Gadgets for Threshold AES: Correlation Robust Hash and Authenticated Garbling

Hongrui Cui        Chenkai Weng
09/28/2023
Correlation Robust Hash Functions
Overview

• Previous half-gates *implementation*.
  • Weakness.
  • Attack.

• A new design of correlation robust hash.
  • Concrete security.
  • Performance.
Half-gates: Garble an AND gate

\[ T_G = H(W_a^0, j) \oplus H(W_a^1, j) \oplus p_b R \]
\[ T_E = H(W_b^0, j') \oplus H(W_b^1, j') \oplus W_a^0 \]
Attack overview

• Exploit the weakness when $H()$ is instantiated by fixed-key AES.
  • $\pi$ modeled as a random permutation.

$$H(x, i) = \pi(2x \oplus i) \oplus 2x \oplus i.$$
Attack overview

• Exploit the weakness when \( H() \) is instantiated by fixed-key AES.
  • \( \pi \) modeled as a random permutation.
    \[
    H(x, i) = \pi(2x \oplus i) \oplus 2x \oplus i.
    \]

• Attacker succeed in running time \( O(2^k / C) \).
  • Circuit with \( k = 80 \) and \( C = 2^{40} \) would be completely broken.
  • Circuit with \( k = 128 \) and \( C = 2^{40} \) has only \( \sim 80 \) bit security.
  • Extend to multi-instance case: Attack is effective when \( C \) is the total size of multiple circuits.

\( k \): bit length of the labels  
\( C \): # of AND gates
Attack overview

- Implementation of the attack is consistent with analysis.

Interpolate:
When $k=80$, $C = 2^{30}$ attack needs 267 machine-months and $3500$. 

(a) Number of $\pi$ queries for the attack to succeed, on a log/log scale. 

(b) The running time of our attack with $C = 2^{30}$ and different values of $k$. 

Experiment
Analysis

Number of queries

Number of AND gates ($C$)

Running Time (ms)

Bit length of garbled labels ($k$)
Better concrete security

- \( \mathcal{M} \mathcal{M} \mathcal{O}^E \)
- \( \mathcal{O}_R^{\text{miTCCR}} \)
- Half-Gate

Hash function
Abstraction
Protocol
Abstraction

\[ \mathcal{O}_R^\text{mitTCCR} (w, i, b) \overset{\text{def}}{=} H(w \oplus R, i) \oplus b \cdot R \]

- Adversary is given \( u \) oracle instances.
- Never queries both \((w, i, 0), (w, i, 1)\).
- Same \( i \) is used at most \( \mu \) times.
The Hash Function

\[ \overline{M_{\text{MO}}^E(x, i)} \equiv E(i, \sigma(x)) \oplus \sigma(x) \]

- \( \sigma(x) \): a linear orthomorphism.
  - \( \sigma(x_L \parallel x_R) = x_R \oplus x_L \parallel x_L \)
- \( E \): modeled as an ideal cipher.
  - Key scheduling for each \( i \).
  - \( i \) starts at a random value.
Concrete Security

• Concrete security of Half-Gates.

\[ \varepsilon = \frac{\mu p}{2^{k-2}} + \frac{(\mu - 1)C}{2^{k-2}} + \frac{(2C)^{\mu+1}}{(\mu + 1)! \cdot 2^{\mu L}} \]

- **Computational security**
- **Statistical security**

\( \mu \): reuse of tweak \( i \).
\( p \): #queries to \( E \).
\( L \): in/output length of \( E \).

• Examples.

<table>
<thead>
<tr>
<th>( k ) (bit)</th>
<th>( C )</th>
<th>Comp. sec. (bit)</th>
<th>Sta. sec. (bit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>( \leq 2^{43.5} )</td>
<td>78</td>
<td>40</td>
</tr>
<tr>
<td>128</td>
<td>( \leq 2^{61} )</td>
<td>125</td>
<td>64</td>
</tr>
</tbody>
</table>
Implementation & optimization

