Subtle Authenticated Encryption
Achieving AE despite Deterministic Decryption Leakage

Guy Barwell  Dan Page  Martijn Stam

Department of Computer Science, University of Bristol

Autumn 2015
Outline

1. Security for the Real World
   - Authenticated Encryption
   - Extending the Security Framework
   - SAE

2. Comparison of Strengthened AE notions
   - BDPS
   - RUP
   - RAE[\tau]

3. Conclusions
   - Conclusion
1 Security for the Real World
   ■ Authenticated Encryption
   ■ Extending the Security Framework
   ■ SAE

2 Comparison of Strengthened AE notions

3 Conclusions
Authenticated Encryption

Two parties share a key and want to communicate “securely”

- Their messages should be *private* and *authentic*
- An adversary wants to stop them doing this

\[ C = E_k^{N,A}(M) \]
Authenticated Encryption

- Two parties share a key and want to communicate “securely”
- Their messages should be *private* and *authentic*
- An adversary wants to stop them doing this
Authenticated Encryption

- Two parties share a key and want to communicate “securely”
- Their messages should be *private* and *authentic*
- An adversary wants to stop them doing this
## Authenticated Encryption

### Goals
- Learn something about the content of a message
- Send a message that was not intended

### Powers
- Some sort of oracle access they’ve discovered/created
- Eg request encryptions or decryptions
## Authenticated Encryption

### Goals
What does the adversary want to do?

- Distinguish encryptions from random
- Distinguish real decryption from one that always rejects

### Powers
What can they do to help them achieve this?

- Make queries to an honest encryption oracle
- Make queries to an honest decryption oracle
Authenticated Encryption: Syntax

An Authenticated Encryption scheme is a pair of algorithms

\[ E : K \times N \times A \times M \rightarrow C \]
\[ D : K \times N \times A \times C \rightarrow M \cup \{ \bot \} \]

Where:
- \( K \) Key space
- \( N \) Nonce space
- \( A \) Associated Data
- \( M \) Message Space
- \( C \) Ciphertext Space
- \( \bot \) Invalid ciphertext symbol
Authenticated Encryption: Syntax

An Authenticated Encryption scheme is a pair of algorithms

\[
\mathcal{E} : K \times N \times A \times M \rightarrow C \\
\mathcal{D} : K \times N \times A \times C \rightarrow M \cup \{\bot\}
\]

Where:
- \(K\) Key space
- \(N\) Nonce space
- \(A\) Associated Data
- \(M\) Message Space
- \(C\) Ciphertext Space
- \(\bot\) Invalid ciphertext symbol
## Authenticated Encryption

<table>
<thead>
<tr>
<th>Goals</th>
<th>Powers</th>
</tr>
</thead>
<tbody>
<tr>
<td>What does the adversary want to do?</td>
<td>What can they do to help them achieve this?</td>
</tr>
<tr>
<td>Distinguish encryptions from random</td>
<td>Make queries to an honest encryption oracle</td>
</tr>
<tr>
<td>Distinguish real decryption from one that always rejects</td>
<td>Make queries to an honest decryption oracle</td>
</tr>
</tbody>
</table>
Authenticated Encryption

**Goals**

What does the adversary want to do?

<table>
<thead>
<tr>
<th>Enc</th>
<th>$\mathcal{E}_k$</th>
<th>?</th>
<th>$\mathcal{D}_k$</th>
<th>Dec</th>
</tr>
</thead>
</table>

**Powers**

What can they do to help them achieve this?

<table>
<thead>
<tr>
<th>Enc</th>
<th>$\mathcal{E}_k$</th>
<th>?</th>
<th>$\mathcal{D}_k$</th>
<th>Dec</th>
</tr>
</thead>
</table>

Reference world is *ideal* rather than *attainable*.
We can immediately recover the recognised notions:

- IND$–CPA is our IND–CPA
- INT–CTXT is our CTI–CCA
- AE (CCA3) is our AE—PASS
A piecewise name scheme for AE notions

We can immediately recover the recognised notions:

- IND$^+$–CPA is our IND–CPA
- INT–CTXT is our CTI–CCA
- AE (CCA3) is our AE—PASS
A piecewise name scheme for AE notions

