# Homomorphic Secret Sharing: Optimizations and Applications

Elette Boyle, *Geoffroy Couteau*, Niv Gilboa, Yuval Ishai, and Michele Orrù

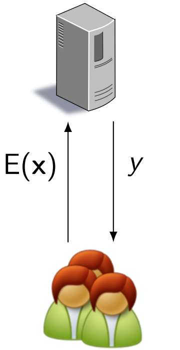
KIT

November 2, 2017

# Secure Outsourcing

# Secure Outsourcing

## From FHE




Secret input x
$\mathrm{Dec}(y) = f(x)$

## From MPC




Secret input x
$y_0 \oplus y_1 = f(x)$

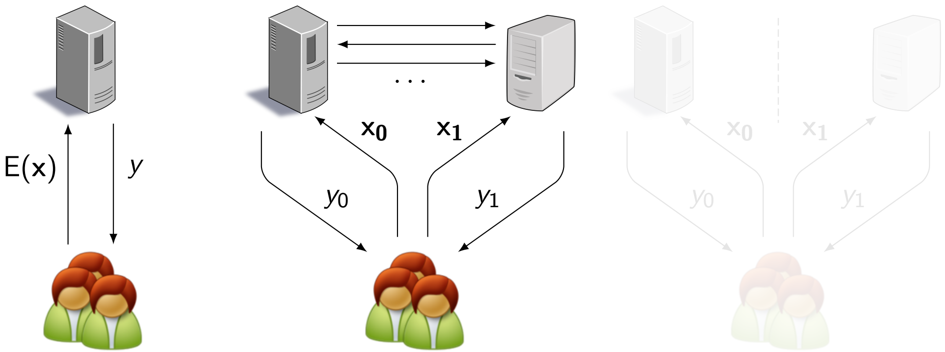
## From HSS




Secret input x
$y_0 \oplus y_1 = f(x)$

# Secure Outsourcing

From FHE | From MPC | From HSS



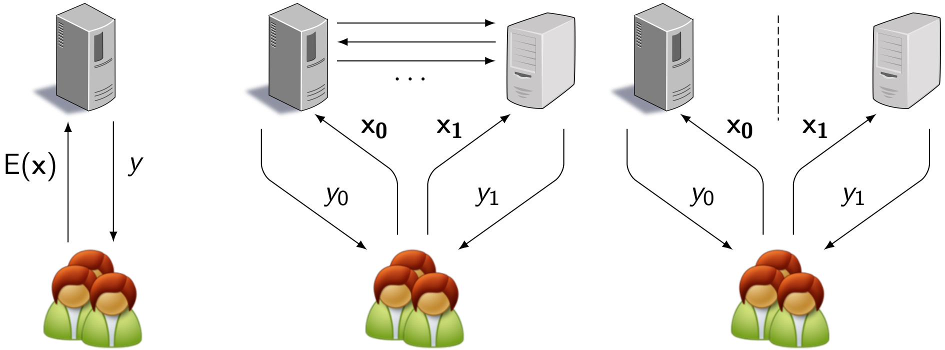

**From FHE:**
Secret input x
$\mathrm{Dec}(y) = f(\mathbf{x})$

**From MPC:**
Secret input x
$y_0 \oplus y_1 = f(\mathbf{x})$

**From HSS:**
Secret input x
$y_0 \oplus y_1 = f(\mathbf{x})$

# Secure Outsourcing

From FHE — From MPC — From HSS




**From FHE**

Secret input $\mathbf{x}$

$\mathrm{Dec}(y) = f(\mathbf{x})$

**From MPC**

Secret input $\mathbf{x}$

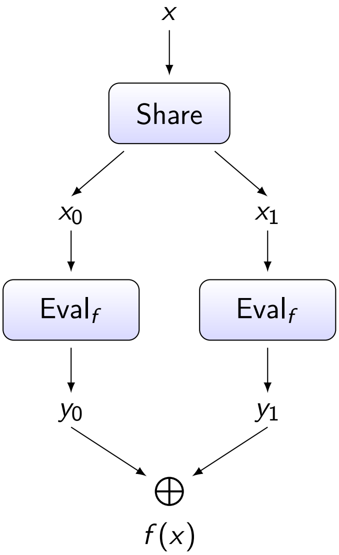
$y_0 \oplus y_1 = f(\mathbf{x})$

**From HSS**

Secret input $\mathbf{x}$

$y_0 \oplus y_1 = f(\mathbf{x})$

# Homomorphic Secret Sharing



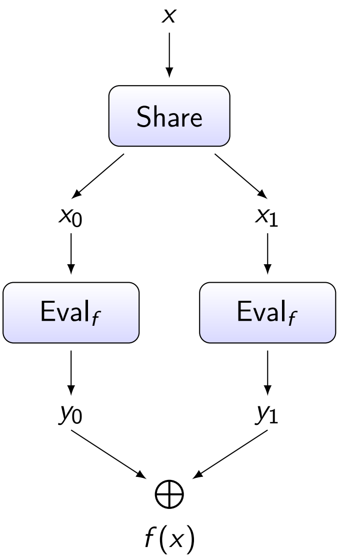

Security. $x$ remains hidden given $x_i$

Correctness. $\mathsf{Eval}_f(x_0) \oplus \mathsf{Eval}_f(x_1) = f(x)$

