Summary

- first and only adhoc tweakable AES-128 ...

- ... which allows us to provide $2^{128}$ guarantee for both integrity and forgery - no birthday security!

- extremely fast in software, on par with OCB3 for long messages

- fast for short messages - minimal overhead as no initialization is needed

- quite small in hardware

- parallelizable

- very simple - almost direct plug-in of AES-128 (reuse existing security analysis and implementations), backward compatible with AES-128

- we provide a nonce-misuse resistant mode if needed
1. **Description of KIASU**
   - Operating mode(s)
   - The tweakable block cipher KIASU-BC

2. **Security**

3. **Performances and Features**

4. **Conclusion**
KIASU≠, KIASU= and KIASU-BC

We have two operating modes KIASU≠ and KIASU=, both built upon the same tweakable block cipher named KIASU-BC.

**Operating modes:**
- KIASU≠ is for nonce-respecting (based on OCB3)
- KIASU= is for nonce-misuse resistance (based on COPA)
- both modes are parallelizable

**The tweakable block cipher KIASU-BC:**
- message of \( n = 128 \) bits
- key of \( k = 128 \) bits
- tweak of \( t = 64 \) bits
Outline

1. **Description of KIASU**
   - Operating mode(s)
   - The tweakable block cipher KIASU-BC

2. **Security**

3. **Performances and Features**

4. **Conclusion**
**nonce-respecting mode:** KIASU≠

KIASU≠ is based on OCB3

For Associated Data (full block):

\[
\begin{align*}
A_1 & \rightarrow E^2_{K,N,1} \\
A_2 & \rightarrow E^2_{K,N,2} \\
\vdots & \rightarrow E^2_{K,N,l_a} \\
0 & \rightarrow + \\
\end{align*}
\]

For Associated Data (partial block):

\[
\begin{align*}
A_1 & \rightarrow E^2_{K,N,1} \\
A_2 & \rightarrow E^2_{K,N,2} \\
\vdots & \rightarrow E^2_{K,N,l_a} \\
0 & \rightarrow + \\
A*10* & \rightarrow E^6_{K,N,l_a} \\
\end{align*}
\]
**nonce-respecting mode:** \( \text{KIASU} \neq \)

\( \text{KIASU} \neq \) is based on \( \text{OCB3} \)

For Plaintext (full block):

\[
\begin{align*}
M_1 \quad & \rightarrow \quad E_K^{0,N,1} \quad \rightarrow \quad C_1 \\
M_2 \quad & \rightarrow \quad E_K^{0,N,2} \quad \rightarrow \quad C_2 \\
& \quad \cdots \quad \cdots \\
M_l \quad & \rightarrow \quad E_K^{0,N,l} \quad \rightarrow \quad C_l \\
\sum \quad & \rightarrow \quad E_K^{1,N,l} \quad \rightarrow \quad \text{tag} \\
\end{align*}
\]

For Plaintext (partial block):

\[
\begin{align*}
M_1 \quad & \rightarrow \quad E_K^{0,N,1} \quad \rightarrow \quad C_1 \\
M_2 \quad & \rightarrow \quad E_K^{0,N,2} \quad \rightarrow \quad C_2 \\
& \quad \cdots \quad \cdots \\
M_l \quad & \rightarrow \quad E_K^{0,N,l} \quad \rightarrow \quad C_l \\
M \ast 10^n \quad & \rightarrow \quad E_K^{4,N,l} \quad \rightarrow \quad \text{pad} \quad \rightarrow \quad C_* \\
\sum \quad & \rightarrow \quad E_K^{5,N,l} \quad \rightarrow \quad \text{tag} \\
\end{align*}
\]
nonce-misuse resistant mode: \texttt{KIASU}=

\texttt{KIASU} is based on COPA

For Associated Data (full block):

\begin{align*}
A_1 & \xrightarrow{E^2_{\text{N},1}} \ldots \xrightarrow{E^2_{\text{N},l_a-1}} A_{l_a-1} \\
0 & \xrightarrow{\oplus} \ldots \xrightarrow{\oplus} \text{Auth}
\end{align*}

