Private Outsourcing of Polynomial Evaluation and Matrix Multiplication using Multilinear Maps

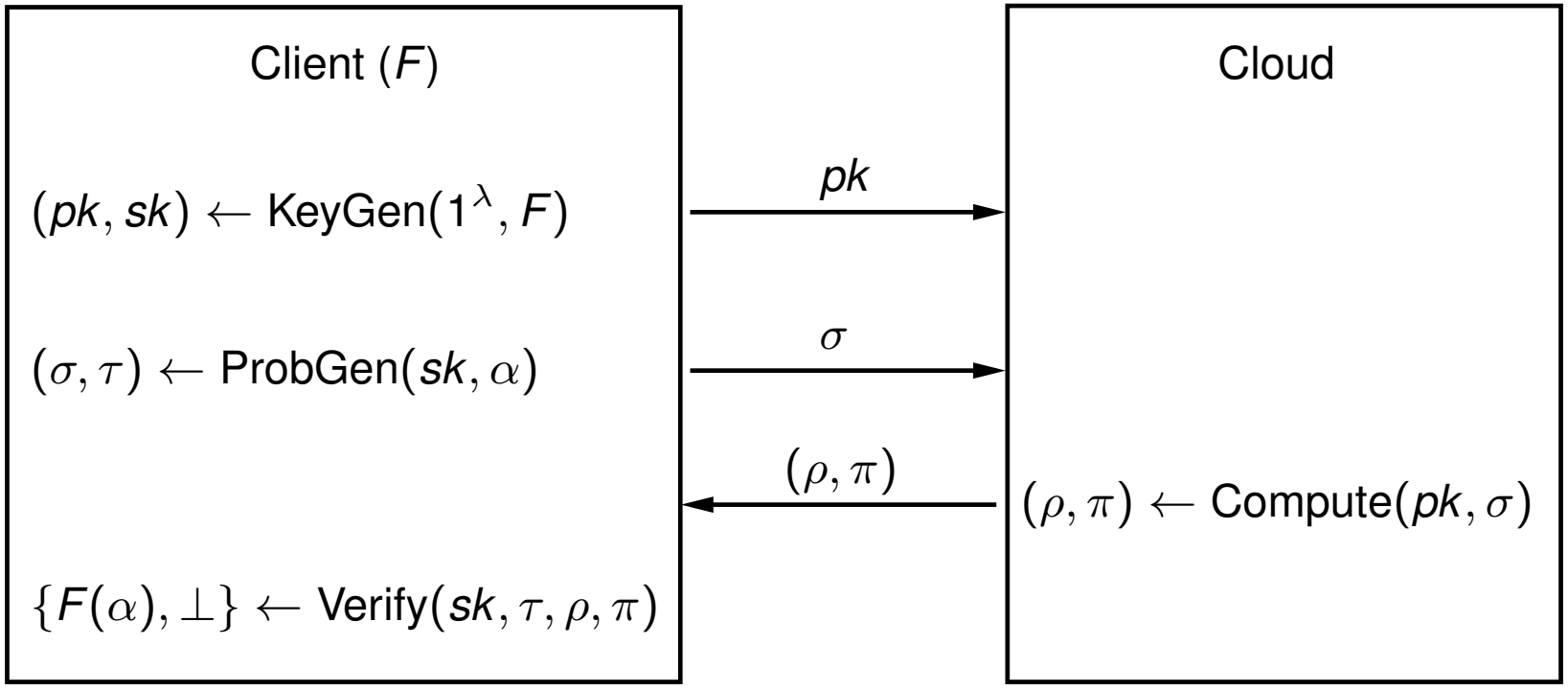
Liang Feng Zhang, Reihaneh Safavi-Naini

Institute for Security, Privacy and Information Assurance
Department of Computer Science
University of Calgary

Cloud Computing

- Weak Clients: Smart Phones; Netbooks
- Clouds: Amazon EC2; Google Compute Engine
- A Typical Model:
  - The client has a computationally intensive function F
  - The client gives F to the cloud
  - To compute $F(\alpha)$, the client gives α to the cloud
  - The cloud returns $\rho = F(\alpha)$ if it is honest
  - The client must verify when the cloud is untrusted
  - The verification should be much more efficient
- Solution: Gennaro, Gentry and Parno [GGP10]

Verifiable Computation (VC)



Correctness: $\text{Verify}(sk, \tau, \rho, \pi) = F(\alpha)$

Security: cannot forge $(\bar{\rho}, \bar{\pi})$ s.t. $\text{Verify}(sk, \tau, \bar{\rho}, \bar{\pi}) \notin \{F(\alpha), \perp\}$