We can immediately recover the recognised notions:

- IND$–CPA is our IND–CPA
- INT–CTXT is our CTI–CCA
- AE (CCA3) is our AE—PASS
### A piecewise name scheme for AE notions

<table>
<thead>
<tr>
<th>Enc</th>
<th>Dec</th>
<th>AE</th>
<th>CCA</th>
<th>CPA</th>
<th>CDA</th>
<th>PAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enc</td>
<td></td>
<td>IND</td>
<td>$\mathcal{E}_k$</td>
<td>$\mathcal{E}_k$</td>
<td>$\mathcal{D}_k$</td>
<td>$\mathcal{D}_k$</td>
</tr>
<tr>
<td></td>
<td>Dec</td>
<td>CTI</td>
<td></td>
<td>$\mathcal{D}_k$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We can immediately recover the recognised notions:

- IND$\$–CPA is our IND–CPA
- INT–CTXT is our CTI–CCA
- AE (CCA3) is our AE—PASS
Decryption Leakage

Decryption is not ideal

In the real world, not all rejections are the same: The adversary may discover some extra information...

e.g.:
- Timing
- Error Codes
- Unsecured buffers (e.g. candidate/encoded plaintexts)
Decryption Leakage

Decryption is not ideal

In the real world, not all rejections are the same: The adversary may discover some extra information...

e.g.:

- Timing
- Error Codes
- Unsecured buffers (eg candidate/encoded plaintexts)
Decryption Leakage

Decryption is not ideal

In the real world, not all rejections are the same: The adversary may discover some leakage

e.g.: Timing, Error codes, temporary buffers, ...

We will assume that:

- Only invalid decryption queries leak.
- Leakage is a deterministic function of its inputs.
Decryption Leakage

Decryption is not ideal

In the real world, not all rejections are the same: The adversary may discover some leakage

e.g.: Timing, Error codes, temporary buffers, . . .

We will assume that:

- Only invalid decryption queries leak.
- Leakage is a deterministic function of its inputs.
Modelling Decryption Leakage

So, our leakage functions looks like:

\[ \Lambda : K \times N \times A \times C \rightarrow \{\top\} \cup L \]

(Where an output of \( \top \) corresponds to a valid message)
Modelling Decryption Leakage

So, our leakage functions looks like:

\[ \Lambda : K \times N \times A \times C \to \{ \top \} \cup L \]

(Where an output of \( \top \) corresponds to a valid message)
Modelling Decryption Leakage

So, our leakage functions looks like:

\[ \Lambda : K \times N \times A \times C \rightarrow \{\top\} \cup L \]

(Where an output of \( \top \) corresponds to a valid message)
Thus our oracles have the syntax:

\[
\begin{align*}
\text{Enc, } \mathcal{E} & : K \times N \times A \times M \rightarrow C \\
\text{Dec, } \mathcal{D} & : K \times N \times A \times C \rightarrow M \cup \{\bot\} \\
\Lambda & : K \times N \times A \times C \rightarrow \{\top\} \cup L
\end{align*}
\]

The adversary will be given access to (some subset of):

\[\text{Enc, Dec, } \mathcal{E}_k, \mathcal{D}_k, \Lambda_k\]
Oracles

Thus our oracles have the syntax:

\[
\begin{align*}
\text{Enc, } \mathcal{E} &: K \times N \times A \times M \rightarrow C \\
\text{Dec, } \mathcal{D} &: K \times N \times A \times C \rightarrow M \cup \{⊥\} \\
\Lambda &: K \times N \times A \times C \rightarrow \{\top\} \cup L
\end{align*}
\]

The adversary will be given access to (some subset of):

\[
\begin{align*}
\text{Enc} & \quad \text{Dec} \\
\mathcal{E}_k & \quad \mathcal{D}_k \\
\Lambda_k
\end{align*}
\]
Oracles

Thus our oracles have the syntax:

\[
\begin{align*}
\text{Enc}, \mathcal{E} & : K \times N \times A \times M \rightarrow C \\
\text{Dec}, \mathcal{D} & : K \times N \times A \times C \rightarrow M \cup \{\perp\} \\
\Lambda & : K \times N \times A \times C \rightarrow \{\top\} \cup L
\end{align*}
\]

