Efficient Distributed DPF KeyGen with Active Security for QA-SD

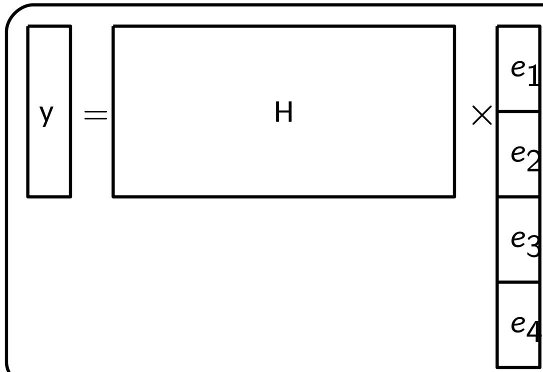
ePrint 2024/429 & 2023/845 & 2024/426

March 26, 2024· Presented by Hongrui Cui

Motivations

Silent generation/PCG of Beaver triples over $\mathbb{F}_2$

- Application 1: Silent GMW Preprocessing
- Application 2: GC-PCG



Alice: z^A, y

Bob: z^B, Δ

Correlation: $z^A + z^B = y \cdot \Delta$

Paradigm for COT/sVOLE PCG

- Generate **sparse** correlations
- Compress with linear map (LPN)

Let $H \in \mathbb{F}_p^{N \times cN}$, $|e| = t$.

Key problem with "Quadratic" correlation

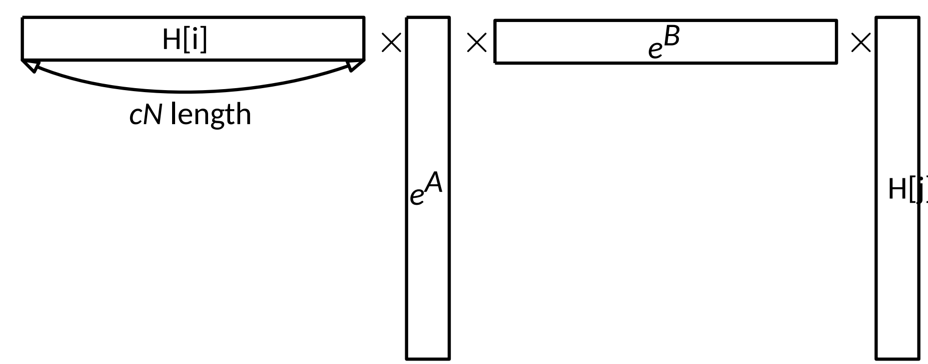
- Quadratic computation blow-up
- Consider $10^6 \rightarrow 10^{12}$

Alice: z^A, y^A, x^A

Bob: z^B, y^B, x^B

$z^A + z^B = (x^A + x^B) \cdot (y^A + y^B)$

Consider $x^A[i] \cdot y^B[j] = \langle H[i], e^A \rangle \cdot \langle H[j], e^B \rangle$



For regular LPN over $\mathbb{F}_p$, $H \leftarrow \mathbb{F}_p^{N \times cN}$, expected $O(c^2N^2)$ work

Previous Solutions

BCGIKS20

- **Ring-LPN**: Replace $\langle H, e\rangle$ with $\langle a(X), e(X)\rangle$ for $a(X), e(X) \in (\mathbb{F}_q[X]/(f(X)))^c$
- Now evaluating cross-term requires $O(c^2 N \log N) = \tilde{O}(N)$ work (with FFT)
- The resulting polynomial $\left\langle a \otimes a, e^A \otimes e^B \right\rangle$ is isomorphic to $\mathbb{F}_q^N$
- **CRT** requires $q > N$

BCGIKS20 (FOCS'20)

- **VD-LPN**

BCGIKRS22

- **EA-LPN** Replace $\langle H, e\rangle$ with $\langle E \cdot A, e\rangle$ for c-sparse E, upper-triangular A
- Now evaluating cross-term requires $O(c^2 t^2 N)$ work
- Requires further cryptanalysis