Efficiency: $T_{\text{ProbGen}} + T_{\text{Verify}} = o(T_{F(\alpha)})$

Privacy

- The client has no reason to trust the cloud with the knowledge of its function F and input α
- Privacy is important when F or α is sensitive
  - F contains financial data and α indicates the client's interest
  - F contains medial data and α indicates the client's identity
- **Input privacy**: hide the input α from the cloud
- **Function privacy**: hide the function F from the cloud
- Our goal: VC with input privacy and function privacy

Multilinear Maps and Assumptions

- Postulated by Boneh and Silverberg [BS02]
- Candidate multilinear maps by [GGH13,CLT13]
- Multilinear map generator $\mathcal{G}$

$$\Gamma = (N, G_1, \ldots, G_k, e, g_1, \ldots, g_k) \leftarrow \mathcal{G}(1^\lambda, k)$$

  - $N = pq$ for λ-bit primes $p \neq q$; $G_i = \langle g_i \rangle$, order N ($i \in [k]$)
  - $e : G_i \times G_j \to G_{i+j}$, where $e(g_i^a, g_j^b) = g_{i+j}^{ab}$ ($i + j \leq k$)
  - $e : G_1 \times \cdots \times G_1 \to G_k$: $e(g_1^{a_1}, \ldots, g_1^{a_k}) = g_k^{a_1 \cdots a_k}$

Multilinear Maps and Assumptions (cont.)

- **SDA**: $(\Gamma, u) \equiv_c (\Gamma, u^q)$, where $u \leftarrow G_i$;
- **MSDH**: $\Pr[\mathcal{A}(\Gamma, g_1, g_1^s, \ldots, g_1^{s^n}) = (a, g_k^{\frac{1}{s+a}})]$, where $s \leftarrow \mathbb{Z}_N$
- **3-Linear**: $k = 3$, $u_0, u_1, u_2, u_3 \leftarrow G_1, a_0, a_1, a_2, a_3 \leftarrow \mathbb{Z}_N$

$$\begin{pmatrix} u_1 & u_2 & u_3 & u_0 \\ u_1^{a_1} & u_2^{a_2} & u_3^{a_3} & u_0^{a_1+a_2+a_3} \end{pmatrix} \equiv_c \begin{pmatrix} u_1 & u_2 & u_3 & u_0 \\ u_1^{a_1} & u_2^{a_2} & u_3^{a_3} & u_0^{a_0} \end{pmatrix}$$

- **3-MDDH**: $k = 3$, $a_0, a_1, a_2, a_3, b \leftarrow \mathbb{Z}_N$

$$(\Gamma, g_1^{a_0}, g_1^{a_1}, g_1^{a_2}, g_1^{a_3}, g_3^{a_0 a_1 a_2 a_3}) \equiv_c (\Gamma, g_1^{a_0}, g_1^{a_1}, g_1^{a_2}, g_1^{a_3}, g_3^{b})$$

Our Results

- Polynomial Evaluation ($k = 2\lfloor \log(n+1) \rfloor + 1$)
  - Function: a high degree poly $f(x) = \sum_{i=0}^{n} f_i x^n \in \mathbb{F}_q[x]$
  - Input: a field element $\alpha \in \mathbb{F}_q$
  - Assumptions: SDA, MSDH
  - Result: a VC Scheme with input and function privacy
- Matrix Multiplication ($k = 3$)
  - Function: a matrix $M = (M_{ij}) \in \mathbb{F}_q^{n \times n}$
  - Input: a vector $x = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$
  - Assumption: SDA, 3-Linear and 3-MDDH
  - Result: a VC Scheme with input and function privacy
- Applications: Private information retrieval

An Encryption Scheme Based on SDA

- $(pk, sk) \leftarrow \mathrm{Gen}(1^\lambda, k)$
  - pick $\Gamma = (N, G_1, \ldots, G_k, e, g_1, \ldots, g_k) \leftarrow \mathcal{G}(1^\lambda, k)$
  - pick $u \leftarrow G_1$, compute $h = u^q$ $pk = (\Gamma, g_1, h)$; $sk = p$
- $c \leftarrow \mathrm{Enc}(pk, m)$: pick $r \leftarrow \mathbb{Z}_N$, compute $c = g_1^m h^r$
- $m \leftarrow \mathrm{Dec}(sk, c)$: compute $m \in \mathbb{M}$ s.t. $c^p = (g_1^p)^m$