The adversary will be given access to (some subset of):

\[
\begin{align*}
\mathcal{E}_k \quad \mathcal{D}_k \\
\text{Enc} \quad \text{Dec} \\
$ \quad \perp \\
\end{align*}
\]
Oracles

Thus our oracles have the syntax:

\[
\begin{align*}
\text{Enc}, \mathcal{E} & : K \times N \times A \times M \rightarrow C \\
\text{Dec}, \mathcal{D} & : K \times N \times A \times C \rightarrow M \cup \{\perp\} \\
\Lambda & : K \times N \times A \times C \rightarrow \{\top\} \cup L
\end{align*}
\]

The adversary will be given access to (some subset of):

\[
\begin{align*}
\text{Enc} & \quad \text{Dec} & \quad \mathcal{E}_k & \quad \mathcal{D}_k & \quad \Lambda_k
\end{align*}
\]

We extend our \textit{power} terminology with the addition of an \textit{s} for \textit{subtle}
Disallowed Queries

Encrypt: \( M \rightarrow C \)

Decrypt: \( C \rightarrow M \cup \{\perp\} \)

Leakage: \( C \rightarrow \{\top\} \cup L \)

Key:
- \( \rightarrow \) Prohibited Queries
- \( \rightarrow \) Superfluous Queries
- \( \leftrightarrow \) Entangled Oracles

An arrow \( A \rightarrow B \) means that queries made to \( A \) restrict queries to \( B \). Arrows within the same row mean inputs cannot be repeated, those from one row to another mean the output of \( A \) cannot later be used as input to \( B \).
Effective Games

So, there are a total of $24 = 3 \times 2^3$ security games, some of which are equivalent:

- AE–sCCA
- AE–sCPA
- AE–sCDA
- AE–sPAS
- AE—CCA
- AE—CPA
- AE—CDA
- AE—PAS
- IND–sCCA
- IND–sCPA
- IND–sCDA
- IND–sPAS
- IND—CCA
- IND—CPA
- IND—CDA
- IND—PAS
- CTI–sCCA
- CTI–sCPA
- CTI–sCDA
- CTI–sPAS
- CTI—CCA
- CTI—CPA
- CTI—CDA
- CTI—PAS

Effective Games

So, there are a total of $24 = 3 \times 2^3$ security games, some of which are equivalent:

- AE–sCCA
- AE—CCA
- IND–sCCA
- IND—CCA
- CTI–sCCA
- CTI—CCA
- AE–sCPA
- AE—CPA
- IND–sCPA
- IND—CPA
- CTI–sCPA
- CTI—CPA
- AE–sCDA
- AE—CDA
- IND–sCDA
- IND—CDA
- CTI–sCDA
- CTI—CDA
- AE–sPAS
- AE—PAS
- IND–sPAS
- IND—PAS
- CTI–sPAS
- CTI—PAS

Effective Games

So, there are a total of $24 = 3 \times 2^3$ security games, some of which are equivalent:

- AE–sPAS
- AE—PAS
- IND–sCDA
- IND—sPAS
- IND—CDA
- IND—PAS
- CTI–sCPA
- CTI—sPAS
- CTI—CPA
- CTI—PAS

Effective Games

So, there are a total of $24 = 3 \times 2^3$ security games, some of which are equivalent:

SAE: Subtle Authenticated Encryption

\[ SAE := \text{AE–sCCA} \]

- Name inspired by WebCryptoAPI
- Security depends on subtleties of implementation
- Simulator Free: \((\mathcal{E}, \mathcal{D}, \Lambda)\) defines the scheme
- Reduces to AE-sPAS
Error Simulatability: A means not an end

“Leakage should not give out useful information”

A new goal: Error Simulatability
Error Simulatability: A means not an end

"Leakage should not give out useful information"

A new goal: Error Simulatability

\[
\Lambda_k \\
? \\
\Lambda_l \\
\text{ERR}
\]
Error Simulatability: A means not an end

“Leakage should not give out useful information”

