PALISADE is a lattice cryptography library developed since 2014. The development of PALISADE has been funded by DARPA, IARPA, Sloan Foundation, and NSA.

The key contributors and implementation partners include:
- Theoretical: Yuval Bitan, Shai Halevi, Craig Gentry, and Vinod Vaikuntanathan
- Industrial/Commercial/Institutional: Duality Technologies, New Jersey Institute of Technology, National University of Singapore, Sabanci University, etc.

Many schemes are supported, including:
- Homomorphic Encryption (HE): Brakerski/Pan-Vercauteren (3 variants) [1-3], Brakerski-Gentry-Vaikuntanathan [4], and Stehl-Steinfeld [5] schemes
- Proxy Re-Encryption for all HE schemes
- Digital Signature [6]
- Identity-Based Encryption [6]
- Ciphertext-Policy Attribute-Based Encryption [7]

PALISADE implements efficient Residue Number System (RNS) algorithms to achieve best practical performance results, e.g., PALISADE was used as the library for a winning genome-wide association studies solution at ISASH'18 [8].

Introduction

PALISADE implements efficient Residue Number System (RNS) algorithms to achieve best practical performance results, e.g., PALISADE was used as the library for a winning genome-wide association studies solution at ISASH'18 [8].

PALISADE and HE Standard

PALISADE implements the security settings recommended in Section 1.2 of the HomomorphicEncryption.org security standard.

One of the key goals of the PALISADE team is to develop a usable HE interface that a non-expert application developer can use to build applications with HE capabilities.

The PALISADE team has also been actively participating in IARPA's RAMPARTS and HECTOR programs building a framework that enables the development of a broad spectrum of secure distributed applications using homomorphic encryption.

Upcoming Releases

The PALISADE team is dedicated to improving the application developer experience when building applications with HE capabilities, and the next two releases are primarily focused on usability improvements.

Release v1.6 (early Q4 2019) includes the following improvements:
- Simplified cross-platform build process using CMake
- Much faster and simpler-to-use serialization
- Reduced footprint
- And Many More!

Release v1.7 (late Q4 2019) includes a more usable and efficient RNS variant of the Cheon-Kim-Kim-Song (CCKS) scheme.

The same API will be used for CCKS as for the other schemes.

Code Sample

Code Sample (Cont’d)

Key Concepts/Classes

<table>
<thead>
<tr>
<th>Cryptocontext wrapper class:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A wrapper that encapsulates the scheme, crypto parameters, encoding parameters, and keys.</td>
</tr>
<tr>
<td>Provides the same API for all HE schemes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ciphertext class:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stores the ciphertext polynomials.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Plaintext class:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stores the plaintext data (both raw and encoded).</td>
</tr>
<tr>
<td>Supports multiple encodings in a polymorphic manner, including PackedEncoding, IntegerEncoding, CoefPackedEncoding, etc.</td>
</tr>
</tbody>
</table>

Code Sample

```c
// Sample Program: CryptoContext

// Get the main parameters
int plaintextModulus = 4095;
int depth = 7;

// Instantiate the crypto context
CryptoContext CK = CryptoContextFactory::getCryptoContextFactory()
    .setSecurityLevel(512); // Security level/security level = 512/128 classes, uosdt.3-depth = 7;

// The same API will be used for CKKS as for the other schemes.

// Sample Program: CKKS Encryption

// The code samples are encrypted
auto ciphertext = encrypt(selectedPlaintext, selectedPublicKey, selectedPlaintext); // The same API will be used for CKKS-Encrypt.

// Sample Program: CKKS decryption

// The code samples are decrypted
auto plaintext = decrypt(ciphertext, selectedPublicKey); // The same API will be used for CKKS-Decrypt.

// Sample Program: CKKS keygen

// The code samples are encrypted
auto keypair = CKKS::KeyGen(); // The same API will be used for CKKS-Encrypt.
```

Contact

Yury Polyakov
Duality Technologies & NJIT
Email: ypolyakov@duality.cloud
Website: https://duality.cloud/

Kurt Rohloff
Duality Technologies & NJIT
Email: krohloff@duality.cloud
Website: https://duality.cloud/

References