Selected Areas in Cryptography 2026
University of Ottawa · 27 August 2026 · Code-based Cryptography

Bilinear Hulls for Support Splitting: Characterization, Optimization, and Applications

Yusuke Aikawa[1] **Keita Ishizuka**[2] **Tomoki Moriya**[2]

[1] The University of Tokyo [2] Mitsubishi Electric Corporation

Speaker: Keita Ishizuka

PEP and the Support Splitting Algorithm

The setting: which candidates depend on the hull, and why the hull sets the cost

Scheme	Underlying problem	Dedicated solver	Hull-based?
LESS	Linear Equivalence (LEP)	low-weight codeword / collision search	no
PERK	Permuted Kernel (PKP)	PKP solvers	no
SPECK (Baldi et al., 2025)	**PECK ≈ PEP** in the High-q regime	**Support Splitting Algorithm**	**yes**

PEP (Permutation Equivalence Problem)

Given generator matrices $G, G' \in \mathbb{F}_q^{k\times n}$ of codes C, C', find a permutation $\pi \in S_n$ sending C to C'.

Action on coordinates

Permutation matrix $P_\pi \in \{0,1\}^{n\times n}$:

$$\mathbf{x} = (x_1,\dots,x_n) \longmapsto \mathbf{x}\cdot P_\pi$$

$$\pi(C) = C\cdot P_\pi = \{\, c\cdot P_\pi : c \in C \,\}$$

In terms of generator matrices

$\pi(C) = C' \iff G' = U \cdot G \cdot P_\pi$ for some

$$U \in GL_k(\mathbb{F}_q), \qquad P_\pi \in \{0,1\}^{n\times n}$$

(U absorbs the choice of generator basis)

SSA (Sendrier, 2000) solves PEP in time $O(n^3 + n^2 q^h \log n), \quad h = \dim \mathrm{Hull}(C)$

Everything in this talk is about that single exponent h.

Why PEP-based schemes are forced into large hulls

Two separate reasons — one quantitative, one structural

1. Small h is simply cheap to attack

The SSA cost is dominated by q^h.
Any scheme with small hull dimension
is broken by the standard solver directly.

2. h = 0 is worse than cheap

Bardet–Otmani–Saeed-Taha (2019):
on trivial-hull (LCD) instances PEP reduces
back to GI, via orthogonal projectors.
With Babai (2016), GI is quasipolynomial.

The other direction was known much earlier: Petrank–Roth (1997) showed GI reduces to PEP, giving the first complexity lower bound. Together, trivial-hull PEP is **no harder than GI**.

So a PEP-based scheme must keep h large — and SPECK takes the extreme.

SPECK's key generation samples **self-orthogonal** codes; recommended parameters take k = n/2:

$$\mathbf{h = \dim \mathrm{Hull}(C) = k = n/2} \quad \Longrightarrow \quad \mathbf{O(n^3 + n^2\, q^{n/2} \log n)}$$

The hull gives no reduction at all there. That exponent is exactly what protects the scheme.

Is the standard inner product the only choice?

The question this paper answers

The hull is defined through the standard inner product. Nothing forces that particular bilinear form.

Identify a bilinear form with its structure matrix M:

$$B(x, y) = x\, M\, y^{T}, \qquad C^{\perp_M} = \{\, y : x\, M\, y^{T} = 0 \ \text{ for all } x \in C \,\}$$

$$\mathrm{Hull}_M(C) = C \cap C^{\perp_M}$$

M = I recovers the usual hull.

Two questions

(a) Is there any non-trivial M for which Hull_M is still a usable SSA invariant?

(b) If so, by how much can the hull dimension actually be reduced?

We answer both. Together the answers give a tight limit on the whole approach.

What SSA needs: permutation invariance

The only requirement Hull_M has to meet

SSA works because its signature is invariant under coordinate permutations. For Hull_M to replace Hull we need exactly:

$$\mathbf{Hull_M(\pi(C)) \;=\; \pi(Hull_M(C))}$$

for all $\pi \in S_n$ and all codes C

Nothing else is required.

- If a bilinear form fails this, matching signatures between C and C' is meaningless — SSA cannot use its hull at all.
- If it holds, every step of SSA goes through verbatim with Hull_M in place of Hull.

