1 An Encryption Scheme with Zero-Knowledge Proof

For self-containment, we briefly recall the lifted ElGamal encryption scheme with the non-interactive zero-knowledge proof (NIZK) system which proves that the plaintext is either 0 or 1, proposed in [1], and provide some intuition for it. Let \( G \) be a group of prime order \( p \) such that the Decisional Diffie-Hellman assumption holds (i.e., the resulting encryption scheme provides a standard security property called semantic security). The public key is \((g, h, f)\) where \(g, f\) are random generators in \( G \) and \( h = g^z \) for \( z \) randomly picked from \( \mathbb{Z}_p = \{0, 1, \ldots, p - 1\} \). The secret key is \( z \). Let \( H \) be a hash function. Encrypting a plaintext \( b \in \{0, 1\} \) gives a ciphertext \((C_1 = g^u, C_2 = h^uf^b)\), for random \( u \in \mathbb{Z}_p \). Decryption computes \( y = C_2C_1^{-z} \) and outputs 1 if \( y = f \) or outputs 0 if \( y = 1 \).

- Encrypting a plaintext 0 outputs the ciphertext \((C_1, C_2)\) where
  \[
  C_1 = g^u, \quad C_2 = h^u
  \]
  where \( u \) is randomly picked from \( \mathbb{Z}_p \). The proof that its plaintext is either 0 or 1 consists of
  \[
  (s_0, t_0, s_1, t_1) \quad \text{where} \quad s_0 = r + t_0u \quad \text{and} \quad t_0 = H(C_1, C_2, g^r, h^r, g^{s_1-ut_1}, h^{s_1-ut_1}f^{t_1}) - t_1
  \]
  where \( r, s_1, t_1 \) are randomly picked from \( \mathbb{Z}_p \).

- Encrypting a plaintext 1 outputs the ciphertext \((C_1, C_2)\) where
  \[
  C_1 = g^u, \quad C_2 = h^uf
  \]
  where \( u \) are randomly picked from \( \mathbb{Z}_p \). The proof that its plaintext is either 0 or 1 consists of
  \[
  (s_0, t_0, s_1, t_1) \quad \text{where} \quad s_1 = r + t_1u \quad \text{and} \quad t_1 = H(C_1, C_2, g^{s_0-ut_0}, h^{s_0-ut_0}f^{-t_0}, g^r, h^r) - t_0
  \]
  where \( r, s_0, t_0 \) are randomly picked from \( \mathbb{Z}_p \).

The verification of the proof is done by computing

\[
X_1 = g^{s_0}C_1^{-t_0}, \quad X_2 = h^{s_0}C_2^{-t_0}, \quad Y_1 = g^{s_1}C_1^{-t_1}, \quad Y_2 = h^{s_1}C_2^{-t_1}f^{t_1},
\]

and checking whether

\[
t_0 + t_1 = H(C_1, C_2, X_1, X_2, Y_1, Y_2).
\]

Correctness can be verified straightforwardly. As for security, we provide the intuition as follows:

- **Soundness.** The relations for \( X_1, X_2 \) prove that the plaintext is 0, while the relations for \( Y_1, Y_2 \) prove that the plaintext is 1. The prover can produce exactly one real proof and one simulated fake proof due to the fact that the term \( t_0 + t_1 \) is constrained to the hash of \( r \). In case of encrypting a bit \( b \), the prover chooses \( t_b \) freely, where \( b \) is the flipped bit of \( b \). The fact that the prover cannot control \( t_b \) implies that it has to know the ciphertext randomness \( u \) in order to produce the correct \( s_b \) (à la the Fiat-Shamir Paradigm [2]). This implies that there exists \( u \) so that \((C_1, C_2)\)
forms an encryption of $b$, and hence the soundness. In other words, since the output of the hash function $H$ cannot be predicted before determining its input, in order to satisfy the condition $t_0 + t_1 = H(C_1, C_2, X_1, X_2, Y_1, Y_2)$, the only possible strategy is to choose at least one of $t_0$ and $t_1$ in a consistent manner after the input for $H$ is determined. Now note that