Efficiency. Running time $\mathsf{poly}(\lambda, |f|)$

Shamir: linear functions
[DHRW16]: all circuits, from MK-FHE

# Homomorphic Secret Sharing




**Security.** $x$ remains hidden given $x_i$

**$\delta$-Correctness.** $\mathsf{Eval}_f(x_0) \oplus \mathsf{Eval}_f(x_1) = f(x)$
Except with probability $\delta$

**Efficiency.** Running time $\mathsf{poly}(\lambda, |f|, 1/\delta)$

[BGI16]:

- DDH $\implies$ $\delta$-HSS for low-depth circuits
- $\delta-$HSS $\implies$ sublinear 2PC

# Our Results

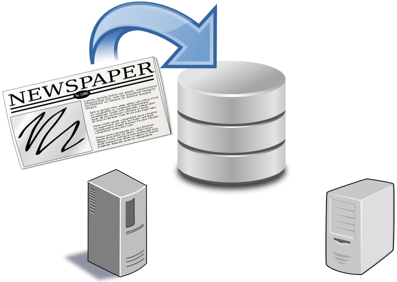
- **Optimizations**
  - Improved key generation
  - Smaller computation ($\times 30$)
  - Smaller share size ($\times 2$)
  - Leakage management techniques
  - Extensions
- **Applications**
  - Secure MPC with minimal interaction
  - Secure database access
  - Correlated randomness generation
- **Implementation**
- **... and more**

# Our Results

- **Optimizations**
  - Improved key generation
  - Smaller computation ($\times 30$)
  - Smaller share size ($\times 2$)
  - Leakage management techniques
  - Extensions
- **Applications**
  - Secure MPC with minimal interaction
  - Secure database access
  - Correlated randomness generation
- **Implementation**
- **… and more**