For Associated Data (partial block):

\begin{align*}
A_1 & \xrightarrow{E^2_{\text{N},1}} \ldots \xrightarrow{E^2_{\text{N},l_a}} A_{l_a} \\
A_{l_a} & \xrightarrow{E^7_{\text{N},l_a}} \text{Auth}
\end{align*}
nonce-misuse resistant mode: KIASU

KIASU is based on COPA

For Plaintext (full block):

\[
\begin{align*}
&M_1 \\
&E_{K}^{0,N,1} \\
&\text{Auth} \\
&E_{K}^{4,N,1} \\
&C_1 \\
&M_{l-1} \\
&E_{K}^{0,N,l-1} \\
&M_l \\
&E_{K}^{0,N,l} \\
&M_l \oplus E_{K}^{0,N,l} \\
&\Sigma \\
&E_{K}^{1,N,l} \\
&E_{K}^{4,N,l-1} \\
&E_{K}^{4,N,l} \\
&E_{K}^{4,N,l-1} \oplus E_{K}^{4,N,l} \\
&E_{K}^{5,N,l} \\
&\text{final} \\
&\text{tag}
\end{align*}
\]
**nonce-misuse resistant mode:** \( \text{KIASU=} \)

\( \text{KIASU=} \) is based on COPA

For Plaintext (single partial block):

\[
\begin{align*}
M \cdot 10^* & \downarrow \\
E_K^{0,N,0} & \downarrow \\
E_K^{1,N,0} & \downarrow \\
\text{Auth} & \downarrow \\
E_K^{4,N,0} & \downarrow \\
E_K^{5,N,0} & \downarrow \\
C' & \downarrow \\
\text{Final'} & \downarrow \\
C_* & \downarrow \downarrow \\
\tag & \downarrow \\
\end{align*}
\]

For Plaintext (partial block):

\[
\begin{align*}
M_1 & \downarrow \\
E_K^{0,N,1} & \downarrow \\
\cdots & \downarrow \\
E_K^{0,N,l-1} & \downarrow \\
E_K^{0,N,l} & \downarrow \\
E_K^{1,N,l} & \downarrow \\
\Sigma & \downarrow \\
E_K^{4,N,l} & \downarrow \\
\cdots & \downarrow \\
E_K^{4,N,l-1} & \downarrow \\
E_K^{4,N,l} & \downarrow \\
\text{Final} & \downarrow \\
M_* & \downarrow \\
C_* & \downarrow \\
\tag & \downarrow \\
\end{align*}
\]

\( XLS \)
Outline

1 Description of KIASU
   ▶ Operating mode(s)
   ▶ The tweakable block cipher KIASU-BC

2 Security

3 Performances and Features

4 Conclusion
The tweakable block cipher KIASU-BC

KIASU-BC is exactly the AES-128 cipher, but with a fixed 64-bit tweak value $T$ XORed to each subkey (on the two first rows).

TWEAKKEY framework (see next presentation and AsiaCrypt 2014)
The tweakable block cipher KIASU-BC

KIASU-BC is exactly the AES-128 cipher, but with a fixed 64-bit tweak value $T$ XORed to each subkey (on the two first rows).

**TWEAKEY framework** (see next presentation and AsiaCrypt 2014)
1. Description of KIASU
   ▶ Operating mode(s)
   ▶ The tweakable block cipher KIASU-BC

2. Security

3. Performances and Features

4. Conclusion
## Security claims (in $\log_2$)

<table>
<thead>
<tr>
<th></th>
<th>Security (bits)</th>
<th>Security (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>nonce-respecting user</strong></td>
<td>KIASU ≠</td>
<td>KIASU =</td>
</tr>
<tr>
<td>Confidentiality for the plaintext</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td>Integrity for the plaintext</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td>Integrity for the associated data</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td><strong>nonce-misuse user</strong></td>
<td>KIASU ≠</td>
<td>KIASU =</td>
</tr>
<tr>
<td>Confidentiality for the plaintext</td>
<td>none</td>
<td>64</td>
</tr>
<tr>
<td>Integrity for the plaintext</td>
<td>none</td>
<td>64</td>
</tr>
<tr>
<td>Integrity for the associated data</td>
<td>none</td>
<td>64</td>
</tr>
</tbody>
</table>
## Conjectured security claims (in $\log_2$)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>KIASU$\neq$</td>
<td>KIASU$=$</td>
<td></td>
</tr>
<tr>
<td>Confidentiality</td>
<td>128</td>
<td>128</td>
<td>none</td>
</tr>
<tr>
<td>for the plaintext</td>
<td>128</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td>Integrity</td>
<td>128</td>
<td>128</td>
<td>none</td>
</tr>
<tr>
<td>for the plaintext</td>
<td>128</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td>Integrity for the</td>
<td>128</td>
<td>128</td>
<td>none</td>
</tr>
<tr>
<td>associated data</td>
<td>128</td>
<td>128</td>
<td>64</td>
</tr>
</tbody>
</table>
Security of KIASU-BC