For example: ERR–PAS

\[
\Lambda_k \\
? \\
\Lambda_l \\
\text{ERR}
\]
Error Simulatability: A means not an end

Error Simulatability

“ Leakage should not give out useful information”

For example: ERR–CCA

\[
\Lambda_k \\
? \\
\Lambda_l
\]

\[
\text{ERR} \quad E_k \quad D_k
\]
Decomposing SAE

SAE decomposes in an intuitive manner

\[ \text{SAE} \iff \text{ERR–CCA} + \text{CTI–CPA} + \text{IND–CPA} \]
Decomposing SAE

SAE decomposes in an intuitive manner

\[ \text{SAE} \iff \text{ERR–CCA} + \text{CTI–CPA} + \text{IND–CPA} \]

Diagram:

- \(E_k\) connected to \(D_k\) and \(\Lambda_k\)
- \(E_k\) connected to \(\Lambda_l\) and \(\Lambda_l\)
- \(\Lambda_l\) connected to \(\Lambda_l\) and \(\Lambda_k\)
- \(\Lambda_k\) connected to \(\Lambda_k\) and \(\Lambda_k\)

SAE (as AE–sPAS)
Comparison of Strengthened AE notions

1. Security for the Real World

2. Comparison of Strengthened AE notions
   - BDPS
   - RUP
   - RAE[τ]

3. Conclusions
## Syntactic Choices

<table>
<thead>
<tr>
<th>$C = \mathcal{E}_k(M)$</th>
<th>$\mathcal{D}_k$</th>
<th>$\Lambda_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M \in M$</td>
<td>$\bot$</td>
<td>$\bot_i \in L$</td>
</tr>
<tr>
<td>$c \in C \setminus \text{im}(\mathcal{E}_k)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **BDPS**: $L$, $M$ disjoint
- **RUP**: $L = M$, add $V$
- **RAE$[\tau]$**: $L$, $M$ disjoint
Syntactic Choices

\[
\begin{array}{c|ccc}
 & D_k & \Lambda_k \\
\hline 
C = \mathcal{E}_k(M) & M \in M & T \\
c \in C \setminus \text{im}(\mathcal{E}_k) & \perp & \perp_i \in L \\
D_k & \\
\end{array}
\]

- BDPS: L, M disjoint
- RUP: L = M, add V
- RAE[\tau]: L, M disjoint
BDPS: Distinguishable Decryption Failures

- Relaxed the assumption that all decryption errors were identical
- Gave definitions, relations and separations in the Probabilistic & random-IV models
- Nonce-based analogues of their definitions and relations

- Error-tolerance definition INV–ERR roughly says “only one error code is likely”

On Symmetric Encryption with Distinguishable Decryption Failures

Boldyreva, Degabriele, Paterson & Stam; FSE 2013
Comparison with past works

<table>
<thead>
<tr>
<th>Our Notion</th>
<th>BDPS Notion</th>
</tr>
</thead>
<tbody>
<tr>
<td>IND–CPA</td>
<td>IND$–CPA</td>
</tr>
<tr>
<td>IND–sCCA</td>
<td>IND$–CCA</td>
</tr>
<tr>
<td>IND–sCPA</td>
<td>IND$–CVA</td>
</tr>
<tr>
<td>CTI–CPA</td>
<td>INT–CTXT*</td>
</tr>
<tr>
<td>CTI–sCPA</td>
<td>INT–CTXT</td>
</tr>
<tr>
<td>AE</td>
<td>INT–CTXT</td>
</tr>
<tr>
<td>SAE</td>
<td>≈IND$–CCA3</td>
</tr>
</tbody>
</table>
RUP: Release of Unverified Plaintext

- Nonce-based definitions, relations and separations.
- Provisioned for the leakage of a candidate plaintext.
- Models Decrypt-then-authenticate (e.g., MtE, M&E).
- Observes that if $\Lambda_k$ can be simulated, then $\Lambda$. does so.

- Key definitions are simulator based.
- Does not allow for any other leakage.