- Denoted as BGN_k (recall [BGN05] for $k = 2$)
- $|\mathbb{M}| = \mathrm{poly}(\lambda)$; $\mathbb{C} = G_1(G_i)$; SDA-based security
- $\mathrm{Enc}(\alpha_1), \mathrm{Enc}(\alpha_2) \Rightarrow \mathrm{Enc}(\alpha_1 + \alpha_2)$ (multiplication)
- $\mathrm{Enc}(\alpha_1), \ldots, \mathrm{Enc}(\alpha_k) \Rightarrow \mathrm{Enc}(\alpha_1 \cdots \alpha_k)$ (pairing)

Computing on the Exponents

- Setting for polynomial evaluation
  - $f(x) = f_0 + f_1 x + \cdots + f_n x^n$; α; $k = \lceil \log(n+1) \rceil$
  - Set up BGN_k with $pk = (\Gamma, g_1, h)$ and $sk = p$
  - For $\ell \in [k]$, $\sigma_\ell = \mathrm{Enc}(\alpha^{2^{\ell-1}})$; $\sigma = (\sigma_1, \ldots, \sigma_k)$
  - $s \in \mathbb{Z}_N$ and $S = \{g_1^{s^{2^{\ell-1}}} : \ell \in [k]\}$
- From $f(x)$ and σ to $\mathrm{Enc}(f(\alpha))$
  - $0 \le i \le n$, $\exists i_1, \ldots, i_k \in \{0,1\}$ s.t. $i = \sum_{\ell=1}^{k} i_\ell 2^{\ell-1}$
  - $f_i \alpha^i = f_i \cdot \alpha^{i_1} (\alpha^2)^{i_2} \cdots (\alpha^{2^{k-1}})^{i_k}$
  - $e(\sigma_1^{i_1}, \ldots, \sigma_k^{i_k})^{f_i} = \mathrm{Enc}(f_i \alpha^i)$; ($\sigma_j^{i_j} \triangleq g_1$ when $i_j = 0$)
  - $\mathrm{Enc}(f(\alpha)) = \prod_{i=0}^{n} \mathrm{Enc}(f_i \alpha^i)$;

Computing on the Exponents (cont.)

- From $f(x)$, σ and S to $\mathrm{Enc}\left(\frac{f(s)-f(\alpha)}{s-\alpha}\right)$ ($(2k+1)$-linear map)
  - $c(s) \triangleq \frac{f(s)-f(\alpha)}{s-\alpha} = \sum_{i=0}^{n-1} \sum_{j=0}^{i} f_{i+1} \alpha^j s^{i-j}$
  - From $f(x), \sigma$ and S to $\pi_{ij} = \mathrm{Enc}(f_{i+1} \alpha^j s^{i-j})$
  - Compute $\mathrm{Enc}\big(c(s)\big) = \prod_{i=0}^{n-1} \prod_{j=0}^{i} \pi_{ij}$
- Setting for matrix multiplication
  - $M = (M_{ij})$ is an $n \times n$ matrix; $x = (x_1, \ldots, x_n)'$ is a vector
  - Set up BGN_3 with $pk = (\Gamma, g_1, h)$ and $sk = p$
  - For $\ell \in [n]$, $\sigma_\ell = \mathrm{Enc}(x_\ell)$; $\sigma = (\sigma_1, \ldots, \sigma_n)$
- From M and $\mathrm{Enc}(x)$ to $\mathrm{Enc}(Mx)$
  - $\rho_i = \prod_{j=1}^{n} \sigma_j^{M_{ij}} = \mathrm{Enc}(\sum_{j=1}^{n} M_{ij} x_j)$ for every $i \in [n]$

Polynomial Evaluation (No Input Privacy)

- KeyGen($1^\lambda, f$):
  - Pick $\Gamma_2 = (N, G_1, G_2, e, g_1, g_2)$, $s \leftarrow \mathbb{Z}_N$, $t = g_1^{f(s)}$;
  - public key $pk = (\Gamma_2, g_1^s, \ldots, g_1^{s^n}, f)$; secret key $sk = s$.
- ProbGen(sk, α): output $\sigma = \alpha$, $\tau = \perp$;
- Compute(pk, σ):
  - compute $c(x)$ such that $f(x) - f(\alpha) = (x - \alpha)c(x)$;
  - compute and output $y = f(\alpha)$ and $\pi = g_1^{c(s)}$;
- Verify(sk, τ, ρ, π): $?e(tg_1^{-y}, g_1) = e(g_1^{s-\alpha}, \pi)$