The security of KIASU-BC is the same as AES-128 for a fixed tweak. The tricky part is to analyse what happens when the tweak varies.

If the key is fixed and one varies the tweak:

KIASU-BC’s tweak schedule has been chosen such that it is itself a good key schedule.

Bad idea: adding a tweak on the entire 128-bit state, since trivial and very good related-tweak differential paths would exist.

If both the key and tweak vary:

KIASU-BC was designed such that no interesting interaction between the key schedule and the tweak schedule will exist. We put a special focus on attacks which are highly impacted by the key schedule:

- related-key related-tweak attacks
- meet-in-the-middle attacks
Security of KIASU-BC

related-key related-tweak attacks

We prove that no good related-key related-tweak attacks differential path exist for KIASU (even boomerang), with a computer-aided search tool.

<table>
<thead>
<tr>
<th>rounds</th>
<th>active SBoxes</th>
<th>upper bound on probability</th>
<th>method used</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>0</td>
<td>$2^0$</td>
<td>trivial</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>$2^{-6}$</td>
<td>Matsui’s</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>$2^{-48}$</td>
<td>Matsui’s</td>
</tr>
<tr>
<td>5</td>
<td>$\geq 14$</td>
<td>$2^{-84}$</td>
<td>Matsui’s</td>
</tr>
<tr>
<td>7</td>
<td>$\geq 22$</td>
<td>$2^{-132}$</td>
<td>ex. split (3R+4R)</td>
</tr>
</tbody>
</table>
Security proofs on operating modes

When the nonce is not reused, we ensure that every call to KIASU-BC will have a distinct tweak input value.

We can directly reuse the OCB3 and COPA operating modes security proofs.

- but we can ensure full 128-bit security
- the proofs are simpler (see OCB3 and OCB3 proofs)

Universal hash based tweakable block ciphers won’t provide full 128-bit security (or with bad efficiency), due to the possibility of collisions between the inputs/outputs of the internal block cipher.
Outline

1 Description of KIASU
   ▶ Operating mode(s)
   ▶ The tweakable block cipher KIASU-BC

2 Security

3 Performances and Features

4 Conclusion
Measuring authenticated encryption speed

One should consider several scenarios when measuring speed for AE:

- $K_{\Delta}N_{\Delta}$: when key and nonce are random  
  (what SUPERCOP is currently measuring?)
- $K_{\Delta}N_{+}$: when key is random, but nonce is counter
- $K_{-}N_{\Delta}$: when key is fixed, but nonce is random
- $K_{-}N_{+}$: when key is fixed, and nonce is counter
- $K_{-}N_{-}$: when both key and nonce are fixed  
  (for nonce-misuse resistant schemes)

It would be great to measure all these 5 cases in SUPERCOP to get a better picture (probably $K_{\Delta}N_{\Delta}$ and $K_{\Delta}N_{+}$ are very similar)

When people present speed results, they should make clear in which of these 5 cases they made the measurements.

KIASU is rather neutral with regards to the first 4 cases (having $K_{-}N_{\Delta}$ or $K_{-}N_{+}$ makes no difference)
Software performances (in c/B) - case $K_\Delta N_\Delta$

