Generic Algorithms [Sho97]

Intuition

Treat group as "black-box" — don't use element *representations*

Formalization

- Random encoding $\sigma : G \to \{0,1\}^*$
- Oracle for group operation [wlog $G = (\mathbb{Z}_q, +)$]

$\sigma(x_0)$	$\sigma(x_1)$	$\sigma(x_2)$	$\sigma(x_3)$	$\sigma(x_4)$		$\cdots$

Alg

Intuition

Treat group as "black-box" — don't use element *representations*

Formalization

- Random encoding $\sigma : G \to \{0,1\}^*$
- Oracle for group operation [wlog $G = (\mathbb{Z}_q, +)$]

$\sigma(x_0)$	$\sigma(x_1)$	$\sigma(x_2)$	$\sigma(x_3)$	$\sigma(x_4)$	$\sigma(x_5)$	$\cdots$

Alg

Generic Algorithms [Sho97]

Intuition

Treat group as "black-box" — don't use element *representations*

Formalization

- Random encoding $\sigma : G \to \{0,1\}^*$
- Oracle for group operation [wlog $G = (\mathbb{Z}_q, +)$]

$\sigma(x_0)$	$\sigma(x_1)$	$\sigma(x_2)$	$\sigma(x_3)$	$\sigma(x_4)$	$\sigma(x_5)$	$\cdots$

Alg - - - - → 0/1, $\sigma(x)$, ...

Generic Interpolation with Errors

- ☞ Interpolation w/ Errors: $(\mathbf{p}(1), \ldots, \mathbf{p}(n)) + \mathbf{e} \quad \longmapsto \quad \mathbf{p}(0)$

Generic Interpolation with Errors

☞ Interpolation w/ Errors: $(\mathbf{p}(1), \ldots, \mathbf{p}(n)) + \mathbf{e} \longmapsto \mathbf{p}(0)$

Theorem

Interpolation under $\gg \frac{n \log n}{k}$ errors is hard for generic algorithms.

Generic Interpolation with Errors

☞ Interpolation w/ Errors: $(\mathbf{p}(1), \ldots, \mathbf{p}(n)) + \mathbf{e} \quad \longmapsto \quad \mathbf{p}(0)$

Theorem

Interpolation under $\gg \frac{n \log n}{k}$ *errors*
is hard for generic algorithms.

Guess-and-check is optimal!

Generic Interpolation with Errors

☞ Interpolation w/ Errors: $(\mathbf{p}(1), \ldots, \mathbf{p}(n)) + \mathbf{e} \longmapsto \mathbf{p}(0)$

Theorem

Interpolation under $\gg \frac{n \log n}{k}$ errors is hard for generic algorithms.

Ideal Game

- Leave $\mathbf{p}$ and $\mathbf{e}$ as indeterminants; encode polynomials $F(\mathbf{p}, \mathbf{e})$

$\sigma(F_1)$	$\cdots$	$\sigma(F_n)$				$\cdots$

Alg

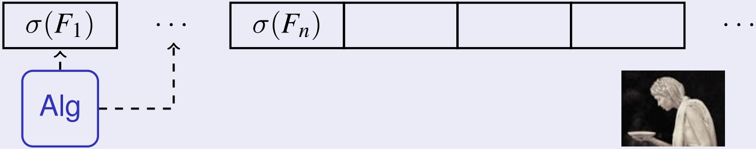
☞ Interpolation w/ Errors: $(\mathbf{p}(1), \ldots, \mathbf{p}(n)) + \mathbf{e} \longmapsto \mathbf{p}(0)$

Theorem

Interpolation under $\gg \frac{n \log n}{k}$ errors
is hard for generic algorithms.

Ideal Game

- Leave $\mathbf{p}$ and $\mathbf{e}$ as indeterminants; encode polynomials $F(\mathbf{p}, \mathbf{e})$



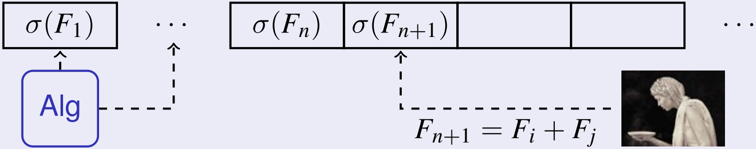
☞ Interpolation w/ Errors: $(\mathbf{p}(1), \ldots, \mathbf{p}(n)) + \mathbf{e} \quad \longmapsto \quad \mathbf{p}(0)$

Theorem

Interpolation under $\gg \frac{n \log n}{k}$ errors
is hard for generic algorithms.

Ideal Game

- Leave $\mathbf{p}$ and $\mathbf{e}$ as indeterminants; encode polynomials $F(\mathbf{p}, \mathbf{e})$



Generic Interpolation with Errors

☞ Interpolation w/ Errors: $(\mathbf{p}(1), \ldots, \mathbf{p}(n)) + \mathbf{e} \longmapsto \mathbf{p}(0)$

Theorem

Interpolation under $\gg \frac{n \log n}{k}$ errors is hard for generic algorithms.