BCCD23

- **QA-SD** Replace univariate polynomial in Ring-LPN with multivariate polynomial
- Generate Beaver triples over $\mathbb{F}_q$ for $q \geq 3$

BBCCDS24

- **QA-SD** over $\mathbb{F}_4$ implies Beaver triples over $\mathbb{F}_2$.
- FFT optimizations and implementation

Distributed Setup of PCG

Ds17

- Distributed setup of DPF keys with black-box 2PC

ZGYZYW24

- Half-tree DPF KeyGen from BDOZ-authenticated inputs and SPDZ-authenticated-payload

Ultimate Goal

- End-to-end MPC with malicious security
- 1. Correct LPN variant
- 2. Matching $\Pi_{\mathsf{FSS.KeyGen}}$ with malicious security

Quasi-Abelian Syndrome Decoding

$\mathbb{F}_q[G] \stackrel{\text{def}}{=} \left\{\sum_{g\in G} a_g g \mid a_g \in \mathbb{F}_q\right\}$

- $G = \{1_G\}$: $\mathbb{F}_q[G] = \mathbb{F}_q$
- $G = \mathbb{Z}/n\mathbb{Z}$: $\mathbb{F}_q[G] = \mathbb{F}_q[X]/(X^n - 1)$

13.1: Finite Abelian Groups

In our investigation of cyclic groups we found that every group of prime order was isomorphic to $\mathbb{Z}_p$, where p was a prime number. We also determined that $\mathbb{Z}_{mn} \cong \mathbb{Z}_m \times \mathbb{Z}_n$ when $\gcd(m,n) = 1$. In fact, much more is true. Every finite abelian group is isomorphic to a direct product of cyclic groups of prime power order; that is, every finite abelian group is isomorphic to a group of the type

$$\mathbb{Z}_{p_1^{\alpha_1}} \times \cdots \times \mathbb{Z}_{p_n^{\alpha_n}},$$

where each p_k is prime (not necessarily distinct).

Multiplication by convolution

$$\left(\sum_{g\in G} a_g g\right)\left(\sum_{g\in G} b_g g\right) \stackrel{\text{def}}{=} \sum_{g\in G}\left(\sum_{h\in G} a_h b_{h^{-1}g}\right) g$$

(Search) QA-SD problem. Given $\mathbf{H} = (\mathbf{1} \mid \mathrm{a})$ a paritycheck matrix of a random systematic quasi-abelian code, a target weight $t \in \mathbb{N}$ and a syndrome $\mathbf{s} \in \mathbb{F}_q[G]$, the goal is to recover an error $\mathbf{e} = (\mathbf{e}_1 \mid \mathbf{e}_2)$ with $\mathbf{e}_i \leftarrow \mathcal{D}_t(\mathbb{F}_q[G])$ such that $\mathbf{He}^T = \mathbf{s}$, i.e. $\mathbf{e}_1 + \mathbf{a}\cdot\mathbf{e}_2 = \mathbf{s}$.

Choice of G

- The most interesting case is $\mathbb{F}_q = \mathbb{F}_2$
- However, when $q = 2$, $G = \{1_G\} \otimes ... \otimes \{1_G\}$ has order 1
- **FOLEAGE** sets $q = 4$, $G = (\mathbb{Z}/3\mathbb{Z})^n$
- $\mathbb{F}_q[G] \cong \mathbb{F}_q[X_1, ..., X_n]/(X_1^3 - 1, ..., X_n^3 - 1) \cong \mathbb{F}_q^{3^n}$

Why $\mathbb{F}_4$:

Let $\left([\![a]\!]^4, [\![b]\!]^4, [\![ab]\!]^4\right)$ be a Beaver triple over $\mathbb{F}_4$. Writing $x = x(0) + \theta \cdot x(1)$ for any $x \in \mathbb{F}_4$, with θ a root of the polynomial $X^2 + X + 1$ (hence $\theta^2 = \theta + 1$), we have

$$a \cdot b = a(0)b(0) + a(1)b(1) + \theta \cdot (a(0)b(1) + a(1)b(0) + a(1)b(1))$$

$$\rightarrow (ab)(0) = a(0)b(0) + a(1)b(1)$$

2-Party Case

$$(a \cdot b)(0) = [\![ab]\!]_A^4(0) + [\![ab]\!]_B^4(0) = a(0)b(0) + a(1)b(1),$$

$$\underbrace{a(0)a(1) + [\![ab]\!]_A^4(0)}_{\text{known by } A} + \underbrace{b(0)b(1) + [\![ab]\!]_B^4(0)}_{\text{known by } B} = \underbrace{(a(0) + b(1))}_{\text{shared by } A,B} \cdot \underbrace{(a(1) + b(0))}_{\text{shared by } A,B}.$$

Optimized Distributed KeyGen

Protocol $\Pi_{\mathsf{rDPF\text{-}CW}}$

PARAMETERS:

- Party $\sigma \in \{0,1\}$ has input $[\![\alpha_i]\!]_\sigma \in \mathbb{F}_3, r_i^\sigma \in \{0,1\}^\lambda, (s_{i,j}^\sigma \| t_{i,j}^\sigma)_{j\in\{0,1,2\}} \in \{0,1\}^{3(\lambda+1)}$.
- An instantiation of chosen $\binom{1}{3}$-OT.

PROTOCOL:

For each party $\sigma \in \{0,1\}$:

1: Sample $z^\sigma \leftarrow_R \{0,1\}^{3(\lambda+1)}$.

2: Define

$$\begin{aligned}
\mathbf{C}_0^\sigma &:= (r_i^\sigma \oplus s_{i,0}^\sigma \| (t_{i,0}^\sigma \oplus \sigma),\ s_{i,1}^\sigma \| t_{i,1}^\sigma,\ s_{i,2}^\sigma \| t_{i,2}^\sigma) \oplus z^\sigma && \triangleright [\![\mathsf{CW}_i]\!]_\sigma \text{ when } \alpha_i = 0\\
\mathbf{C}_1^\sigma &:= (s_{i,0}^\sigma \| t_{i,0}^\sigma,\ r_i^\sigma \oplus s_{i,1}^\sigma \| (t_{i,1}^\sigma \oplus \sigma),\ s_{i,2}^\sigma \| t_{i,2}^\sigma) \oplus z^\sigma && \triangleright [\![\mathsf{CW}_i]\!]_\sigma \text{ when } \alpha_i = 1\\
\mathbf{C}_2^\sigma &:= (s_{i,0}^\sigma \| t_{i,0}^\sigma,\ s_{i,1}^\sigma \| t_{i,1}^\sigma,\ r_i^\sigma \oplus s_{i,2}^\sigma \| (t_{i,2}^\sigma \oplus \sigma)) \oplus z^\sigma && \triangleright [\![\mathsf{CW}_i]\!]_\sigma \text{ when } \alpha_i = 2\\
\mathbf{M}_0^\sigma &:= (\mathbf{C}_0^\sigma, \mathbf{C}_1^\sigma, \mathbf{C}_2^\sigma),\ \mathbf{M}_1^\sigma := (\mathbf{C}_1^\sigma, \mathbf{C}_2^\sigma, \mathbf{C}_0^\sigma),\ \mathbf{M}_2^\sigma := (\mathbf{C}_2^\sigma, \mathbf{C}_0^\sigma, \mathbf{C}_1^\sigma)
\end{aligned}$$

3: Invoke $\binom{1}{3}$-OT with party $\bar\sigma$ as follows:

- Party $\bar\sigma$ plays the role of the sender with inputs $\mathbf{M}^{\bar\sigma}_{[\![\alpha_i]\!]_{\bar\sigma}}$.
- Party σ plays the role of the receiver and inputs $[\![\alpha_i]\!]_\sigma \in \mathbb{F}_3$.
- Party σ gets $\mathbf{M}^{\bar\sigma}_{[\![\alpha_i]\!]_{\bar\sigma}}[[\![\alpha_i]\!]_\sigma] \in \{0,1\}^{3(\lambda+1)}$ while party $\bar\sigma$ gets nothing.