# Application: Secure Database Access

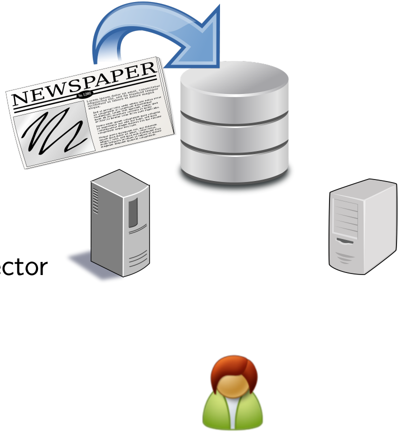
# Application: Secure Database Access



# Application: Secure Database Access

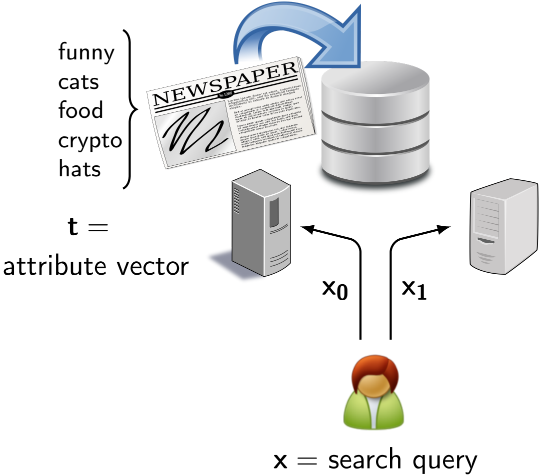
funny
cats
food
crypto
hats

$\mathbf{t} =$ attribute vector




$\mathbf{x} =$ search query

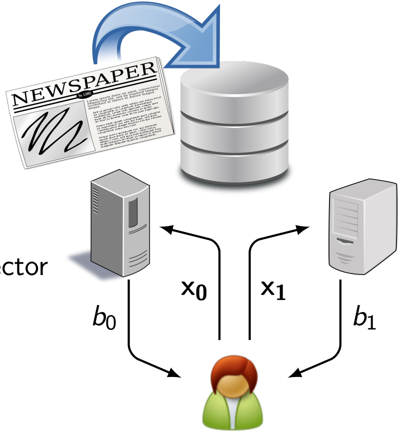
# Application: Secure Database Access



# Application: Secure Database Access

funny
cats
food
crypto
hats

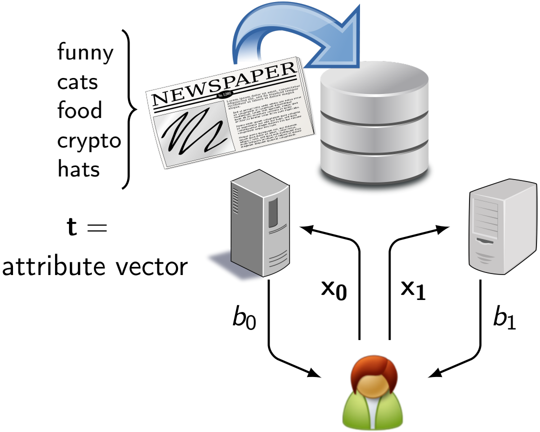
$\mathbf{t} =$ attribute vector




$\mathbf{x} =$ search query

$b_0 \oplus b_1 = f(\mathbf{x}, \mathbf{t})$

# Application: Secure Database Access




$\mathbf{x}$ = search query

$b_0 \oplus b_1 = f(\mathbf{x}, \mathbf{t})$

- Expressive matching functions
- Near optimal communication
- 100 gates: $< 0.1$s
- Handles access control

# Group-Based HSS

**RMS Programs:**

IT :

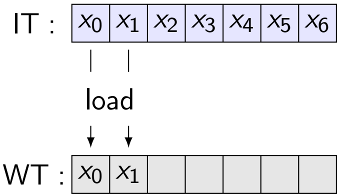
| | | | | | | |
|---|---|---|---|---|---|---|
| $x_0$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ |

WT :