So the design space is: which structure matrices M give a permutation-invariant hull?
Theorem 1 answers this completely — and the answer is very restrictive.

Theorem 1 — Characterization

Which bilinear forms survive the permutation-invariance test

Let M be symmetric and nondegenerate, n ≥ 3. The following are equivalent.

(i) **every $\pi \in S_n$ is an isometry of B:** $B(\pi(x), \pi(y)) = B(x, y)$

(ii) $M = aI + bJ$ **for some** $a, b \in F_q$ **(I = identity, J = all-ones matrix)**

(iii) $\pi(C^{\perp_M}) = (\pi(C))^{\perp_M}$ **for every π and every code C**

(iv) $\pi(\mathrm{Hull}_M(C)) = \mathrm{Hull}_M(\pi(C))$ **for every π and every code C**

A two-parameter family, and nothing else.

Every SSA-compatible bilinear hull is one of these. The usual hull is the point (a, b) = (1, 0).

Proof sketch of Theorem 1

Elementary linear algebra plus one orbit count

(i) ⇒ (ii) is where the rigidity comes from.

1. The isometry condition, for every permutation matrix P:

   $$(xP)\,M\,(yP)^{T} = x\,M\,y^{T} \ \text{ for all } x, y \quad \Longleftrightarrow \quad P\,M\,P^{T} = M$$

2. Entry-wise, for all $\pi \in S_n$ and all i, j:

   $$M_{\pi(i),\,\pi(j)} = M_{ij}$$

3. S_n acting diagonally on [n] × [n] has exactly two orbits — the diagonal and the off-diagonal:

   $$\{\,(i, i)\,\} \qquad \text{and} \qquad \{\,(i, j) : i \neq j\,\}$$

4. So M is constant on each orbit:

   $$M_{ii} = \alpha, \quad M_{ij} = \beta \ \ (i \neq j) \quad \Longrightarrow \quad M = (\alpha - \beta)\,I + \beta\,J$$

The remaining implications (ii) ⇒ (iii) ⇒ (iv) ⇒ (i) are direct computations.
No representation theory beyond counting the two orbits of S_n on ordered pairs.

Theorem 2 — The dimension barely moves

Statement and its consequence on self-orthogonal codes

Let C be an $[n, k]_q$ code, let $M = aI + bJ$ with $a \neq 0$, and let $\mathrm{Hull}_M(C) = C \cap C^{\perp_M}$.

Theorem 2

$$\left| \dim \mathrm{Hull}_M(C) - \dim \mathrm{Hull}(C) \right| \leq 1$$

for every choice of $a, b \in F_q$ with $a \neq 0$

Corollary — the reduction is guaranteed on SPECK's codes

If C is self-orthogonal, $b \neq 0$, and $v = G \cdot 1^T \neq 0$, then

$$\dim \mathrm{Hull}_M(C) = k - 1$$

Not just a generic bound — the extremal case is achieved exactly on the codes SPECK uses.

Self-dual codes are the worst case for SSA (h = n/2) and the best case for us.

Proof sketch of Theorem 2

Two lemmas plus a rank-one perturbation

Materials

Lemma A — Rank perturbation

For matrices $A, B \in \mathbb{F}_q^{k\times k}$:

$$|\,\mathrm{rank}(A+B) - \mathrm{rank}(A)\,| \;\le\; \mathrm{rank}(B)$$

(standard; e.g., Horn–Johnson §0.4)

Lemma B — Hull dimension from Gram

For any generator matrix G of C:

$$\dim \mathrm{Hull}_M(C) = k - \mathrm{rank}(G\,M\,G^T)$$

($\mathrm{Hull}_M(C) = \{\, uG : uGMG^T = 0 \,\}$ as $u \in \mathbb{F}_q^k$)

Gram matrix under $M = aI + bJ$

$$G\,M\,G^T = a \cdot G\,G^T + b \cdot v\,v^T, \qquad v = G \cdot 1^T \in \mathbb{F}_q^{\,k}$$

The perturbation is rank one: $\mathrm{rank}(b \cdot v\,v^T) \le 1$.

