Flexible Authenticated Encryption

Sanketh Menda, Julia Len, Viet Tung Hoang, Mihir Bellare, and Thomas Ristenpart

The Third NIST Workshop on Block Cipher Modes of Operation 2023
One scheme to rule them all?
One scheme to rule them all?

Fast on AES-NI servers
One scheme to rule them all?

???

- Fast on AES-NI servers
- Lightweight
One scheme to rule them all?

- Fast on AES-NI servers
- Lightweight

Incompatible
One scheme to rule them all?

- Fast on AES-NI servers
- Lightweight
- Nonce misuse resistant

Incompatible
One scheme to rule them all?

- Fast on AES-NI servers
- Nonce misuse resistant
- Lightweight
- Online

Incompatible
One scheme to rule them all?

- Fast on AES-NI servers
- Lightweight
- Nonce misuse resistant
- Online

Incompatible

Incompatible
One scheme to rule them all?

Fast on AES-NI servers
Nonce misuse resistant
Robust
Lightweight
Online

Incompatible
Incompatible
One scheme to rule them all?

- Fast on AES-NI servers
- Nonce misuse resistant
- Robust
- Lightweight
- Online
- Fast

Incompatible
One scheme to rule them all?

```plaintext

<table>
<thead>
<tr>
<th>Nonce misuse resistant</th>
<th>Fast on AES-NI servers</th>
<th>Lightweight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robust</td>
<td>Online</td>
<td>Fast</td>
</tr>
</tbody>
</table>

Incompatible

Incompatible

Incompatible
```
One scheme to rule them all?

- Fast on AES-NI servers
- Nonce misuse resistant
- Robust

- Lightweight
- Online
- Fast

Incompatible
## One scheme to rule them all?

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Incompatible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast on AES-NI servers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nonce misuse resistant</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robust</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lightweight</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Online</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fast</td>
<td>Incompatible</td>
<td></td>
</tr>
</tbody>
</table>

Applications are asking for more features.

But these features are incompatible.

We cannot build one scheme with all these features.
Lots of different schemes

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Target hardware</th>
<th>Misuse resistant</th>
<th>Robust</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES—GCM</td>
<td>AES-NI</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>AES-GCM-SIV</td>
<td>AES-NI</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Ascon</td>
<td>Lightweight</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>AES-AEZ</td>
<td>AES-NI</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
Lots of different schemes

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Target hardware</th>
<th>Misuse resistant</th>
<th>Robust</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES—GCM</td>
<td>AES-NI</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>AES-GCM-SIV</td>
<td>AES-NI</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Ascon</td>
<td>Lightweight</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>AES-AEZ</td>
<td>AES-NI</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Each of these schemes supports a different feature set.

Only getting more complicated. Ascon-SIV? Ascon-AEZ?

Up to developers to pick the most appropriate scheme.
Designing many different schemes scales poorly!

Subset of folders in https://github.com/openssl/openssl/tree/219bd6ac7061c40bd24f896f8652994d62d109de/crypto
Designing many different schemes scales poorly!

Libraries are going to get even more complicated.

Need to write a new standard for each new scheme.

Need to analyze each scheme independently.
Choosing an AEAD in BoringSSL

```cpp
auto aead = EVP_aead_aes_128_gcm();

// Or AES-GCM-SIV or XChaCha20/Poly1305 or CTR-HMAC or ...
auto ctx = EVP_AEAD_CTX_new(aead, key, tag_len);
EVP_AEAD_CTX_seal(ctx, out, nonce, in, ad); // Encryption
EVP_AEAD_CTX_open(ctx, out, nonce, in, ad); // Decryption
```
(Slightly simplified)
Goals for Flexible AEAD
Goals for Flexible AEAD

A. Minimize library complexity.
Goals for Flexible AEAD

A. Minimize library complexity.

B. Simplify analysis.
Goals for Flexible AEAD

A. Minimize library complexity.

B. Simplify analysis.

C. Easy-to-use APIs.
Real world AEAD implementations
Real world AEAD implementations

Application -> Library

- AES-GCM
- AES-GCM-SIV
- Ascon
- AES-AEZ

Library ->

- GHASH
- AES
- Ascon

...
Real world AEAD implementations

Applications use libraries not standalone schemes.