$$X_1 = g^{s_0 - u t_0}, X_2 = h^{s_0 - u t_0} f^{-b t_0}, Y_1 = g^{s_1 - u t_1}, Y_2 = h^{s_1 - u t_1} f^{(1-b) t_1}$$

if $(C_0, C_1)$ is a ciphertext of $b$. The shapes of $X_1$ and $X_2$, in particular the exponent $-b t_0$ of $f$ in $X_2$, imply that $t_0$ is uniquely determined from $X_1$ and $X_2$ unless $b = 0$. Similarly, the shapes of $Y_1$ and $Y_2$, in particular the exponent $(1-b) t_1$ of $f$ in $Y_2$, imply that $t_1$ is uniquely determined from $Y_1$ and $Y_2$ unless $b = 1$. Therefore, unless $b = 0$ or $b = 1$, there is no room to adjust the value of $t_0$ or $t_1$ after the input for $H$ is determined as required above. By the contraposition, the existence of a valid proof implies that the plaintext $b$ must be $0$ or $1$, ensuring the soundness.

- **Zero-Knowledge.** Observe that, in both cases, the proof elements $(s_0, t_0, s_1, t_1)$ distribute identically, due to the uniform randomness of $u, r, s_i, t_i$ (in case of encrypting $b$). Hence, the information on the plaintext bit is hidden from the proof.

## 2 Further security enhancement technique by using sign-preserving randomization

As mentioned at the end of the Database security enhancement techniques against regression attack section, we can further improve the output privacy by using “sign-preserving randomization” of the threshold Tversky index. Namely, the server calculates $r_1 \cdot \overline{T}(\vec{p}, \vec{q}) + r_2$ with random integers $0 \leq r_2 < r_1$ for each $\vec{p}$ and uses it instead of the true value $\overline{T}(\vec{p}, \vec{q})$ (the range of the dummy values $\phi_1, \ldots, \phi_n$ should also be changed accordingly). This modification makes it more difficult for the user to guess the true $\overline{T}(\vec{p}, \vec{q})$; it does not affect the correctness of the protocol, since $r_1 \cdot \overline{T}(\vec{p}, \vec{q}) + r_2$ has the same sign as $\overline{T}(\vec{p}, \vec{q})$. Therefore, it is qualitatively obvious that the output privacy is improved by the sign-preserving randomization. However, it is hard to quantitatively evaluate the achieved improvement; more detailed evaluations will be a future research topic.

## 3 Ranges of $\overline{T}$ for typical parameter settings

In the Parameter settings of the protocol section, we mentioned that the range of $\overline{T}$ is not too large for practical cases of chemical compound search. Here we list the ranges of typical parameter settings in Table S1. In our protocol, it is sufficient to check whether or not the $\overline{T}$ is positive value. Therefore, the range which is necessary to be verified is smaller in practical.

## 4 The attack algorithm by using a hit query

In this section, we showed the full algorithm to reveal the server’s fingerprint when a “hit” query is given, as mentioned in the Thresholding largely improves output privacy section.

Here we assume following case: The user is allowed to send more than one query and the database consists of a single fingerprint $p$. For each trial, the user learns from the server whether or not $\overline{T}(\vec{p}, \vec{q}) > 0$. Prior to his/her attack, the user is given a hit query $\vec{q}$ such that $\overline{T}(\vec{p}, \vec{q}) > 0$. The goal of the algorithm shown here is to reveal the server’s fingerprint $\vec{p}$. Here, we denote the reverse bit of $x$ as $\text{REV}(x)$.

**Theorem S1.** Let there be bit vectors $\vec{p}$ and $\vec{q}$ where $\overline{T}(\vec{p}, \vec{q}) > 0$ holds for $\vec{p}$ and $\vec{q}$, and a bit vector $\vec{q}$ which is defined by $\vec{q} = (q_1, \ldots, \text{REV}(q_1), \ldots, q_l)$. Then the inequality $\overline{T}(\vec{p}, \vec{q}) \geq \overline{T}(\vec{p}, \vec{q})$ holds if and only if $q_i = p_i$. 
Table S1. Range of $\overline{TI}$ for the most typical parameter settings.