- Performance with different hash functions

<table>
<thead>
<tr>
<th>Hash function</th>
<th>NI support?</th>
<th>$k$</th>
<th>Comp. sec. (bits)</th>
<th>100 Mbps</th>
<th>2 Gbps</th>
<th>localhost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zahur et al.</td>
<td>Y</td>
<td>128</td>
<td>89</td>
<td>0.4</td>
<td>7.8</td>
<td>23</td>
</tr>
<tr>
<td>SHA-3</td>
<td>N</td>
<td>128</td>
<td>125</td>
<td>0.27</td>
<td>0.27</td>
<td>0.28</td>
</tr>
<tr>
<td>SHA-256</td>
<td>N</td>
<td>128</td>
<td>125</td>
<td>0.4</td>
<td>1.1</td>
<td>1.2</td>
</tr>
<tr>
<td>SHA-256</td>
<td>Y</td>
<td>128</td>
<td>125</td>
<td>0.4</td>
<td>2.1</td>
<td>2.45</td>
</tr>
<tr>
<td>MMO$^E$</td>
<td>Y</td>
<td>128</td>
<td>125</td>
<td>0.4</td>
<td>7.8</td>
<td>15</td>
</tr>
<tr>
<td>MMO$^E$</td>
<td>Y</td>
<td>88</td>
<td>86</td>
<td>0.63</td>
<td>12</td>
<td>15</td>
</tr>
</tbody>
</table>

We optimized it to 20 since then
Extra Note

• Correlation robust hash function is also important to other MPC protocols, e.g. oblivious transfers.

Authenticated Garbling
Overview

• Semi-honest GC flaws against active adversary
  • Selective failure attack against privacy
  • Inconsistent circuit attack against correctness

• How to use authenticated garbling to fix those attacks
  • Selective Failure -> Distributed Garbling
  • Inconsistent Circuit -> IT-MAC Authentication

• Further improvements
Semi-honest Garbled Circuit

\[
\begin{array}{c|c|c|c}
    i & j & k \\
    
    0 & 0 & 0 \\
    0 & 1 & 0 \\
    1 & 0 & 0 \\
    1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c}
    \Lambda_i & \Lambda_j & \text{ciphertext} \\
    \lambda_i & \lambda_j & H(L_{i,\lambda_i}, L_{j,\lambda_j}) \oplus L_{k,\lambda_k} \\
    \lambda_i & \bar{\lambda}_j & H(L_{i,\lambda_i}, L_{j,\bar{\lambda}_j}) \oplus L_{k,\lambda_k} \\
    \bar{\lambda}_i & \lambda_j & H(L_{i,\bar{\lambda}_i}, L_{j,\lambda_j}) \oplus L_{k,\lambda_k} \\
    \bar{\lambda}_i & \bar{\lambda}_j & H(L_{i,\bar{\lambda}_i}, L_{j,\bar{\lambda}_j}) \oplus L_{k,\bar{\lambda}_k} \\
\end{array}
\]

\[
\begin{array}{c|c|c|c}
    \Lambda_i & \Lambda_j & \text{ciphertext} \\
    0 & 0 & H(L_{i,0}, L_{j,0}) \oplus L_{k,0} \oplus (\lambda_i \cdot \lambda_j \oplus \lambda_k) \Delta_A \\
    0 & 1 & H(L_{i,0}, L_{j,1}) \oplus L_{k,0} \oplus (\lambda_i \cdot \lambda_j \oplus \lambda_k) \Delta_A \\
    1 & 0 & H(L_{i,1}, L_{j,0}) \oplus L_{k,0} \oplus (\bar{\lambda}_i \cdot \lambda_j \oplus \lambda_k) \Delta_A \\
    1 & 1 & H(L_{i,1}, L_{j,1}) \oplus L_{k,0} \oplus (\bar{\lambda}_i \cdot \bar{\lambda}_j \oplus \lambda_k) \Delta_A \\
\end{array}
\]

- \( \Lambda_i = \lambda_i \oplus z_i \rightarrow \) Masked wire value
- \( \Delta_A \rightarrow \) Garbler’s key in Free-XOR
Security Issues against Active Adversaries

• Attack 1: Selective Failure
  • Suppose $P_B$ decrypts $$$ and failed
  • $P_A$ learns $z_i = \Lambda_i$, $z_j = \overline{\Lambda_j}$

• Attack 2: Circuit Logic Inconsistency
  • $P_A$ flips each AND gate output
  • AND -> NAND

\[
\begin{array}{c|c|c}
\Lambda_i & \Lambda_j & \text{ciphertext} \\
\hline
0 & 0 & H(L_{i,0}, L_{j,0}) \oplus L_{k,0} \oplus (\Lambda_i \cdot \Lambda_j \oplus \lambda_k)\Delta_A \\
0 & 1 & $$$ \\
1 & 0 & H(L_{i,1}, L_{j,0}) \oplus L_{k,0} \oplus (\overline{\Lambda_i} \cdot \Lambda_j \oplus \lambda_k)\Delta_A \\
1 & 1 & H(L_{i,1}, L_{j,1}) \oplus L_{k,0} \oplus (\overline{\Lambda_i} \cdot \overline{\Lambda_j} \oplus \lambda_k)\Delta_A \\
\end{array}
\]
Previous Solutions