Privacy: no privacy; Security: MSDH (k=2)

Polynomial Evaluation (Input Privacy)

- KeyGen($1^\lambda, f(x)$): $f(x) = f_0 + f_1 x + \cdots + f_n x^n$; $k = \lceil \log(n+1) \rceil$
  - $\Gamma \leftarrow \mathcal{G}(1^\lambda, 2k+1)$, $s \leftarrow \mathbb{Z}_N$, $t = g_1^{f(s)}$; $u \leftarrow G_1$, $h = u^q$;
  - $sk = (p, q, s, t)$, $pk = (\Gamma, h, g_1^s, \ldots, g_1^{s^{2^{k-1}}}, f)$.
- ProbGen(sk, α):
  - pick $r_\ell \leftarrow \mathbb{Z}_N$ and compute $\sigma_\ell = g_1^{\alpha^{2^{\ell-1}}} h^{r_\ell}$ for $\ell \in [k]$
  - $\sigma = (\sigma_1, \ldots, \sigma_k), \tau = \perp$.
- Compute(pk, σ): output $\rho = \mathrm{Enc}(f(\alpha))$, $\pi = \mathrm{Enc}(c(s))$
- Verify(sk, τ, ρ, π):
  - compute $y \in \mathbb{Z}_q$ such that $\rho^p = (g_k^p)^y$
  - check if $e(t/g_1^y, g_{2k}^p) = e(g_1^{s-\alpha}, \pi^p)$

Privacy: SDA; Security: MSDH $(2k+1)$

Polynomial Evaluation (Input and Function Privacy)

- KeyGen$(1^\lambda, f(x))$:
  - Γ, $s \leftarrow \mathbb{Z}_N$, $t = g_1^{f(s)}$; $u \leftarrow G_1$, $h = u^q$; $v_i \leftarrow \mathbb{Z}_N$, $\gamma_i = g_1^{f_i} h^{v_i}$;
  - $sk = (p, q, s, t)$; $pk = (\Gamma, h, g_1^s, \ldots, g_1^{s^{2^{k-1}}}; \gamma_0, \ldots, \gamma_n)$.
- ProbGen(sk, x): $\sigma = (\sigma_1, \ldots, \sigma_k)$ and $\tau = \perp$;
  - $r_\ell \leftarrow \mathbb{Z}_N$, $\sigma_\ell = g_1^{\alpha^{2^{\ell-1}}} h^{r_\ell}$ for every $\ell \in [k]$
- Compute(pk, σ): output $\rho = \mathrm{Enc}(f(\alpha))$ and $\pi = \mathrm{Enc}(c(s))$
- Verify(sk, τ, ρ, π):
  - compute $y \in \mathbb{Z}_q$ such that $\rho^p = (g_{k+1}^p)^y$
  - check if $e(t/g_1^y, g_{2k+1}^p) = e(g_1^{s-\alpha}, \pi^p)$

PRF with Closed-Form Efficiency

- A Construction Based on 3-Linear Assumption:
  - $\Gamma \leftarrow \mathcal{G}(1^\lambda, 3)$; $A_j, B_j, C_j \leftarrow G_1, \alpha_i, \beta_i, \gamma_i \leftarrow \mathbb{Z}_N$
  - $\mathsf{F}_K : [n]^2 \rightarrow G_1, (i,j) \rightarrow A_j^{\alpha_i} B_j^{\beta_i} C_j^{\gamma_i}$
- Closed-Form Efficiency: $\mathbf{Comp}_i = \prod_{j=1}^{n} \mathsf{F}_K(i,j)^{x_j}$ $(i \in [n])$
  - $A = \prod_{i=1}^{n} A_i^{x_i}, B = \prod_{i=1}^{n} B_i^{x_i}, C = \prod_{i=1}^{n} C_i^{x_i}$
  - $\mathbf{Comp}_i = A^{\alpha_i} B^{\beta_i} C^{\gamma_i}$ for every $i \in [n]$
- Introduced by Benabbas, Gennaro and Vahlis [BGV11]

Matrix Multiplication (Input Privacy)