How To Securely Release Unverified Plaintext in Authenticated Encryption

Andreeva, Bogdanov, Luykx, Mennink, Mouha & Yasuda; AC 2014
Syntactic Choices

\[ C = \mathcal{E}_k(M) \]
\[ c \in C \setminus \text{im}(\mathcal{E}_k) \]

<table>
<thead>
<tr>
<th>( \mathcal{D}_k )</th>
<th>( \Lambda_k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( M \in M )</td>
<td>( \top )</td>
</tr>
<tr>
<td>( \bot )</td>
<td>( \bot_i \in L )</td>
</tr>
<tr>
<td>( D_k )</td>
<td></td>
</tr>
</tbody>
</table>

- BDPS: \( L, M \) disjoint
- RUP: \( L = M, \) add \( V \)
- RAE[\( \tau \)]: \( L, M \) disjoint
Syntactic Choices

\[ C = \mathcal{E}_k(M) \]

\[ c \in C \setminus \text{im}(\mathcal{E}_k) \]

- **BDPS**: \( L, M \) disjoint
- **RUP**: \( L = M \), add \( V \)
- **RAE[\tau]**: \( L, M \) disjoint
RUP: Release of Unverified Plaintext

- Authenticity definitions directly translate
- Confidentiality definitions do not
  (due to lack of access to $V_k$)
- Most interesting of these is “DI”, being similar to ERR–CPA

How To Securely Release Unverified Plaintext in Authenticated Encryption
Andreeva, Bogdanov, Luykx, Mennink, Mouha & Yasuda; AC 2014
## Comparison with past works

<table>
<thead>
<tr>
<th>Recent Literature</th>
<th>Our Notion</th>
<th>BDPS Notion</th>
<th>RUP Notion</th>
</tr>
</thead>
<tbody>
<tr>
<td>IND–CPA</td>
<td>IND–CPA</td>
<td>IND$–CPA</td>
<td>IND–CPA</td>
</tr>
<tr>
<td></td>
<td>IND–sCCA</td>
<td>IND$–CCA</td>
<td>IND–CPA</td>
</tr>
<tr>
<td>INT–CTXT</td>
<td>CTI–CPA</td>
<td>INT–CTXT*</td>
<td>INT–CTXT</td>
</tr>
<tr>
<td>INT–CTXT*</td>
<td>INT–CTXT</td>
<td>INT–RUP</td>
<td>INT–RUP</td>
</tr>
<tr>
<td>AE</td>
<td>AE</td>
<td>≈IND$–CCA3</td>
<td>RUPAE</td>
</tr>
<tr>
<td>SAE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
RUP: A strengthened definition for AE

RUPAE := CTI–sCPA + DI + IND–CPA

⇐⇒ CTI–sCPA + ERR–CPA + IND–CPA

⇐⇒ SAE
RUP: A strengthened definition for AE

\[
\text{RUP}_{\text{AE}} := \underbrace{\text{INT–RUP}}_{\text{CTI–sCPA}} + \underbrace{\text{PA2}}_{\text{DI}} + \underbrace{\text{IND–CPA}}_{\text{SAE}}
\]

\[
\iff \underbrace{\text{CTI–sCPA}}_{\text{SAE}} + \underbrace{\text{ERR–CPA}}_{\text{IND–CPA}} + \underbrace{\text{IND–CPA}}_{\text{IND–CPA}}
\]
RUP: A strengthened definition for AE

$$RUPAE := \overset{\text{INT}\text{–RUP}}{CTI\text{–sCPA}} + \overset{\text{PA2}}{\overset{\text{DI}}{CTI\text{–sCPA}}} + \overset{\text{IND}\text{–CPA}}{\overset{\text{ERR}\text{–CPA}}{\overset{\text{SAE}}{\text{IND}\text{–CPA}}}}}$$

How To Securely Release Unverified Plaintext in Authenticated Encryption

*Andreeva, Bogdanov, Luykx, Mennink, Mouha & Yasuda; AC 2014*
### Syntactic Choices

<table>
<thead>
<tr>
<th>( C = \mathcal{E}_k(M) )</th>
<th>( c \in C \setminus \text{im}(\mathcal{E}_k) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{D}_k )</td>
<td>( \Lambda_k )</td>
</tr>
<tr>
<td>( M \in M )</td>
<td>( \bot_i \in L )</td>
</tr>
<tr>
<td>( \bot )</td>
<td>( \bot )</td>
</tr>
</tbody>
</table>