<table>
<thead>
<tr>
<th>$p_i$</th>
<th>$q_i$</th>
<th>$\gamma$</th>
<th>$\theta$</th>
<th>$\overline{TI}(p, q)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>1.0</td>
<td>0.8</td>
<td>1.0</td>
<td>167</td>
</tr>
<tr>
<td>1.0</td>
<td>1.0</td>
<td>0.8</td>
<td>0.5</td>
<td>167</td>
</tr>
<tr>
<td>1.0</td>
<td>1.0</td>
<td>0.9</td>
<td>1.0</td>
<td>1</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>0.7</td>
<td>2.0</td>
<td>997</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>0.8</td>
<td>1.0</td>
<td>167</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>0.9</td>
<td>1.0</td>
<td>333</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>1.0</td>
<td>1.0</td>
<td>1</td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>0.7</td>
<td>1.0</td>
<td>499</td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>0.8</td>
<td>1.0</td>
<td>167</td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>0.9</td>
<td>1.0</td>
<td>167</td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>1</td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>0.7</td>
<td>0.5</td>
<td>167</td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>0.8</td>
<td>0.5</td>
<td>167</td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>0.9</td>
<td>0.5</td>
<td>167</td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>1</td>
</tr>
</tbody>
</table>

Proof. If $p_i = q_i = 1$, $|p \cap q| = |p \cap q| - 1$ and $|q| = |q| - 1$. Therefore $\overline{TI}(p, q) - \overline{TI}(p, q) = \lambda_1 - \lambda_3 = \gamma \theta g^{-1}(\theta^{-1} - 1 + \alpha) \geq 0$ from the definition of $\overline{TI}$ where $\alpha, \gamma, \theta_n, g^{-1} \geq 0$ and $1 \geq \theta > 0$.

If $p_i = q_i = 0$, $|q| = |q| + 1$. Therefore $\overline{TI}(p, q) - \overline{TI}(p, q) = \lambda_3 = \gamma \theta_n g^{-1} \beta \geq 0$ from the definition of $\overline{TI}$ where $\gamma, \theta_n, g^{-1}, \beta \geq 0$. \hfill $\square$

According to Theorem S1 and considering that $\theta > 0.5$ is used for similarity search, it is trivial that $\overline{TI}$ mostly keeps decreasing along with bits of $q$ being reversed sequentially and finally reaches a minus value. Therefore, we can find a bit vector $\hat{q}$ such that it satisfies

$$\overline{TI}(p, \hat{q}) > 0, \exists i \in \{1, \ldots, \ell\}; 0 > \overline{TI}(p, (\hat{q}_1, \ldots, \Rev(\hat{q}_i), \ldots, \hat{q}_\ell)),$$

and we call the bit vector “detective” query.

The inequality $\overline{TI}(p, q) > 0 > \overline{TI}(p, (\hat{q}_1, \ldots, \Rev(\hat{q}_i), \ldots, \hat{q}_\ell))$ holds if and only if $\hat{q}_i = p_i$ according to Theorem S1. Therefore the attacker knows whether or not $\hat{q}_i$ is equal to the database’s fingerprint $p_i$ from the observation of $\overline{TI}(p, (\hat{q}_1, \ldots, \Rev(\hat{q}_i), \ldots, \hat{q}_\ell))$. We show full description of the attack algorithm in Algorithm S1.
5 Derivation and calculation of upper bound of the probability for making at least one hit query

Here we provide a detailed derivation of the upper bound of (2) in the Thresholding largely improves database privacy section. In order to simplify the problem, the condition $\overline{TI}(p, q) > 0$ is replaced by the equal condition $1 - \theta \cdot |p \cap q| > \text{HD}(p, q)$ where $\text{HD}(p, q)$ is the Hamming distance of $p$ and $q$. For further simplification, we use the bound $\hat{f}_p \geq f_p$, derived from $|p| \geq |p \cap q|$, where

$$\hat{f}_p := \sum_q \Pr(Y = q) \cdot \Pr\left(1 - \frac{\theta}{\theta} \cdot |p| > \text{HD}(p, Y) \mid Y = q\right),$$

to obtain the following probability, which is the upper bound of the probability (2).

$$\sum_p \Pr(X = p) \cdot \left(1 - (1 - \hat{f}_p)^x\right) \quad (S1)$$

We use the known approximation $(1 - f_p)^x \simeq 1 - x \cdot f_p$ to avoid underflow for the computation. In this evaluation, we calculate the probability (S1) based on the assumption that both the user and the server create fingerprints by using the same model which generates each bit independently according to the same Bernoulli distribution with probability $w$ for the occurrence of the true bit. We notice that this simple assumption is a rather rough approximation to the real problem where bit generation is not always independent and $w$ is not equal for all the bits; however, we still think that this evaluation reflects an important tendency of the effect of thresholding and helps us to understand the output privacy of the proposed protocol.