Both KIASU\(\neq\) and KIASU\(=\) can be parallelized

<table>
<thead>
<tr>
<th>KIASU(\neq)</th>
<th>512B</th>
<th>1024B</th>
<th>4096B</th>
<th>65536B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Haswell</td>
<td>1.37</td>
<td>1.04</td>
<td>0.80</td>
<td>0.72</td>
</tr>
<tr>
<td>Intel Sandy Bridge</td>
<td>2.05</td>
<td>1.61</td>
<td>1.15</td>
<td>0.99</td>
</tr>
<tr>
<td>Intel Haswell (no AES-NI)</td>
<td>19.31</td>
<td>13.47</td>
<td>9.08</td>
<td>7.71</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>KIASU(=)</th>
<th>512B</th>
<th>1024B</th>
<th>4096B</th>
<th>65536B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Haswell</td>
<td>2.32</td>
<td>1.88</td>
<td>1.59</td>
<td>1.49</td>
</tr>
<tr>
<td>Intel Sandy Bridge</td>
<td>3.79</td>
<td>3.13</td>
<td>2.55</td>
<td>2.06</td>
</tr>
<tr>
<td>Intel Haswell (no AES-NI)</td>
<td>26.77</td>
<td>20.91</td>
<td>16.61</td>
<td>15.22</td>
</tr>
</tbody>
</table>
Software performances (in c/B) - Fast on small messages

KIASU is fast for small messages, as it requires no initialization.

- sponge-like designs require strong initialization, AES-GCM-like designs usually prepare computation tables
- "simple IMIX" is a weighted average simulating sizes of typical IP packages:
  7 parts of 40B, 4 parts of 576B, 1 part of 1500B
- maximum transmission unit (MTU) for Ethernet is 1500 bytes

<table>
<thead>
<tr>
<th></th>
<th>40B</th>
<th>576B</th>
<th>1500B</th>
<th>IMIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>KIASU≠</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel Haswell</td>
<td>9.45</td>
<td>1.31</td>
<td>0.96</td>
<td>1.74</td>
</tr>
<tr>
<td>Intel Sandy Bridge</td>
<td>10.85</td>
<td>2.01</td>
<td>1.51</td>
<td>2.43</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>40B</th>
<th>576B</th>
<th>1500B</th>
<th>IMIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>KIASU=</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel Haswell</td>
<td>25.03</td>
<td>2.30</td>
<td>2.30</td>
<td>3.86</td>
</tr>
<tr>
<td>Intel Sandy Bridge</td>
<td>31.53</td>
<td>3.54</td>
<td>3.64</td>
<td>5.50</td>
</tr>
</tbody>
</table>
Hardware performances

- easy to reuse existing tricks from AES-128 FPGA/ASIC implementations
- save implementation and area cost if co-implemented with AES-128
- being fast for small messages is very valuable, as small messages is a typical use-case of hardware applications

For FPGA (ongoing work):

- Marc Stöttinger and He Wei from NTU worked on a first (not yet optimized) round-based FPGA implementation of KIASU-BC
- 1989 slices (neither internal BRAM nor external RAM) for 1.08Gbit/s throughput on a Virtex-5 FPGA

For ASIC (ongoing work):

- we estimate that KIASU-BC can be implemented with 3000GE (reusing smallest known AES-128 implementation - 2400 GE)
- we estimate that one has to add an extra 1000 GE for implementing KIASU≠, and 2000 GE for KIASU=
Others features

**KIASU-BC is backward compatible with AES-128**: simply set $T = 0$. This will save implementation overheads.

**KIASU will perform well on many platforms, even legacy ones**, since it is very close to AES-128. This might not be true for candidates that perform multiplications in a big Galois field.

**Tweakable block ciphers are very useful building blocks**:

- block cipher, stream cipher
- parallel MAC
- parallel authenticated encryption: like OCB3 or COPA, but simpler design/proofs and much higher security bounds
- hash function: use the tweak input as block counter (HAIFA framework) or to perform randomized hashing
- tree hashing: use the tweak to encode the position in the tree
- PRNG, KDF, disk encryption
Outline

1. Description of KIASU
   - Operating mode(s)
   - The tweakable block cipher KIASU-BC

2. Security

3. Performances and Features

4. Conclusion
KIASU-BC is the first AES-based ad-hoc tweakable block cipher

KIASU:

- ✔ faster than AES-GCM: extremely fast in software, especially for the message sizes that matter
- ✔ smaller than AES-GCM: good hardware profile
- ✔ more versatile than AES-GCM: good performances in any platform
- ✔ much higher security than AES-GCM: full 128-bit security
- ✔ much simpler than AES-GCM: simple design and proofs
- ✔ more features than AES-GCM: can easily switch to a nonce-misuse resistant mode
- ✔ parallelizable
Thank you!