- KeyGen($1^\lambda, M$):
  - Pick Γ, K and $a \leftarrow \mathbb{Z}_N$; $T_{ij} = g_1^{p^2 a M_{ij}} \cdot \mathsf{F}_K(i,j)$ for $(i,j) \in [n]^2$
  - Pick $u \leftarrow G_1$, $h = u^q$; $sk = (p, q, K, a)$; $pk = (\Gamma, h, M, T)$
- ProbGen(sk, x): $\sigma = (\sigma_1, \ldots, \sigma_n), \tau = (\tau_1, \ldots, \tau_n)$
  - $r_j \leftarrow \mathbb{Z}_N$, $\sigma_j = g_1^{x_j} h^{r_j}$, $\tau_i = e(\prod_{j=1}^n \mathsf{F}_K(i,j)^{x_j}, g_2^p)$ $(i,j \in [n])$
- Compute(pk, σ):
  - compute $\rho_i = \prod_{j=1}^n \sigma_j^{M_{ij}}$ and $\pi_i = \prod_{j=1}^n e(T_{ij}, \sigma_j)$ for $i \in [n]$
- Verify(sk, τ, ρ, π):
  - compute y_i s.t. $\rho_i^p = (g_1^p)^{y_i}$ and verify if $e(\pi_i, g_1^p) = g_3^{p^3 a y_i} \cdot \tau_i$
  - output $y = (y_1, \ldots, y_n)$ if the 2nd equality holds for $i \in [n]$

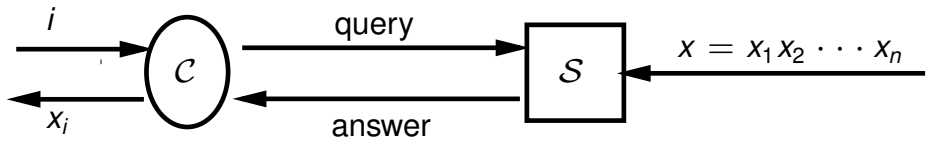
Privacy: SDA; Security: 3-Linear and 3-MDDH

Matrix Multiplication (Input and Function Privacy)

- KeyGen$(1^\lambda, M)$:
  - Γ, K and $a \leftarrow \mathbb{Z}_N$; $T_{ij} = g_1^{p^2 a M_{ij}} \cdot \mathsf{F}_K(i,j)$; $u \leftarrow G_1$, $h = u^q$
  - $v_{ij} \leftarrow \mathbb{Z}_N$, $\gamma_{ij} = g_1^{M_{ij}} h^{v_{ij}}$
  - $sk = (p, q, K, a)$ and $pk = (\Gamma, h, \gamma, T)$
- ProbGen(sk, x): output $\sigma = (\sigma_1, \ldots, \sigma_n)$, $\tau = (\tau_1, \ldots, \tau_n)$
  - $r_j \leftarrow \mathbb{Z}_N$, $\sigma_j = g_1^{x_j} h^{r_j}$; $\tau_i = e(\prod_{j=1}^n \mathsf{F}_K(i,j)^{x_j}, g_2^p)$ $((i,j) \in [n]^2)$
- Compute(pk, σ): output $\rho = (\rho_1, \ldots, \rho_n)$, $\pi = (\pi_1, \ldots, \pi_n)$
  - $\rho_i = \prod_{j=1}^n e(\gamma_{ij}, \sigma_j)$; $\pi_i = \prod_{j=1}^n e(T_{ij}, \sigma_j)$
- Verify(sk, τ, ρ, π):
  - compute y_i s.t. $\rho_i^p = (g_2^p)^{y_i}$ and check if $e(\pi_i, g_1^p) = \eta^{p y_i} \cdot \tau_i$
  - output $y = (y_1, \ldots, y_n)$ if the 2nd equality holds for $i \in [n]$

Applications: Private Information Retrieval

- Private information retrieval: [CGKS95,KO97]



- PIR server computation is intensive $\Rightarrow$ outsourcing
- Solution 1: using the scheme for polynomial evaluation
  - $f(x) = f_0 + f_1 x + \cdots + f_n x^n$, where $f(i) = D_i$ for $i \in [n]$
  - $\alpha = i$ for retrieving D_i
- Solution 2: using the scheme for matrix multiplication
  - D is considered as a matrix $M = (M_{uv})$, $i \leftrightarrow (u, v)$
  - $\alpha = (\alpha_1, \ldots, \alpha_{\sqrt{n}})$ is the vth unit vector ($\alpha_v = 1$, $\alpha_{v'} = 0$)

Comparisions and Future Work

- [GGP10]: FHE, Boolean circuits
- [BF11]: FHE, FEs that compute MACs (Hard to realize)
- [PRV12]: Attribute-hiding KP-ABE, Boolean formulas
- This work: FHE-free; no Boolean circuits or formulas
- Future work: multilinear map-based VC schemes with special properties such as public verification, public delegation, multi-function delegation

Thank you!