Can LWC and PEC be Friends?:
Evaluating Lightweight Ciphers in Privacy-enhancing Cryptography

Kalikinkar Mandal\textsuperscript{1} and Guang Gong\textsuperscript{2}

\textsuperscript{1}Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3

\textsuperscript{2}Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, N2L 3G1, CANADA

October 20, 2020

NIST LWC Workshop 2020, Gaithersburg, USA
Outline

1. Applications

2. Binary Circuits of LWC Candidates

3. Secure Evaluation
   - 2PC computation
   - Homomorphic evaluation

4. Summary
Lightweight AE and Hash

**Protecting communication:** security and privacy of data in transit
- Confidentiality, Authenticity, Integrity

**Storage security:** encrypted data at rest
- Data security

**Enabling computation** over encrypted data: Privacy-enhancing cryptographic (PEC) techniques
- Secure multiparty computation (SMPC)
- Fully homomorphic encryption (FHE)
- Zero-knowledge Proofs (ZKPs)

**Friendly interface** between lightweight ciphers and PEC techniques
This protocol is from [MJSC16]

1 Analytics Outsourcing using FHE

\[(pk, sk) \leftarrow H\text{KeyGen}(1^\lambda)\]
\[K \leftarrow A\text{KeyGen}(1^\lambda)\]
\[C^h(K) \leftarrow H\text{Enc}(pk, K)\]
\[C^a(M_i) \leftarrow A\text{Enc}(K, M_i)\]

\[C^h(M_i) = \text{CTC}(\text{evk}, A\text{Dec}, C^h(K), C^a(M_i))\]

\[C^h(f(M_0, \cdots, M_{\ell-1})) = \text{HEval}(f, \text{evk}, \{C^h(M_i)\})\]
Sharing IoT data among servers: data safe, single point of failure

Settings:

\[(K_0, \cdots, K_{n-1}) \leftarrow Sh(K)\]

Dist. sym-decryption:

\[(m_0, \cdots, m_{n-1}) \leftarrow C^{dd}(\oplus K_i, c)\]

Naive solution:

\[(M_0, \cdots, M_{n-1}) \leftarrow Sh(M)\]

\[C_i \leftarrow AEnc(K, M_i)\]

\(n\) encryption

Dist. decryption:

\[C = AEnc(K, M)\]

1 encryption

Saving \(n - 1\) encryption
Fully homomorphic encryption (FHE) scheme [Gen09]

\[ FHE = (HKeyGen, HEnc, HDec, HEval) \]

**Key generation:** \((pk, sk, evk) \leftarrow HKeyGen(1^\lambda)\)

**Encryption:** \(c \leftarrow HEnc(pk, m)\)

**Decryption:** \(m \leftarrow HEnc(sk, c)\)

**Homomorphic evaluation:**

- **Add:** \(Add(HEnc(pk, m_1), HEnc(pk, m_2)) = HEnc(pk, m_1 + m_2)\)
- **Mul:** \(Mul(HEnc(pk, m_1), HEnc(pk, m_2)) = HEnc(pk, m_1 m_2)\)
- **Any function** \(f\)
  \[ HEnc(f(m_0, \cdots, m_{\ell-1})) \leftarrow HEval(evk, f, \{c_i\}_{i=0}^{\ell-1}) \]
Homomorphic Encryption Standard\textsuperscript{2}

- **HomomorphicEncryption.org** is an open consortium to standardize homomorphic encryption
  - industry, government, academia

- **Goals**
  - Unified and simplified API
  - Clear and understandable security properties

- **Some open-source homomorphic encryption schemes [1]**
  - HELib, Microsoft SEAL, PALISADE,
  - TFHE/FHEW, HeaAn, Λ ◦ λ, NFLlib, cuHE, Lattigo

\textsuperscript{2}https://homomorphicencryption.org
Garbler: Garbling a circuit

\[ E_{k_0}^0(E_{k_1}^0(k_4^0)) \]
\[ E_{k_0}^0(E_{k_1}^1(k_4^1)) \]
\[ E_{k_1}^1(E_{k_1}^0(k_4^1)) \]
\[ E_{k_0}^1(E_{k_1}^1(k_4^0)) \]

\[ E_{k_2}^0(E_{k_3}^0(k_5^0)) \]
\[ E_{k_2}^0(E_{k_3}^1(k_5^0)) \]
\[ E_{k_2}^1(E_{k_3}^0(k_5^1)) \]
\[ E_{k_2}^1(E_{k_3}^1(k_5^0)) \]

\[ E_{k_4}^0(E_{k_5}^0(k_6^0)) \]
\[ E_{k_4}^0(E_{k_5}^1(k_6^1)) \]
\[ E_{k_4}^1(E_{k_5}^0(k_6^1)) \]
\[ E_{k_4}^1(E_{k_5}^1(k_6^0)) \]

Evaluator: Evaluating a garbled circuit

- get correct input layer keys \((k_i^b)\) using oblivious transfer (OT)
- correctly decrypt each garbled gate
**GC Optimizations:** Permute-and-Point [BMR90], Row Reduction [NPS99], Free XOR [KS08], ..., Half Gates [ZRE15]