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | |

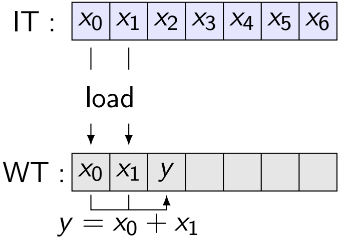
# Group-Based HSS

**RMS Programs:**



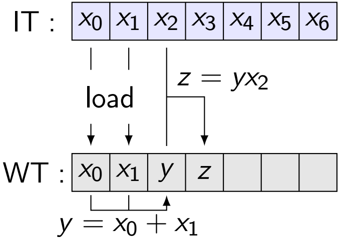
# Group-Based HSS

**RMS Programs:**



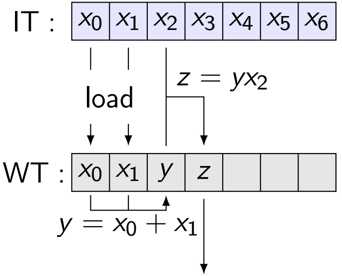
# Group-Based HSS

**RMS Programs:**



# Group-Based HSS

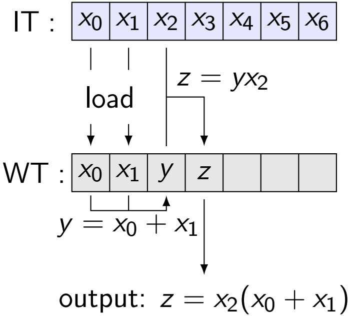
**RMS Programs:**




output: $z = x_2(x_0 + x_1)$

# Group-Based HSS

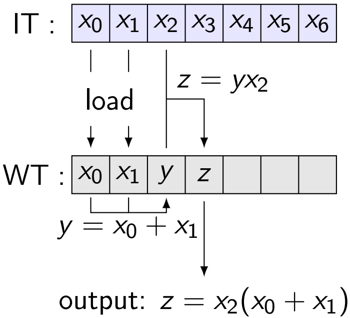
**RMS Programs:**




**Powerful model:**
LOGSPACE, $NC^1$, branching programs, formulas...

# Group-Based HSS

**RMS Programs:**




**Powerful model:**
LOGSPACE, $NC^1$, branching programs, formulas...

**Share Types:** Fix $(\mathbb{G}, g)$.

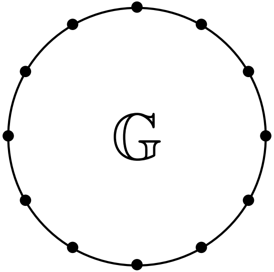
Level 1. $(g^x, g^x)$ ("encryption")
Level 2. $\langle x \rangle = (x_0, x_1)$, $x_0 + x_1 = x$.
Level 3. $\{x\} = (g_0, g_1)$, $g_0 = g_1 \cdot g^x$

**Operations on shares:**

Sum. $\langle x \rangle + \langle y \rangle = \langle x + y \rangle$
Product. $\mathrm{Pair}(g^x, \langle y \rangle) \mapsto \{xy\}$
Conversion. $\{xy\} \mapsto \langle xy \rangle$
$=$ distributed dlog w/ failure $\delta$

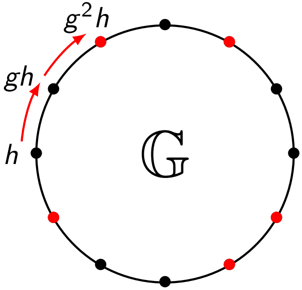
# Conversion Procedure

$(\mathbb{G}, g)$, parties have $(h_0, h_1 = g^x h_0)$, $x \in \{0, 1\}$



# Conversion Procedure

$(\mathbb{G}, g)$, parties have $(h_0, h_1 = g^x h_0)$, $x \in \{0, 1\}$




$\color{red}\bullet$: $\mathsf{PRF}(\bullet) = 0$

$\mathsf{Convert}(h) \mapsto$ smallest $i$ s.t. $\mathsf{PRF}(g^i h) = 0$

Failure probability = ratio of distinguished points $\approx 1/$(running time of Convert)

# Optimization: Fast Share Conversion

[BGI17]: Group $\mathbb{G}$ with generator $g$

$$h \qquad gh \qquad g^2 h \qquad g^3 h \qquad g^4 h \qquad \cdots$$

# Optimization: Fast Share Conversion

[BGI17]: Group $\mathbb{G}$ with generator $g$

stream : $\mathsf{msb}(h)$ $\mathsf{msb}(gh)$ $\mathsf{msb}(g^2 h)$ $\mathsf{msb}(g^3 h)$ $\mathsf{msb}(g^4 h)$ $\cdots$

$\rightarrow$ look for $0^d$ in stream

# Optimization: Fast Share Conversion

[BGI17]: Group $\mathbb{G}$ with generator $g$

$$\text{stream}: \quad \text{msb}(h) \quad \text{msb}(gh) \quad \text{msb}(g^2 h) \quad \text{msb}(g^3 h) \quad \text{msb}(g^4 h) \quad \cdots$$

$\longrightarrow$ look for $0^d$ in stream

**Conversion-friendly group:**

- $\mathbb{G} \subset \mathbb{F}_p$
- $p = 2^a - b$, $b$ small
- $g = 2$

Less than a machine word instruction per step!