- **BDPS**: \( L, M \) disjoint
- **RUP**: \( L = M \), add \( V \)
- **RAE[\( \tau \)]**: \( L, M \) disjoint
RAE: Robust Authenticated Encryption

- Nonce-based model
- Accurately models Decrypt-then-Decode (eg Encode-then-encrypt)
- Allows leakage to be any element of the message space that is not of valid length (rather artificial limitation)
- Variable Length stretch
- Attainable rather than ideal security model

Robust Authenticated-Encryption: AEZ and the Problem that it Solves

Hoang, Krovetz & Rogaway; EC 2015
RAE: Variable Length Stretch and Attainable security

Variable Length Stretch

- Ciphertext expansion is an input parameter to $E_k$
- Gives the user control over ciphertext expansion
- Allows user to specify $\tau = 0$ without breaking security claims

Attainable Security

- Security measured against “best possible” world
- Contrasts with popular ideal (unobtainable) world
- User must be made aware of generic attacks

Robust Authenticated-Encryption: AEZ and the Problem that it Solves

Hoang, Krovetz & Rogaway; EC 2015
RAE: Robust Authenticated Encryption

- Nonce-based model
- Accurately models Decrypt-then-Decode (eg Encode-then-encipher)
- Allows leakage to be any element of the message space \( that \ is \ not \ of \ valid \ length \)
- Variable Length stretch
- Attainable rather than ideal security model

Robust Authenticated-Encryption: AEZ and the Problem that it Solves

Hoang, Krovetz & Rogaway; EC 2015
RAE: Robust Authenticated Encryption

- Nonce-based model
- Accurately models Decrypt-then-Decode (e.g., Encode-then-encipher)
- Allows leakage to be any element of the Leakage space that is not of valid length
- Variable Length stretch
- Attainable rather than ideal security model

Robust Authenticated-Encryption: AEZ and the Problem that it Solves

Hoang, Krovetz & Rogaway; EC 2015
RAE: Robust Authenticated Encryption

- Nonce-based model
- Accurately models Decrypt-then-Decode (eg Encode-then-encipher)
- Allows leakage to be any element of the Leakage space that is not of valid length
- Variable Length stretch
- Attainable rather than ideal security model

\[
\text{RAE}[\tau] := \text{Restriction of RAE to user-independent } \tau
\]

Robust Authenticated-Encryption: AEZ and the Problem that it Solves
Hoang, Krovetz & Rogaway; EC 2015
Comparison of Robust AE notions

RAE[$\tau$] \[\text{[HKR15]}\] \rightarrow SAE \rightarrow RUPAE \[\text{[ABLMMY14]}\]

RAE[$\tau$] \leftarrow SAE \leftarrow RUPAE

IND$\$-$CCA3 \[\text{[BDPS13]}\]
Conclusions

1. Security for the Real World

2. Comparison of Strengthened AE notions

3. Conclusions
   - Conclusion
To summarise

In this talk, we have

The full paper is available on the IACR eprint http://eprint.iacr.org/2015/895; or, http://ia.cr/2015/895
To summarise

In this talk, we have

- Provided an intuitive mechanism for naming AE notions
- Defined SAE: a strengthened definition of AE that is simulator free
- (briefly) Compared with some alternative frameworks
- Observed the equivalence between (common variants of) RUP and RAE

The full paper is available on the IACR eprint
To summarise

In this talk, we have

- Provided an intuitive mechanism for naming AE notions
- Defined SAE: a strengthened definition of AE that is simulator free
- (briefly) Compared with some alternative frameworks
- Observed the equivalence between (common variants of) RUP and RAE

The full paper is available on the IACR eprint
To summarise

In this talk, we have

- Provided an intuitive mechanism for naming AE notions
- Defined SAE: a strengthened definition of AE that is simulator free
- (briefly) Compared with some alternative frameworks
- Observed the equivalence between (common variants of) RUP and RAE