**Using Half Gates:**

\[
\begin{align*}
\begin{array}{ccc}
x_0 & \rightarrow & x_4 \\
x_1 & \rightarrow & x_4 \\
x_2 & \rightarrow & x_5 \\
x_3 & \rightarrow & x_5 \\
\end{array}
\end{align*}
\]

free \quad 2 \text{ ciphertexts} \quad \text{free}

\# AND in a circuit matters!
State-of-the-art: Sym-key & PEC

Stream ciphers for FHE

- FILP [MJSC16], Kreyvium [CCF+18], and Rasta [DEG+18]

Block ciphers for MPC & FHE

- LowMC [ARS+15], MiMC [AGR+16], GMiMC [AGP+19], and MARVELlous [AD18, AABS+20]

Hash functions for MPC & ZK Proofs

- GMiMC [AGP+19], and MARVELlous [AD18, AABS+20], Poseidon [GKR+19]
Our Results
Why binary circuits for lightweight ciphers?
- SMPC protocols, FHE schemes, ZK proofs

**AE core of an AE Scheme:** Nonlinear primitive
- Nonlinear primitive + Linear logic for mode
- Underlying permutation is the AE core of a sponge-based AE

Different AE cores in round 2 candidates
- 24 different AE cores
## Binary Circuits for LWC Candidates

<table>
<thead>
<tr>
<th>Candidates</th>
<th>Core-primitive</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACE</td>
<td>ACE</td>
<td>Permutation</td>
</tr>
<tr>
<td>COMET, ESTATE, mixFeed, SAEAES</td>
<td>AES</td>
<td>Block cipher, Tweakable Block cipher</td>
</tr>
<tr>
<td>ASCON, ISAP</td>
<td>ASCON</td>
<td>Permutation</td>
</tr>
<tr>
<td>COMET</td>
<td>CHAM</td>
<td>Block cipher</td>
</tr>
<tr>
<td>DryGASCON</td>
<td>DryGASCON</td>
<td>Permutation</td>
</tr>
<tr>
<td>ESTATE, GIFT-COFB, HyENA, LOTUS-AEAD, LOCUS-AEAD, SUNDAE-GIFT</td>
<td>GIFT-64/128</td>
<td>Block cipher, Tweakable block cipher</td>
</tr>
<tr>
<td>Gimli</td>
<td>GIMLI</td>
<td>Permutation</td>
</tr>
<tr>
<td>Grain-128AEAD</td>
<td>GRAIN</td>
<td>Stream cipher</td>
</tr>
<tr>
<td>ISAP, Elephant</td>
<td>KECCAK</td>
<td>Permutation</td>
</tr>
<tr>
<td>KNOT</td>
<td>KNOT</td>
<td>Permutation</td>
</tr>
<tr>
<td>ORANGE, PHOTON-Beetle</td>
<td>PHOTON</td>
<td>Permutation</td>
</tr>
<tr>
<td>Oribatida</td>
<td>SIMP-n-θ</td>
<td>Block cipher</td>
</tr>
<tr>
<td>Pyjamask</td>
<td>PYJAMASK</td>
<td>Block cipher</td>
</tr>
<tr>
<td>Saturn</td>
<td>SATURNIN</td>
<td>Block cipher</td>
</tr>
<tr>
<td>SPIX, SpoC</td>
<td>sLiSCP-LIGHT-192/256</td>
<td>Permutation</td>
</tr>
<tr>
<td>Sparkle</td>
<td>SPARKLE</td>
<td>Permutation</td>
</tr>
<tr>
<td>COMET</td>
<td>SPECK</td>
<td>Block cipher</td>
</tr>
<tr>
<td>Elephant</td>
<td>SPONGENT</td>
<td>Permutation</td>
</tr>
<tr>
<td>Spook</td>
<td>CLYDE-128 and SHADOW-512</td>
<td>Permutation</td>
</tr>
<tr>
<td>ForkAE, Romulus, SKINNY-AEAD</td>
<td>SKINNY</td>
<td>Block cipher</td>
</tr>
<tr>
<td>Subterranean</td>
<td>SUBTERRANEAN</td>
<td>Permutation</td>
</tr>
<tr>
<td>TinyJambu</td>
<td>TINYJAMBU</td>
<td>Stream cipher</td>
</tr>
<tr>
<td>WAGE</td>
<td>WAGE</td>
<td>Permutation</td>
</tr>
<tr>
<td>Xoodyak</td>
<td>XOODYAK</td>
<td>Permutation</td>
</tr>
</tbody>
</table>
Binary Circuit Stat of AE Cores

Generating Boolean circuits of AE cores
-Using the CBMC-GC compiler [FHK+14]

Gates:
XOR, AND, NOT

AND %
Photon: %7.02
Sparkle: %42.69
WAGE: %35.70

---

[Graph showing the distribution of gates and AND percentages for different algorithms]
Task 1: Secure evaluation of AE cores in MPC
- Securely evaluating the AE cores in the two-party computation
- Garbled circuit