Ideal Game

- Leave $\mathbf{p}$ and $\mathbf{e}$ as indeterminants; encode polynomials $F(\mathbf{p}, \mathbf{e})$

$\sigma(F_1)$ $\cdots$ $\sigma(F_n)$ | $\sigma(F_{n+1})$ | $\sigma(F_{n+2})$ | $\cdots$

Alg

Generic Interpolation with Errors

☞ Interpolation w/ Errors: $(\mathbf{p}(1), \ldots, \mathbf{p}(n)) + \mathbf{e} \quad \longmapsto \quad \mathbf{p}(0)$

Theorem

Interpolation under $\gg \frac{n \log n}{k}$ *errors*
is hard for generic algorithms.

Ideal Game

- Leave $\mathbf{p}$ and $\mathbf{e}$ as indeterminants; encode polynomials $F(\mathbf{p}, \mathbf{e})$

$\sigma(F_1)$ $\cdots$ $\sigma(F_n)$ $\sigma(F_{n+1})$ $\sigma(F_{n+2})$ $\sigma(F_{n+3})$ $\cdots$

Alg

Generic Interpolation with Errors

☞ Interpolation w/ Errors: $(\mathbf{p}(1), \ldots, \mathbf{p}(n)) + \mathbf{e} \longmapsto \mathbf{p}(0)$

Theorem

Interpolation under $\gg \frac{n \log n}{k}$ errors is hard for generic algorithms.

Ideal Game

- Leave $\mathbf{p}$ and $\mathbf{e}$ as indeterminants; encode polynomials $F(\mathbf{p}, \mathbf{e})$

$$\boxed{\sigma(F_1)} \quad \cdots \quad \boxed{\sigma(F_n)}\boxed{\sigma(F_{n+1})}\boxed{\sigma(F_{n+2})}\boxed{\sigma(F_{n+3})} \quad \cdots$$

Alg $\dashrightarrow \neq \sigma(F_0)$

Generic Interpolation with Errors

☞ Interpolation w/ Errors: $(\mathbf{p}(1), \ldots, \mathbf{p}(n)) + \mathbf{e} \longmapsto \mathbf{p}(0)$

Theorem

Interpolation under $\gg \frac{n \log n}{k}$ errors is hard for generic algorithms.

Ideal Game

- Leave $\mathbf{p}$ and $\mathbf{e}$ as indeterminants; encode polynomials $F(\mathbf{p}, \mathbf{e})$

$\sigma(F_1)$ $\cdots$ $\sigma(F_n)$ | $\sigma(F_{n+1})$ | $\sigma(F_{n+2})$ | $\sigma(F_{n+3})$ $\cdots$

Alg $\dashrightarrow \neq \sigma(F_0)$

- Differs from real game only if $\exists\, F_i \not\equiv F_j$, but $(F_i - F_j)(\mathbf{p}, \mathbf{e}) = 0$. Analyze event for "strange" distribution of $\mathbf{p}, \mathbf{e}$.

Analysis of Ideal Game

To Show

For all $F = F_i - F_j \not\equiv 0$, $\Pr[F(\mathbf{p}, \mathbf{e}) = 0]$ is small.

Analysis of Ideal Game

To Show

For all $F = F_i - F_j \not\equiv 0$, $\Pr[F(\mathbf{p}, \mathbf{e}) = 0]$ is small.

Sketch

1. F is linear in $\mathbf{p}, \mathbf{e}$ (because inputs $F_1, \ldots, F_n$ are).

Analysis of Ideal Game

To Show

For all $F = F_i - F_j \not\equiv 0$, $\Pr[F(\mathbf{p}, \mathbf{e}) = 0]$ is small.

Sketch

1. F is linear in $\mathbf{p}, \mathbf{e}$ (because inputs $F_1, \ldots, F_n$ are).
2. e variables in $\mathbf{e}$ are uniform (others are zero).

Analysis of Ideal Game

To Show

For all $F = F_i - F_j \not\equiv 0$, $\Pr[F(\mathbf{p}, \mathbf{e}) = 0]$ is small.

Sketch

1. F is linear in $\mathbf{p}, \mathbf{e}$ (because inputs $F_1, \ldots, F_n$ are).
2. e variables in $\mathbf{e}$ are uniform (others are zero).
3. F depends on *some* uniform variable (either in $\mathbf{p}$ or $\mathbf{e}$).

Analysis of Ideal Game

To Show

For all $F = F_i - F_j \not\equiv 0$, $\Pr[F(\mathbf{p}, \mathbf{e}) = 0]$ is small.