# Optimization: Fast Share Conversion

[BGI17]: Group $\mathbb{G}$ with generator $g$

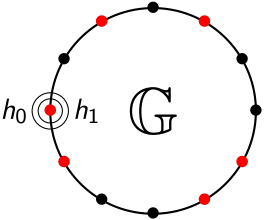
stream : $\quad \mathsf{msb}(h) \quad \mathsf{msb}(gh) \quad \mathsf{msb}(g^2h) \quad \mathsf{msb}(g^3h) \quad \mathsf{msb}(g^4h) \quad \cdots$

$\longrightarrow$ look for $0^d$ in stream

**Conversion-friendly group:**

- $\mathbb{G} \subset \mathbb{F}_p$
- $p = 2^a - b$, $b$ small
- $g = 2$

Less than a machine word instruction per step!




$100001011 \cdots$

$\Pr[P_0 \mid P_1] \geq 1/2$

$\Longrightarrow$ high correlation

# Optimization: Fast Share Conversion

[BGI17]: Group $\mathbb{G}$ with generator $g$

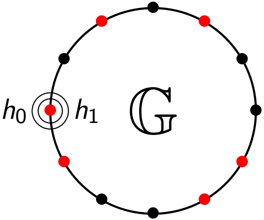
stream : $\quad \mathrm{msb}(h) \quad \mathrm{msb}(gh) \quad \mathrm{msb}(g^2 h) \quad \mathrm{msb}(g^3 h) \quad \mathrm{msb}(g^4 h) \quad \cdots$

$\longrightarrow$ look for $0^d$ in stream

**Conversion-friendly group:**

- $\mathbb{G} \subset \mathbb{F}_p$
- $p = 2^a - b$, $b$ small
- $g = 2$

Less than a machine word instruction per step!




$100001011\cdots$

$\Pr[P_0$ ⚑ $\mid P_1$ ⚑ $] \geq 1/2$

$\Longrightarrow$ high correlation

① separating distinguished points

# Optimization: Fast Share Conversion

[BGI17]: Group $\mathbb{G}$ with generator $g$

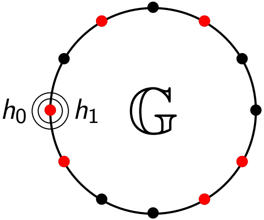
stream : $\mathrm{msb}(h)$ $\mathrm{msb}(gh)$ $\mathrm{msb}(g^2h)$ $\mathrm{msb}(g^3h)$ $\mathrm{msb}(g^4h)$ $\cdots$

$\longrightarrow$ look for $0^d$ in stream

**Conversion-friendly group:**

- $\mathbb{G} \subset \mathbb{F}_p$
- $p = 2^a - b$, $b$ small
- $g = 2$

Less than a machine word instruction per step!




$100001011\cdots$

$\Pr[P_0 \mid P_1 ] \geq 1/2$

$\Longrightarrow$ high correlation

---

(1) separating distinguished points

(2) looking for $10^d$ in stream $(\times 2)$

# Optimization: Fast Share Conversion

[BGI17]: Group $\mathbb{G}$ with generator $g$

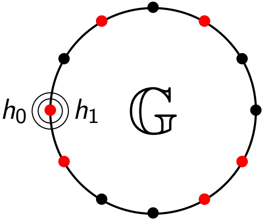
stream : $\mathrm{msb}(h)$ $\mathrm{msb}(gh)$ $\mathrm{msb}(g^2h)$ $\mathrm{msb}(g^3h)$ $\mathrm{msb}(g^4h)$ $\cdots$

$\rightarrow$ look for $0^d$ in stream

**Conversion-friendly group:**

- $\mathbb{G} \subset \mathbb{F}_p$
- $p = 2^a - b$, $b$ small
- $g = 2$

Less than a machine word instruction per step!