Combining Lemmas A and B

$$|\,\mathrm{rank}(G\,M\,G^T) - \mathrm{rank}(a \cdot G\,G^T)\,| \;\le\; \mathrm{rank}(b \cdot v\,v^T) \;\le\; 1$$

$$\Longrightarrow \quad |\,\dim \mathrm{Hull}_M(C) - \dim \mathrm{Hull}(C)\,| \;\le\; 1 \qquad \blacksquare$$

Application to SPECK

PECK reduces to PEP in the High-q regime

SPECK (Baldi et al., 2025) is built on PECK. Its key generation samples a **self-orthogonal** $[n, k]_q$ code (recommended $k = n/2$) — exactly the case of the Corollary.

PECK instance $\{c, G, H'\}$ ▶ **PEP instance** $\{G, H'\}$ ▶ **Hull_M preprocessing** $M = I + J$ ▶ **SSA** on dim $n/2 - 1$

Two facts make this work

- High-q regime ($q > (n/e)^2$): the expected number of PECK solutions is ≈ 1, so PECK and PEP are computationally equivalent [Baldi et al.].
- Preprocessing is a single rank-one kernel computation, costing $O(k^2)$:

$$G\,G^{\mathsf{T}} = 0 \implies G\,M\,G^{\mathsf{T}} = v\,v^{\mathsf{T}}, \qquad \mathrm{Hull}_M(C) = \{\, uG : u \in \ker(v\,v^{\mathsf{T}}) \,\}$$

Bilinear-hull SSA solves PECK in

$$O(n^3 + n^2\, q^{n/2-1} \log n)$$

a factor of q below standard SSA

Concrete numbers on SPECK's parameters

n = 252, k = 126; $\log_2$-cost of SSA on the underlying PEP instance

Instance	q	$\log_2 C_{SSA}$	$\log_2 C_{SSA}^B$	Δ	Best known attack on SPECK
Speck-Low	127	880.6	873.6	**7.0**	$\approx 2^{130}$ (PKP collision search)
Speck-High	8861	1652.3	1639.2	**13.1**	exponential in n

SSA is not the bottleneck

- Both bilinear-hull costs still exceed 2^{870}.
- In the Low-q regime the cheapest known attack is PKP-based collision search at $\approx 2^{130}$, which does not use the hull at all.
- **We make no claim about SPECK's bit-security.**

Where it does matter — design side

- The result gives the **tight** SSA cost on PEP instances. For a target λ:

$$h \geq \lceil \lambda / \log_2 q \rceil + 1$$

rather than $\lceil \lambda / \log_2 q \rceil$ — relevant to future PEP-based designs shrinking n or q.

Read the other way round

Generalizing the hull to arbitrary bilinear forms **cannot endanger SPECK**: whatever M is chosen, the SSA cost stays far above the best known attack.

A cost reduction on SSA, not a better attack on SPECK.

Where the technique stops

LESS and PERK are outside the framework

LESS — Linear Equivalence (LEP)

For $M = aI + bJ$ with $b \neq 0$, Hull_M is **not** invariant under monomial transformations $\mu = D \cdot \pi$.

LEP's dedicated solvers do not use the hull anyway.

PERK — Permuted Kernel (PKP)

Hulls play no role in PKP.

The framework does not apply at all.

Why the closure route does not rescue LEP either

- LEP reduces to PEP via the closure construction, but it inflates the length from n to $n(q-1)$.
- For $q \geq 5$ the closure is always weakly self-dual, so the resulting PEP instance has near-maximal hull dimension — SSA performs poorly on it whatever bilinear form is used.

Hull-based techniques target PEP specifically. The scope of the result is exactly the PEP-based family.

Experimental validation

SageMath 10.0 — 66 self-orthogonal codes, fixed M = I + J

Parameters	Trials	dim Hull	dim Hull$_M$	Red.	Time (ms)
n = 16, q = 5	10	8	7	1	0.1
n = 64, q = 127	7	32	31	1	0.8
n = 128, q = 127	3	64	63	1	1.5
n = 252, q = 127	5	**126**	**125**	**1**	**2.0**
n = 252, q = 8861	3	**126**	**125**	**1**	**5.0**

What it confirms

- **66 / 66 trials**: the reduction is exactly 1.
- **No search over (a, b)**: the fixed M = I + J works.
- A single rank query; **2–5 ms** even at n = 252.

The factor-q speedup carries **no hidden preprocessing overhead**: milliseconds of preprocessing against a $q^{n/2-1}$ main loop.