• Cut-and-choose [LP07, NO09, HKE13, NST17…]
• $P_A$ prepares $\rho$ different garbled circuits/gates
• $P_B$ checks $\frac{\rho}{2}$ of them (by requesting random seeds)

To achieve statistical soundness of $2^{-\rho}$
$P_A$ needs to garble $\rho$ circuits 🙁
IT-MAC

• TinyOT-style bit authentication
• Open(x) → Sending \( [x, \ x] \)
• Opening to \( \bar{x} \) <-> Sending \( [x, \ \oplus \ \Delta] \) <-> Guessing \( \Delta \)

• Efficient Instantiation:
  • Base OT + Extension [IKNP03, KOS15, Roy22, ...]
  • COT PCG [BCGI18, BCGIKS19, YWLZW20, CRR21, RRT23...]

Authentication Equation
\[
\text{Sender} + x = \text{Receiver} + x + x \cdot \Delta
\]
Distributed Garbling

- $P_A$ needs to know $\lambda_i, \lambda_j$ to launch selective failure attack

- The attack fails if we share
  - $\lambda_i = a_i \oplus b_i$
  - $\lambda_j = a_j \oplus b_j$

$P_A$ can still garble if $b_i, b_j, b_k, \hat{b}_k$ are authenticated by

$$\lambda_i \cdot \lambda_j = \hat{a}_k \oplus \hat{b}_k$$

- $\lambda_i \cdot \lambda_j \oplus \lambda_k \cdot \Delta_A = (\hat{a}_k \oplus \hat{b}_k \oplus a_k \oplus b_k) \cdot \Delta_A$

<table>
<thead>
<tr>
<th>$\Lambda_i$</th>
<th>$\Lambda_j$</th>
<th>ciphertext</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$H(L_{i,0}, L_{j,0}) \oplus L_{k,0} \oplus (\lambda_i \cdot \lambda_j \oplus \lambda_k)\Delta_A$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$$$$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$H(L_{i,1}, L_{j,0}) \oplus L_{k,0} \oplus (\hat{\lambda}_i \cdot \lambda_j \oplus \lambda_k)\Delta_A$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$H(L_{i,1}, L_{j,1}) \oplus L_{k,0} \oplus (\hat{\lambda}_i \cdot \hat{\lambda}_j \oplus \lambda_k)\Delta_A$</td>
</tr>
</tbody>
</table>

$$b = b \oplus b \cdot \Delta_A$$

$b \in \{b_i, b_j, b_k, \hat{b}_k\}$
Consistency Checking

- $P_B$ wants to ensure that $(\lambda_i \oplus \Lambda_i) \cdot (\lambda_j \oplus \Lambda_j) = \lambda_k \oplus \Lambda_k$
- Use an additional AuthGC to let $P_B$ learn the correct $\Lambda_k$ [WRK17, DILO22]
- Add an additional round and let $P_B$ publish $\Lambda_i, \Lambda_j, \Lambda_k$

Linear relation on $\Delta_B$-authenticated values

$$\hat{a}_k \oplus \hat{b}_k \oplus \Lambda_j (a_i \oplus b_i) \oplus \Lambda_i (a_j \oplus b_j) \oplus \Lambda_i \Lambda_j = a_k \oplus b_k \oplus \Lambda_k$$
Preprocessing

- TinyOT-style protocol [NNOB12, WRK17, KRRW18]
- Ring-LPN based PCG [BCGIKS20]
Compressed Preprocessing

- Actually, $H_\infty(b)$ only needs to be $\tilde{O}(\rho)$-bit [DIL022]
- $b$ only prevents selective failure-resilience
- Together with efficient COTs, this brings constant amortized communication in preprocessing [CWXY23]

<table>
<thead>
<tr>
<th>2PC</th>
<th>Rounds</th>
<th>Communication per AND gate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prep.</td>
<td>Online</td>
</tr>
<tr>
<td>Half-gates</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>HSS-PCG</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>KRRW-PCG</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>DILO [18]</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>DILOv2 [18]</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>This work, v.1</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>This work, v.2</td>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>