Achievable CCA2 Relaxation for Homomorphic Encryption

Adi Akavia  
Craig Gentry  
Shai Halevi  
Margarita Vald

University of Haifa, Israel  
TripleBlind, USA  
Algorand Foundation, USA  
Intuit Israel Inc., Israel
Secure outsourcing using Homomorphic Encryption (HE)

- Protects data in-use
- Low client complexity
- Deep computation is expensive – e.g., refreshing

\[ \text{Client} \quad \text{Enc}_{pk}(x) \quad \text{Server} \quad \text{Enc}_{pk}(f(x)) \]
Client-aided secure outsourcing using HE

- Protects data in-use
- Low client complexity
- Deep computation is expensive
  - refreshing by client, fast

Q: privacy against malicious servers?
Our Results I

on privacy against malicious server in client-aided protocols

**Insufficiency:** CPA-security \textcolor{red}{\textit{does not}} guarantee privacy against \textcolor{blue}{\textit{malicious}} servers.

**Define new notion — funcCPA,** and **prove** it is:

- \textcolor{green}{\checkmark} \textbf{Sufficient} for privacy against \textcolor{blue}{\textit{malicious}} servers,
- \textcolor{green}{\checkmark} \textbf{Achievable} from \textcolor{green}{\textbf{circuit privacy}}\textsuperscript{+}
  
  Moreover, known schemes can be transformed to \textcolor{green}{\textbf{circuit-private}}\textsuperscript{+}
Our Results II

Can we prove existing HE scheme are funcCPA-secure?

**Achievable:** leveled BV, BGV, … are leveled **funcCPA**-secure.

**Challenging:** **funcCPA** implies circular-security for (non-leveled) BV and BGV
Insufficiency of CPA: Our Attack (simplified)

**Theorem (Informal).** Exist* CPA-secure PKE ("assuming \(\exists\) CPA-secure PKE) so that client-aided outsourcing protocols instantiated with it are vulnerable to **input-recovery attack** by malicious servers.

**Proof Idea:** Starting from CPA-secure schemes, modify Enc, Dec as:

- **Enc’\(_{pk}(m)\):** If \(m=sk\), output \(0|m\)
  --test by checking whether \(Dec_m Enc_{pk}(r) = r\)
  Otherwise, output \(1|Enc_{pk}(m)\)

- **Dec’\(_{sk}(c’\):** Parse \(c’ = b|c\)
  If \(b=0\), output \(sk\)
  Otherwise, output \(Dec_{sk}(c)\)
Insufficiency of CPA: Our Attack (simplified)

Theorem (Informal). Exist* CPA-secure PKE ("assuming ∃ CPA-secure PKE) so that client-aided outsourcing protocols instantiated with it are vulnerable to input-recovery attack by malicious servers.

Proof Idea: Starting from CPA-secure schemes, modify Enc, Dec as:

◇ $\text{Enc'}_{pk}(m)$: If $m=sk$, output $0|m$
  --test by checking whether $\text{Dec}_m \text{Enc}_{pk}(r) = r$
  Otherwise, output $1|\text{Enc}_{pk}(m)$

◇ $\text{Dec'}_{sk}(c')$: Parse $c' = b|c$
  If $b=0$, output $sk$
  Otherwise, output $\text{Dec}_{sk}(c)$

Attacker sends ct = (0 | something) to be re-encrypted.
\textbf{funcCPA}-security: Definition & Sufficiency

\textbf{Informal.} \quad \textbf{funcCPA} extends CPA by supporting \textbf{Refresh} queries
\quad *more generally \text{Enc}(g(\text{Dec}(c)))

\textbf{Theorem (informal).} \quad \text{Client-aided protocols instantiated with a funcCPA-secure encryption guarantee privacy against malicious servers.}
Pictorially: **CPA**-security Definition

\[ \text{Challenger} \]

\[ (pk, sk) \leftarrow \text{Gen} \]

\[ b \leftarrow_R \{0,1\} \]

\[ c \leftarrow \text{Enc}_pk(m_b) \]

\[ \text{Adversary} \]

\[ pk \]

\[ m_0, m_1 \]

\[ c \]

\[ b' \]

**CPA-security:** \[ \forall \text{ppt adversary}, \quad \Pr[b'=b] \leq \frac{1}{2} + \text{negl} \]
Pictorially: **funcCPA**-security Definition

Challenger

\[(pk, sk) \leftarrow \text{Gen} \]

\[m \leftarrow \text{Dec}_{sk}(e)\]

\[e' \leftarrow \text{Enc}_{pk}(m)\]

\[b \leftarrow_R \{0, 1\}\]

\[c \leftarrow \text{Enc}_{pk}(m_b)\]

\[m \leftarrow \text{Dec}_{sk}(e)\]

\[e' \leftarrow \text{Enc}_{pk}(m)\]