Self-orthogonal generator matrices $G = (I_k \mid A)$ with $A\,A^T = -I_k$, randomized by $U \in GL_k(F_q)$ and a coordinate permutation.

Conclusion

A complete answer for the bilinear-form route

1. **Hull_M is SSA-compatible $\iff$ $M = aI + bJ$** (Theorem 1)
2. **Every such M changes dim Hull by at most one** (Theorem 2)
3. $\Longrightarrow$ **no bilinear form lowers the SSA cost by more than a factor of q** — a tight limit
4. Only PEP-based SPECK admits it ($\log_2 q \approx 13$ at $q = 8861$); LESS and PERK lie outside
5. SageMath: preprocessing is milliseconds, and no search over (a, b) is needed

Open directions

Beyond bilinear
higher-order tensor or non-bilinear invariants — can they beat −1?

Other equivalence problems
matrix / rank-metric setting (MEDS, ALTEQ)

Hybrid Low-q analysis
bilinear-hull SSA combined with PECK-constraint filtering

Thank you

Questions?

Keita Ishizuka · Mitsubishi Electric Corporation · keitaishizuka1994@gmail.com

Joint work with Yusuke Aikawa (The University of Tokyo) and Tomoki Moriya (Mitsubishi Electric)

Backup — SSA in one slide

Sendrier (2000), building on Leon (1982)

A signature is an invariant map S assigning to each coordinate an element of some set E with

$$\mathbf{S(\pi(C), \pi(i)) = S(C, i) \quad \text{for all } C, i, \pi}$$

- The signature used in practice: puncture at i, take the weight enumerator of C \ {i} (or a hash of it).
- Matching signatures across C and C' recovers π coordinate by coordinate.
- Weight enumeration of an $[n, k']_q$ code costs up to $q^{k'}$ — hence the exponential term.
- Since Hull is permutation-invariant, SSA runs on the hull: $q^k \to q^h$.

Related solvers. For LEP the dedicated solvers are Leon's low-weight-codeword algorithm, Beullens' collision search,
and the recent improvements of Bardet et al. and Budroni–Esser. **None of them depends on the hull.**

Backup — the algorithm

Bilinear-hull SSA on a PECK instance (High-q regime)

Stage 1 — preprocessing with $M = I + J$

$$\mathbf{v \leftarrow G 1^{T}}\ , \qquad \mathbf{W \leftarrow v\, v^{T} = G\, M\, G^{T}}\ , \qquad \mathbf{\ker(W) = \langle c_1, \ldots, c_{k-1} \rangle}$$

($G\, M\, G^{T} = v\, v^{T}$ because $G\, G^{T} = 0$ for a self-orthogonal code)

Basis of $\mathrm{Hull}_M(C)$: $\{ c_i\, G \}$, of dimension $k - 1$. $\mathrm{Hull}_M(C')$ is obtained the same way from H'.

Stage 2 — run standard SSA on $(\mathrm{Hull}_M(C), \mathrm{Hull}_M(C'))$ and return P

- Stage 1 costs $O(k^2)$ field operations — one rank-one kernel computation.
- Stage 2 is unchanged SSA, but on dimension $n/2 - 1$ instead of $n/2$.
- For $\mathrm{Hull}_M(C')$ from H': either derive G' at cost $O(n^3)$ (already paid inside SSA), or use $\mathrm{Hull}_M(C') = \{ y \in C' : y\, M\, G'^{T} = 0 \}$ directly.

Backup — High-q vs Low-q

Why the reduction needs the High-q regime

Expected number of PECK solutions (Baldi et al., Prop. 2):

$$\mathbf{N_{sol} \;=\; 1 + (\,|A_n(m)| - 1\,) \,/\, q^{n-k}}$$

High-q: $q > (n/e)^{1/(1-R)}$

$N_{sol} \approx 1$ with overwhelming probability

⇒ PECK and PEP are computationally equivalent

⇒ our reduction applies

Low-q: $q \ll (n/e)^{1/(1-R)}$

$N_{sol} \gg 1$

solving PEP alone does not identify the correct PECK permutation

- At SPECK's parameters R = 1/2, so the threshold is $q > (n/e)^2$.
- A precise hybrid analysis for Low-q — bilinear-hull SSA on PEP combined with PECK-constraint filtering — is left open.