Applications use many AEAD schemes, not just the one scheme.

Libraries implement schemes using common components.
Formalizing real world AEAD implementations
Formalizing real world AEAD implementations

- Reduce the number of components.
- Modularly analyze the components.
- Misuse resistant APIs and safe defaults.
Choosing an AEAD with Flexible AEAD

```c
auto config = { mr: true, rob: false, hardware: aes-ni };
auto aead = aead_from_config(config);
auto ctx = EVP_AEAD_CTX_new(aead, key, tag_len);
EVP_AEAD_CTX_seal(ctx, out, nonce, in, ad); // Encryption
EVP_AEAD_CTX_open(ctx, out, nonce, in, ad); // Decryption
(Slightly simplified, reimagined from BoringSSL)
```
## What is a configuration?

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Value 1</th>
<th>Value 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>config.nonce_hiding</td>
<td>✔️</td>
<td>❌</td>
</tr>
<tr>
<td>config.misuse_resistance</td>
<td>✔️</td>
<td>❌</td>
</tr>
<tr>
<td>config.key_length</td>
<td>128</td>
<td>256</td>
</tr>
<tr>
<td>config.target_hardware</td>
<td>AES-NI</td>
<td>Lightweight</td>
</tr>
</tbody>
</table>
What is a configuration?

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Value 1</th>
<th>Value 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>config.nonce_hiding</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>config.misuse_resistance</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>config.key_length</td>
<td>128</td>
<td>256</td>
</tr>
<tr>
<td>config.target_hardware</td>
<td>AES-NI</td>
<td>Lightweight</td>
</tr>
</tbody>
</table>

- Encodes desired features as a dictionary.
- Default to safer choices.
- Tooling to generate configs.
- Tooling to verify configs.
Implementing configurations

Session key encodes the config and a CR-KDF of the key, config, and session AD.

Supports session ADs by default.

Can safely reuse key material across configs and sessions.

The session key is not exportable.
Gracefully handles broken configs and leaked keys

- **config 1**
  - CR-KDF
  - Session Key 1
    - subkey 1
  - **Wrong**

- **config 2**
  - CR-KDF
  - Session Key 2
    - subkey 2
  - **Correct**
Gracefully handles broken configs and leaked keys

- config 1 -> CR-KDF -> Session Key 1 (subkey 1)
- config 2 -> CR-KDF -> Session Key 2 (subkey 2)

- config 1 -> CR-KDF -> Session Key 1 (subkey 1)
- config 2 -> CR-KDF -> Session Key 2 (subkey 2)
Public permutations: a natural starting point

We can build all of symmetric cryptography from a permutation.
Public permutations: a natural starting point

We can build all of symmetric cryptography from a permutation.

SHA3 and Ascon are based on permutations.
Public permutations: a natural starting point

We can build all of symmetric cryptography from a permutation.

SHA3 and Ascon are based on permutations.

Recent work on building permutations using AES-NI instructions.
CIV: SIV-inspired MR context committing AEAD

Associated Data

Message

Build PRF, CR-PRF, and CTR from an underlying permutation.

CR-PRF

Subkey1

Context committing and nonce misuse resistant.

PRF

Subkey2

CTR-Encrypt

Fast, competitive with AES-based misuse resistant schemes.

Synthetic IV

Tag

Ciphertext

Subkey3
OCH: OCB3-inspired NMR context committing AEAD

- Nonce
- $E_{K1}[0]$ → Ctxt[1]
- $E_{K1}[1,N]$ → Ctxt[2]
- $E_{K1}[2,N]$ → Ctxt[3]
- Checksum
- CR-PRF$_{K2}$ → Tag
- AD

- Build TBC and CR-PRF from an underlying permutation.
- Context committing and nonce hiding.
- Fast, competitive with AES-based AEAD schemes.

Next Steps

Our Suggestion

What do y’all think of this?

Robust AEAD? Compactly committing AEAD?

API design improvements? Cross-language challenges?

Reach out!

snkth.com