**funcCPA-security:** \(\forall\) ppt adversary, \(\Pr[b'=b] \leq \frac{1}{2} + \text{negl}\)
Pictorially: **Leveled funcCPA** Definition

Queries are answered by next-level ciphertexts

Challenger

\[(pk_t, sk_t)_t \gets Gen\]

\[m \gets Dec_{sk_{current}}(e)\]

\[e' \gets Enc_{pk_{next}}(m)\]

\[b \leftarrow_R \{0, 1\}\]

\[c \leftarrow Enc_{pk_t}(m_b)\]

\[m \leftarrow Dec_{sk_{current}}(e)\]

\[e' \leftarrow Enc_{pk_{next}}(m)\]

Adversary

\[\{pk\}_t\]

\[ctxt e\]

\[ctxt e'\]

\[m_0, m_1, t\]

\[c\]

\[ctxt e\]

\[ctxt e'\]

\[b'\]

**leveled funcCPA-security:** \(\forall\) ppt adversary, \(\Pr[b'=b] \leq \frac{1}{2} + \text{negl}\)
Leveled \textbf{funcCPA}: Achievability by Existing Schemes

\textbf{Theorem.} Every \textbf{CPA}-secure \textbf{leveled HE} with \textbf{independent level keys} is \textbf{leveled funcCPA}-secure.

\textbf{Observation:} \textbf{BV, BGV, B/FV} (with a small modification) have independent level keys.

\textbf{Proof Idea.} Simulate answers to \textbf{funcCPA} queries by encryption of arbitrary message. Indistinguishable views by (\textbf{CPA}-security and) level keys independence.
**funcCPA**: Achievability from **Circuit-Privacy**

**Def (informal):** A HE scheme $E = (\text{Gen}, \text{Enc}, \text{Dec}, \text{Eval})$ is **circuit-private** if

$$\text{Eval}_{pk}(C; c_1, \ldots, c_t) \approx \text{Enc}_{pk}(C(\text{Dec}_{sk}(c_1), \ldots, \text{Dec}_{sk}(c_t)))$$

where:
- keys – properly generated
- ciphertexts – maliciously generated

**Prior defs for circuit-privacy:**

- **semi-honest:** both keys & ciphertexts – properly generated
- **malicious:** both keys & ciphertexts – maliciously generated
funcCPA: Achievability from Circuit-Privacy$^+$

Def (informal): A HE scheme $E=(\text{Gen, Enc, Dec, Eval})$ is circuit-private$^+$ if

$$\text{Eval}_{pk}(C; c_1, \ldots, c_t) \approx \text{Enc}_{pk}(C(\text{Dec}_{sk}(c_1), \ldots, \text{Dec}_{sk}(c_t)))$$

where: keys — properly generated
ciphertexts — maliciously generated

Theorem: Suppose $E$ is CPA-secure and circuit-private$^+$ w.r.t $C$,

Then $E$ is funcCPA w.r.t $C$.

Proof idea. Answer funcCPA queries using Eval. Indistinguishable by circuit-privacy$^+$
Construction: \textbf{Circuit-Privacy$^+$}

**Theorem:** Known HE schemes (e.g., BV and FHEW) can be transformed into \textit{circuit-private}$^+$. 

**Proof:**

\textbf{Idea 1.} \textbf{Sanitize} $^*$ \textbf{Enc} and \textbf{Eval} outputs to make them stat. close.

<table>
<thead>
<tr>
<th>Sanitization [DS16]:</th>
<th>If $\text{Dec}<em>{sk}(c_1) = \text{Dec}</em>{sk}(c_2)$</th>
<th>Then $\text{Sanitize}<em>{pk}(c_1) \approx_s \text{Sanitize}</em>{pk}(c_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(1)</td>
<td>(2)</td>
</tr>
</tbody>
</table>

\textbf{Problem:} \textbf{Eval} has \underline{no correctness} guarantee on malicious inputs ciphertexts (i.e., no (1) and hence no (2))

\textbf{Idea 2.} \textbf{Sanitize} also \underline{inputs} to \textbf{Eval} so, they are \underline{stat. close} to \underline{fresh} re-encryption (of some msg)
Conclusions

We propose new security notion – **funcCPA** – and show it is:

- Related to **circular-security**, though not known to be equivalent

- **Achievable:**
  1) via **generic** transformation
  2) for **existing** (leveled) schemes

- **Sufficient** for **privacy** in client-aided protocols against **malicious** servers

<table>
<thead>
<tr>
<th>Encryption</th>
<th>Type of client-aided protocol</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPA</td>
<td>w. natural property</td>
<td>semi-honest</td>
</tr>
<tr>
<td>leveled</td>
<td>next-level client’s response</td>
<td>malicious</td>
</tr>
<tr>
<td><strong>funcCPA</strong></td>
<td>all</td>
<td>malicious</td>
</tr>
</tbody>
</table>

**Open:** Prove that fully hom. BGV, B/FV... are **funcCPA**, assuming circular-security.