4: Define $[\![\mathsf{CW}_i]\!]_\sigma := \mathbf{M}^{\bar\sigma}_{[\![\alpha_i]\!]_{\bar\sigma}}[[\![\alpha_i]\!]_\sigma] \oplus z^\sigma$ and broadcast $[\![\mathsf{CW}_i]\!]_\sigma$.

5: Construct $\mathsf{CW}_i := [\![\mathsf{CW}_i]\!]_\sigma \oplus [\![\mathsf{CW}_i]\!]_{\bar\sigma} \in \{0,1\}^{3(\lambda+1)}$.

6: Output $(\mathsf{CW}_{i,0}, \mathsf{CW}_{i,1}, \mathsf{CW}_{i,2})$.

Distributed KeyGen for Half-Tree

Protocol Π_{DPF}

This protocol invokes $\Pi_{\mathsf{BatchCheck}}$ (Figure 2) as a sub-protocol.

Initialize: For each $b \in \mathbb{F}_2$, P_b samples $\Delta_b \leftarrow \mathbb{F}_{2^\lambda}$ such that $\mathsf{lsb}(\Delta_b) = b$, and sends $(\mathsf{init}, b, \Delta_b)$ to $\mathcal{F}_{\mathsf{aBit}}$.

Protocol inputs: Two parties P_0 and P_1 hold n BDOZ-style authenticated sharings $\langle\!\langle \alpha^{(i)} \rangle\!\rangle = (\langle\!\langle \alpha^{(i)} \rangle\!\rangle_0, \langle\!\langle \alpha^{(i)} \rangle\!\rangle_1)$ for all $i \in [0, n)$ as well as a SPDZ-style authenticated sharing $[\![\beta]\!] = ([\![\beta]\!]_0, [\![\beta]\!]_1)$. Let $N = 2^n$ for some $n \in \mathbb{N}$. Let $\mathcal{H}_0 : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$ be a CCR hash function and $\mathcal{H}_1 : \{0,1\}^\lambda \rightarrow \{0,1\}^{2\lambda}$ such that $\mathcal{H}_1(x) := \mathcal{H}_0(x) \,\|\, \mathcal{H}_0(x \oplus 1)$.

Generate SPDZ-style authenticated sharings of DPF outputs: Let $\langle\!\langle \alpha^{(i)} \rangle\!\rangle_b = (\alpha_b^{(i)}, \mathsf{K}_b[\alpha_{1-b}^{(i)}], \mathsf{M}_b[\alpha_b^{(i)}])$ and $[\![\beta]\!]_b = (\beta_b, \mathsf{M}_b[\beta])$ for each $b \in \{0,1\}$. The parties P_0 and P_1 do the following.

1. Both parties call $\mathcal{F}_{\mathsf{coin}}$ to sample a public randomness $W \in \mathbb{F}_{2^\lambda}$. Each party P_b sets $(s_b^{(0,0)} \,\|\, t_b^{(0,0)}) := \Delta_b \oplus W \in \{0,1\}^\lambda$.
2. For each $b \in \{0,1\}$, for each $i \in [0,n)$, P_b computes the following:

$$\mathsf{CW}_b^{(i)} := \left(\bigoplus_{j \in [0,2^i)} \mathcal{H}_0(s_b^{(i,j)} \,\|\, t_b^{(i,j)}) \right) \oplus \Delta_b \oplus \left(\alpha_b^{(i)} \cdot \Delta_b \oplus \mathsf{K}_b[\alpha_{1-b}^{(i)}] \oplus \mathsf{M}_b[\alpha_b^{(i)}] \right) \in \{0,1\}^\lambda,$$