Task 2: Homomorphic evaluation of AE cores
- Homomorphically evaluating the AE cores using a FHE scheme
- FHE works over binary circuits (e.g., TFHE)
Our Settings for AE Core Evaluations

Secure evaluation of core AE circuits

State: $S$  Key: $K$  Nonce: $N$

Garbler

$S = S_1 \oplus S_2$

$K = K_1 \oplus K_2$

$N = N_1 \oplus N_2$

Evaluator

Inputs:

$K_1, N_1$

$K_2, N_2$

$C$

Garbled circuit

$S_1 \oplus S_2 \leftarrow C(K_1 \oplus K_2, N_1 \oplus N_2)$

Outputs:

$S_1$

$S_2$
**Functionality:** Boolean sharing of message blocks

**Circuit** $C^{bms}(S, R_i, C_i)$:

1. $(S_r, S_c) = S \leftarrow C(S)$
2. $RK_i \leftarrow S_r \oplus R_i$
3. $M^1_i \leftarrow R_i$, $M^2_i \leftarrow C_i \oplus RK_i$
4. $M_i \leftarrow M^1_i \oplus M^2_i = R_i \oplus C_i \oplus S_r \oplus R_i$

**Input:** $R_i$

**Output:** $M^1_i$

**Common:** $S, C_i$

Diagram:
- **Input:** $R_i$
- **Output:** $M^1_i, M^2_i$
- **Common:** $S, C_i$
Boolean Message Sharing in Sponge

Init State: $S = K || N = S^1 \oplus S^2$, Ctxt: $C = (C_0, \cdots C_{\ell-1})$
Init State: $S = K || N = S^1 \oplus S^2$, Ctxt: $C = (C_0, \cdots C_{\ell - 1})$

Garbler

$$ (M^1_0, \cdots M^1_{\ell - 1}) $$

Input:

$S^1, R_0$

Output: $M^1_0 = R_0$

Garbled circuit + OT

Evaluator

$$ (M^2_0, \cdots M^2_{\ell - 1}) $$

$C^{\text{bms}}$

Input:

$S^2$

Output:

$M^2_0$

Garbled circuit + OT

$\vdots$

Input:

$R_{\ell - 1}$

Output: $M^1_{\ell - 1} = R_{\ell - 1}$

Garbled circuit + OT

$M^2_{\ell - 1}$
Experimental Evaluation

Settings:
- Considering 2-server settings
- Evaluating AE core circuits
- Use garbled circuit for 2PC computation

Implementation:
- Develop a generic implementation in C++ on top of the EMP-toolkit libraries [WMK16]
- Executing codes on a desktop with 3.00GHz Intel Coffee lake CPU, and 32 GB RAM running on Ubuntu 18.04
Timing for 2PC AE Cores Evaluation

![Graph showing timing for various algorithms in milliseconds.](image)
Homomorphic Evaluation of AE Cores

Initialize State: \((s_0, \cdots, s_{b-1}) = S = K\|N\)

Encrypted key: \(\text{HEnc}(K) = (\text{HEnc}(k_0), \cdots, \text{HEnc}(k_{n-1}))\)

\(C = \text{AE core circuit}\)

Homomorphically evaluating
Homomorphically Evaluating Sponge

State $S = (S_r, S_c)$  
Context $C = (C_0, \cdots), C_i = \text{AEnc}(K, M_i)$

Compute: $\text{HEnc}(M_i)$ from $C_i$ where $C_i = \text{AEnc}(K, M_i)$
Experimental Evaluation

Settings:
- Homomorphically evaluating AE core circuits
  - Encrypt input state
  - Obtain encrypted output state
- Use the TFHE scheme [CGGI16]
  - Efficient for Boolean operations such as XOR, AND and OR
  - Gate-level bootstrapping ⇒ circuit depth is not an issue

Implementation:
- Develop a generic implementation in C++ on top of the TFHE library
- Executing codes on a desktop with 3.00GHz Intel Coffee lake CPU, and 32 GB RAM running on Ubuntu 18.04
Time for AE Cores Eval using TFHE
Estimating Cost for Overall Mode

Estimating time for individual modes

- Using AE core evaluation cost + linear operation costs from micro-benchmark results

<table>
<thead>
<tr>
<th>Operation</th>
<th>Number of ciphertexts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>64</td>
</tr>
<tr>
<td>TFHE.Enc</td>
<td>0.002457 s</td>
</tr>
<tr>
<td>TFHE.Dec</td>
<td>0.000065 s</td>
</tr>
<tr>
<td>TFHE.XOR</td>
<td>2.502675 s</td>
</tr>
<tr>
<td>TFHE.AND</td>
<td>2.476224 s</td>
</tr>
<tr>
<td>TFHE.NOT</td>
<td>0.000058 s</td>
</tr>
</tbody>
</table>
Summary

• Secure evaluation of lightweight ciphers for privacy-enhancing cryptographic applications

• Optimized Boolean circuits of some core primitives of NIST LWC round 2 candidates

• Privacy-preserving evaluation of AE core circuits
  - 2PC evaluation: garbled circuit
  - FHE evaluation: TFHE

This work is in progress ...
Thanks for your attention!

Questions?