The full paper is available on the IACR eprint [http://eprint.iacr.org/2015/895](http://eprint.iacr.org/2015/895); or, [http://ia.cr/2015/895](http://ia.cr/2015/895)
To summarise

In this talk, we have

- Provided an intuitive mechanism for naming AE notions
- Defined SAE: a strengthened definition of AE that is simulator free
- (briefly) Compared with some alternative frameworks
- Observed the equivalence between (common variants of) RUP and RAE

The full paper is available on the IACR eprint

In the full paper we provide

- The historical context behind modern AE definitions.
- An intuitive mechanism for naming AE notions.
- SAE: A simulator free strengthening of AE.
- Comparison between SAE and BDPS, RUP & RAE (we find many similarities, and discuss their differences)
- Proof that their strongest of security notions essentially coincide.
- A reminder that subtle security depends on the implementation, giving an optimisation that renders a particular RAE scheme insecure.

The full paper is available on the IACR eprint [http://eprint.iacr.org/2015/895](http://eprint.iacr.org/2015/895); or, [http://ia.cr/2015/895](http://ia.cr/2015/895)
Thank you for your time

The full paper is available on the IACR eprint
Thank you for your time

Any Questions

The full paper is available on the IACR eprint
<table>
<thead>
<tr>
<th>Quick Shortcuts</th>
<th>Decryption Failures</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 Outline</td>
<td>25 RUP: Release of Unverified Plaintext</td>
</tr>
<tr>
<td>4 Authenticated Encryption</td>
<td>27 RUP: Release of Unverified Plaintext</td>
</tr>
<tr>
<td>7 Authenticated Encryption</td>
<td>28 Comparison with past works</td>
</tr>
<tr>
<td>8 Authenticated Encryption: Syntax</td>
<td>29 RUP: A strengthened definition for AE</td>
</tr>
<tr>
<td>10 A piecewise name scheme for AE notions</td>
<td>31 RAE: Robust Authenticated Encryption</td>
</tr>
<tr>
<td>11 Decryption Leakage</td>
<td>32 RAE: Variable Length Stretch and Attainable security</td>
</tr>
<tr>
<td>12 Modelling Decryption Leakage</td>
<td>33 RAE: Robust Authenticated Encryption</td>
</tr>
<tr>
<td>13 Oracles</td>
<td>34 Comparison of Robust AE notions</td>
</tr>
<tr>
<td>14 Disallowed Queries</td>
<td>36 To summarise</td>
</tr>
<tr>
<td>15 Effective Games</td>
<td>37 In the full paper we provide</td>
</tr>
<tr>
<td>16 SAE: Subtle Authenticated Encryption</td>
<td>38 Thank you for your time</td>
</tr>
<tr>
<td>17 Error Simulatability: A means not an end</td>
<td>40 Comparison with past works</td>
</tr>
</tbody>
</table>
## Comparison with past works

<table>
<thead>
<tr>
<th>Recent Literature</th>
<th>Our Notion</th>
<th>BDPS Notion</th>
<th>RUP Notion</th>
</tr>
</thead>
<tbody>
<tr>
<td>IND–CPA</td>
<td>IND–CPA</td>
<td>IND$–CPA</td>
<td>IND–CPA</td>
</tr>
<tr>
<td></td>
<td>IND–sCCA</td>
<td>IND$–CCA</td>
<td>IND$–CPA</td>
</tr>
<tr>
<td></td>
<td>IND–sCPA</td>
<td>IND$–CVA</td>
<td>IND–CPA</td>
</tr>
<tr>
<td>INT–CTXT</td>
<td>CTI–CPA</td>
<td>INT–CTXT*</td>
<td>INT–CTXT</td>
</tr>
<tr>
<td></td>
<td>CTI–sCPA</td>
<td>INT–CTXT</td>
<td>INT–RUP</td>
</tr>
<tr>
<td>AE</td>
<td>AE</td>
<td>AE</td>
<td>AE</td>
</tr>
<tr>
<td>SAE</td>
<td>≈IND$–CCA3</td>
<td>RUPAE</td>
<td></td>
</tr>
</tbody>
</table>

Guy Barwell

Subtle Authenticated Encryption