Now, the probability (S1) and $\hat{f}_p$ are denoted by $w$ as follows.

$$\sum_{|p|=0}^{\ell} \binom{\ell}{|p|} w^{|p|} (1 - w)^{\ell - |p|} \cdot \left(1 - (1 - \hat{f}_p)^x\right).$$

$$\hat{f}_p = \sum_{j=0}^{\lfloor \frac{1 - \theta}{\theta} |p| \rfloor} \sum_{i=0}^{j} \binom{|p|}{i} \binom{\ell - |p|}{j - i} w^{|p| - 2i + j} (1 - w)^{\ell - |p| + 2i - j}.$$

For the calculation of the probability (S1) in the Thresholding largely improves database privacy section, we used $w = 0.28$ which is the average ratio for the occurrence of the true bit over all 166 MACCS keys stored in ChEMBL.
Algorithm S1 Full description of the attack algorithm by using a hit query

1: function Rev($x$)
2:     if $x = 1$ then
3:         $r \leftarrow 0$
4:     else
5:         $r \leftarrow 1$
6:     end if
7:     return $r$
8: end function
9:
10: Initialization: $q = \tilde{q}$, $O = \{1, 2, \ldots, \ell\}$.
11: while $\mathsf{Tl}(p, q) > 0$ do
12:     $i \in O$.
13:     $q_i \leftarrow \text{Rev}(q_i)$  \texttt{Updating $q$ for searching $\hat{q}$.}
14:     $O \leftarrow O \setminus \{i\}$.
15: end while
16: $\hat{q} = q$.
17: for $i = 1$ to $\ell$ do
18:     $q = \hat{q}$.
19:     $q_i \leftarrow \text{Rev}(\hat{q}_i)$
20:     if $\mathsf{Tl}(p, q) < 0$ then
21:         $r_i = \hat{q}_i$  \texttt{$p_i = \hat{q}_i$ is verified.}
22:     else
23:         $r_i = \text{Rev}(\hat{q}_i)$  \texttt{$p_i \neq \hat{q}_i$ is verified.}
24:     end if
25: end for
26: return $r$
6 Distribution of $w$ used in the experiments

In the Security analyses for padding dummies, we tested our method on four different distribution $w$. Here we list those three distributions. $w_i$ is shown in ascending order by $i$.

- $w_{\text{ChEMBL}}$: 
  - 0.00376525, 0.00352925, 0.00362674, 0.00342014, 0.00345187, 0.00335901, 0.00220916, 0.00213333, 0.00223779, 0.00249160, 0.00250398, 0.00265719, 0.00266493, 0.00291254, 0.00253957, 0.00258522, 0.00242196, 0.00237630, 0.00234845, 0.00238481, 0.00212327, 0.00216428, 0.00109336, 0.00102140, 0.000857357, 0.000885213, 0.000829501, 0.000877475, 0.000793132, 0.000735872, 0.000639922, 0.000667005, 0.000578019, 0.000667005, 0.000483617, 0.000435643, 0.000486713,
  - 0.00386894, 0.00395173, 0.00368942, 0.00352925, 0.00335901, 0.00304485, 0.00303705, 0.00291254, 0.00287562, 0.00253957, 0.00258522, 0.00242196, 0.00237630, 0.00234845, 0.00212327, 0.00216428, 0.00109336, 0.00102140, 0.000857357, 0.000885213, 0.000829501, 0.000877475, 0.000793132, 0.000735872, 0.000639922, 0.000667005, 0.000578019, 0.000667005, 0.000483617, 0.000435643, 0.000486713,
References