$100001011\cdots$

$\Pr[P_0 \mid P_1] \geq 1/2$

$\Longrightarrow$ high correlation

---

① separating distinguished points

② looking for $10^d$ in stream ($\times 2$)

③ randomizing conversions

# Optimization: Fast Share Conversion

[BGI17]: Group $\mathbb{G}$ with generator $g$

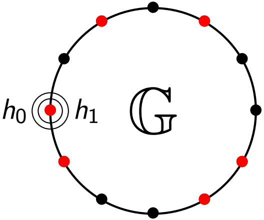
stream : msb($h$) msb($gh$) msb($g^2h$) msb($g^3h$) msb($g^4h$) $\cdots$

$\longrightarrow$ look for $0^d$ in stream

**Conversion-friendly group:**

- $\mathbb{G} \subset \mathbb{F}_p$
- $p = 2^a - b$, $b$ small
- $g = 2$

Less than a machine word instruction per step!




$1\overbrace{0\underbrace{00}0}1011\cdots$

$\Pr[P_0 \; | \; P_1 \;] \geq 1/2$

$\Longrightarrow$ high correlation

---

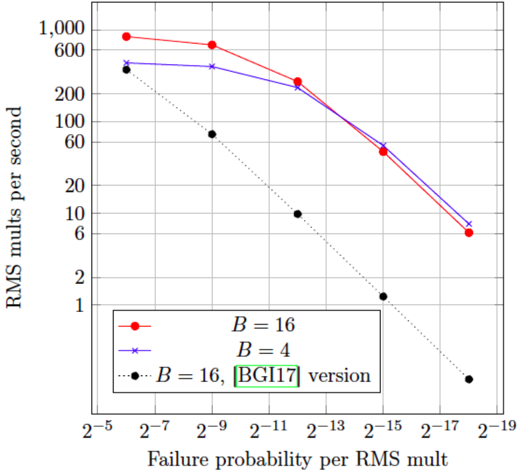
① separating distinguished points

② looking for $10^d$ in stream ($\times 2$)

③ randomizing conversions

④ average-case analysis ($\times 8$)

# Optimization: Fast Share Conversion




Implementation: `https://www.di.ens.fr/~orru/hss/`

# Optimization: Smaller Share Size

Level 1 share of $x$ = several ElGamal ciphertexts

---

**Idea 1:** randomness reuse $\rightarrow$ use $\mathrm{pk} = (g, h_0, \ldots, h_k)$

Factor 2 compression, but requires many secret keys

---

**Idea 2:** use a single key $c$, a public matrix $V = (\mathbf{v}_1 | \mathbf{v}_2 | \ldots | \mathbf{v}_k)$, and assume

$\{V, g, g^{c \bullet \mathbf{v}_1}, \ldots g^{c \bullet \mathbf{v}_k},\ \text{random } c\} \approx \{V, g, g^{d_1}, \ldots g^{d_k},\ \text{random } d_1, \ldots, d_k\}$

(Holds generically)

# Further Results

**Leakage Management:** Failures depends on inputs and secret key.

generate *leakage-absorbing pads* at key setup, use formulas for *masked evaluation*, e.g. $(\mathsf{EG}(x), \langle\!\langle b \rangle\!\rangle_c, \langle\!\langle x \oplus b \rangle\!\rangle_c) \mapsto \langle\!\langle xy \oplus b \rangle\!\rangle_c \rightarrow$ reduces leakage rate from $\delta$ to $O(\delta^2)$

---

**Extentions:** optimization for branching programs, threshold functions, degree-2 functions, extention to large outputs

---

**Further Applications:** MPC (e.g. voting), generating correlated randomness

# Further Results

**Leakage Management:** Failures depends on inputs and secret key.

generate *leakage-absorbing pads* at key setup, use formulas for *masked evaluation*, e.g. $(\mathsf{EG}(x), \langle\!\langle b \rangle\!\rangle_c, \langle\!\langle x \oplus b \rangle\!\rangle_c) \mapsto \langle\!\langle xy \oplus b \rangle\!\rangle_c \to$ reduces leakage rate from $\delta$ to $O(\delta^2)$

---

**Extentions:** optimization for branching programs, threshold functions, degree-2 functions, extention to large outputs

---

**Further Applications:** MPC (e.g. voting), generating correlated randomness

*Thank you for your attention*

Questions?