Sketch

1. F is linear in $\mathbf{p}, \mathbf{e}$ (because inputs $F_1, \ldots, F_n$ are).
2. e variables in $\mathbf{e}$ are uniform (others are zero).
3. F depends on *some* uniform variable (either in $\mathbf{p}$ or $\mathbf{e}$).

Suppose F doesn't depend on any variables in $\mathbf{p}$.

Analysis of Ideal Game

To Show

For all $F = F_i - F_j \not\equiv 0$, $\Pr[F(\mathbf{p}, \mathbf{e}) = 0]$ is small.

Sketch

1. F is linear in $\mathbf{p}, \mathbf{e}$ (because inputs $F_1, \ldots, F_n$ are).
2. e variables in $\mathbf{e}$ are uniform (others are zero).
3. F depends on *some* uniform variable (either in $\mathbf{p}$ or $\mathbf{e}$).

   Suppose F doesn't depend on any variables in $\mathbf{p}$.

   Then F depends on $\geq n - k$ positions of $\mathbf{e}$.
   (Dual of Reed-Solomon code.)

Analysis of Ideal Game

To Show

For all $F = F_i - F_j \not\equiv 0$, $\Pr[F(\mathbf{p}, \mathbf{e}) = 0]$ is small.

Sketch

1. F is linear in $\mathbf{p}, \mathbf{e}$ (because inputs $F_1, \ldots, F_n$ are).
2. e variables in $\mathbf{e}$ are uniform (others are zero).
3. F depends on *some* uniform variable (either in $\mathbf{p}$ or $\mathbf{e}$).

   Suppose F doesn't depend on any variables in $\mathbf{p}$.

   Then F depends on $\geq n - k$ positions of $\mathbf{e}$.
   (Dual of Reed-Solomon code.)

   With overwhelming prob, F depends on some uniform e_i.

Analysis of Ideal Game

To Show

For all $F = F_i - F_j \not\equiv 0$, $\Pr[F(\mathbf{p}, \mathbf{e}) = 0]$ is small.

Sketch

1. F is linear in $\mathbf{p}, \mathbf{e}$ (because inputs $F_1, \ldots, F_n$ are).
2. e variables in $\mathbf{e}$ are uniform (others are zero).
3. F depends on *some* uniform variable (either in $\mathbf{p}$ or $\mathbf{e}$).

   Suppose F doesn't depend on any variables in $\mathbf{p}$.

   Then F depends on $\geq n - k$ positions of $\mathbf{e}$.
   (Dual of Reed-Solomon code.)

   With overwhelming prob, F depends on some uniform e_i.
4. By Schwartz's Lemma, $\Pr[F(\mathbf{p}, \mathbf{e}) = 0]$ small.

Relation to Decisional DH

Question

- Recall: error correction is easy, given DH oracle.
- What about DDH?

Relation to Decisional DH

Question

- Recall: error correction is easy, given DH oracle.
- What about DDH?

Our Proposal

Augment generic algorithms with a DDH oracle.

Models “gap” groups: DDH is easy, but DH believed hard.

Relation to Decisional DH

Question

- Recall: error correction is easy, given DH oracle.
- What about DDH?

Our Proposal

Augment generic algorithms with a DDH oracle.

Models "gap" groups: DDH is easy, but DH believed hard.

Theorem

For $e \cdot k = \omega(n \log n)$, there is no efficient DDH-augmented generic algorithm for interpolating noisy polynomials.

Relation to Decisional DH

Question

- Recall: error correction is easy, given DH oracle.
- What about DDH?

Our Proposal

Augment generic algorithms with a DDH oracle.

Models "gap" groups: DDH is easy, but DH believed hard.

Theorem

For $e \cdot k = \omega(n \log n)$, there is
no efficient DDH-augmented generic algorithm
for interpolating noisy polynomials.

☞ Converse does not appear to hold.
I.e., error correction seems strictly harder than DDH.

Conclusions and Open Problems

Conclusions

- Characterized hardness of ECE for a spectrum of errors.
- Given evidence for DDH $<$ ECE.
- Suggested a new approach for linking DH and DLOG.

Conclusions and Open Problems

Conclusions

- Characterized hardness of ECE for a spectrum of errors.
- Given evidence for DDH $<$ ECE.
- Suggested a new approach for linking DH and DLOG.

Questions

- Construct crypto schemes based on hardness of ECE?
- Tighten gap between # errors for DLOG and DH reductions?
- Non-generic ECE algorithms (index calculus)?

Conclusions and Open Problems

Conclusions

- Characterized hardness of ECE for a spectrum of errors.
- Given evidence for DDH $<$ ECE.
- Suggested a new approach for linking DH and DLOG.

Questions

- Construct crypto schemes based on hardness of ECE?
- Tighten gap between # errors for DLOG and DH reductions?
- Non-generic ECE algorithms (index calculus)?

Thank you!