and sends $\mathsf{CW}_b^{(i)}$ to P_{1-b}. For each $i \in [0,n)$, both parties compute $\mathsf{CW}^{(i)} := \mathsf{CW}_0^{(i)} \oplus \mathsf{CW}_1^{(i)}$, and each party P_b computes:

$$\left(s_b^{(i+1,2j)} \,\|\, t_b^{(i+1,2j)} \right) := \mathcal{H}_0 \left(s_b^{(i,j)} \,\|\, t_b^{(i,j)} \right) \oplus t_b^{(i,j)} \cdot \mathsf{CW}^{(i)} \text{ for each } j \in [0, 2^i),$$

$$\left(s_b^{(i+1,2j+1)} \,\|\, t_b^{(i+1,2j+1)} \right) := \mathcal{H}_0 \left(s_b^{(i,j)} \,\|\, t_b^{(i,j)} \right) \oplus \left(s_b^{(i,j)} \,\|\, t_b^{(i,j)} \right) \oplus t_b^{(i,j)} \cdot \mathsf{CW}^{(i)} \text{ for each } j \in [0, 2^i).$$

Distributed KeyGen for Half-Tree (Continued)

3. For each $b \in \{0,1\}$, P_b computes

$$\mathsf{CW}_b^{(n)} := \left(\bigoplus_{j\in[0,N)} \mathcal{H}_1(s_b^{(n,j)} \,\|\, t_b^{(n,j)})\right) \oplus (\beta_b \,\|\, \mathsf{M}_b[\beta]) \in \{0,1\}^{2\lambda},$$

and sends $\mathsf{CW}_b^{(n)}$ to P_{1-b}. Then, both parties compute $\mathsf{CW}^{(n)} := \mathsf{CW}_0^{(n)} \oplus \mathsf{CW}_1^{(n)}$. For each $b \in \{0,1\}$, P_b computes

$$[\![u^{(j)}]\!]_b := \left(u_b^{(j)} = t_b^{(n,j)}, \mathsf{M}_b[u^{(j)}] = (s_b^{(n,j)} \,\|\, t_b^{(n,j)})\right) \text{ for each } j \in [0,N),$$

$$[\![v^{(j)}]\!]_b = \left(v_b^{(j)} \,\|\, \mathsf{M}_b[v^{(j)}]\right) := \mathcal{H}_1\left(s_b^{(n,j)} \,\|\, t_b^{(n,j)}\right) \oplus t_b^{(n,j)} \cdot \mathsf{CW}^{(n)} \text{ for each } j \in [0,N).$$

4. As in the **Rand** process of protocol Π_{2PC} (Figure 4), both parties call functionality $\mathcal{F}_{\mathsf{aBit}}$ to generate $[\![r]\!]$ with a random $r \in \mathbb{F}_{2^\lambda}$. Then, both parties call functionality $\mathcal{F}_{\mathsf{coin}}$ to sample a random challenge $\chi \in \mathbb{F}_{2^\lambda}$, and locally compute

$$[\![a]\!] := \sum_{j\in[0,N)} \chi^j \cdot [\![u^{(j)}]\!] + \sum_{j\in[0,N)} \chi^{N+j} \cdot [\![v^{(j)}]\!] + [\![r]\!].$$

5. As in the **Open** process of protocol Π_{2PC}, both parties open $[\![a]\!]$ to obtain $\tilde{a} = a_0 + a_1 \in \mathbb{F}_{2^\lambda}$ by letting P_0 send a_0 to P_1 and P_1 send a_1 to P_0 in parallel. Then, both parties run sub-protocol $\Pi_{\mathsf{BatchCheck}}$ (Figure 2) on input $([\![a]\!], \tilde{a})$ to check $a = \tilde{a}$.

6. For each $j \in [0,N)$, both parties obtain $[\![u^{(j)}]\!] = ([\![u^{(j)}]\!]_0, [\![u^{(j)}]\!]_1)$ and $[\![v^{(j)}]\!] = ([\![v^{(j)}]\!]_0, [\![v^{(j)}]\!]_1)$.