Subtle Authenticated Encryption

Achieving AE despite Deterministic Decryption Leakage

Guy Barwell Dan Page Martijn Stam

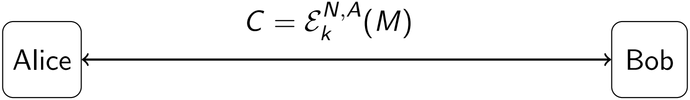
Department of Computer Science, University of Bristol

Autumn 2015

Outline

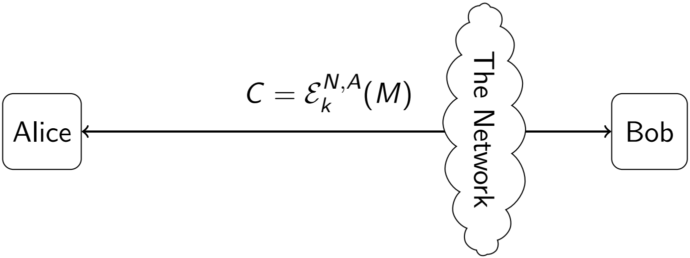
Security for the Real World

Authenticated Encryption



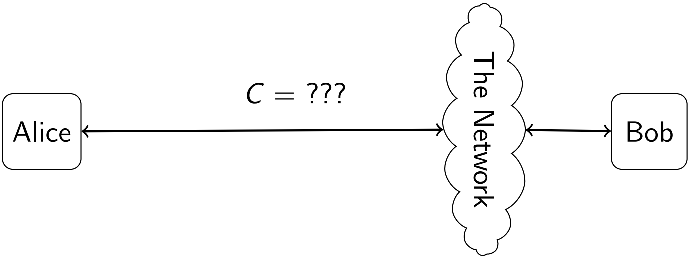
- Two parties share a key and want to communicate “securely”
- Their messages should be *private* and *authentic*
- An adversary wants to stop them doing this

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



- 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

What does the adversary want to do?

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

Powers

What can they do to help them achieve this?

- 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

$$\mathcal{E} : \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{M} \rightarrow \mathsf{C}$$
$$\mathcal{D} : \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{C} \rightarrow \mathsf{M} \cup \{\perp\}$$

Where:

K	Key space
N	Nonce space
A	Associated Data
M	Message Space
C	Ciphertext Space
$\perp$	Invalid ciphertext symbol

Authenticated Encryption: Syntax

An Authenticated Encryption scheme is a pair of algorithms

$$\begin{aligned} \mathcal{E} &: \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{M} \rightarrow \mathsf{C} \\ \mathcal{D} &: \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{C} \rightarrow \mathsf{M} \cup \{\perp\} \end{aligned}$$

Where:

K	Key space
N	Nonce space
A	Associated Data
M	Message Space
C	Ciphertext Space
$\perp$	Invalid ciphertext symbol

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

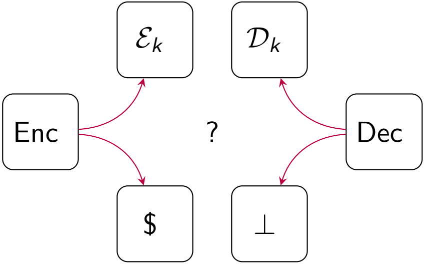
Authenticated Encryption

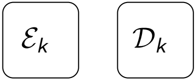
Goals

What does the adversary want to do?

Powers

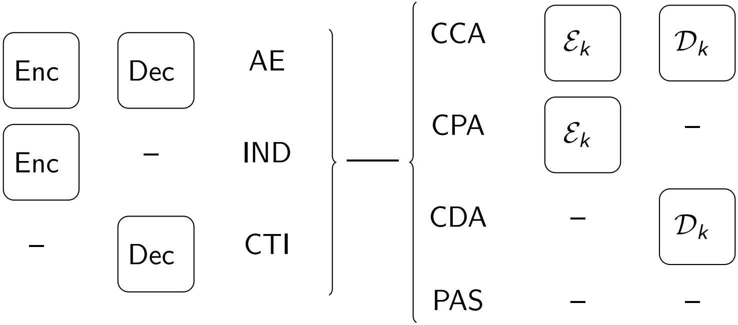
What can they do to help them achieve this?





Reference world is *ideal* rather than *attainable*.

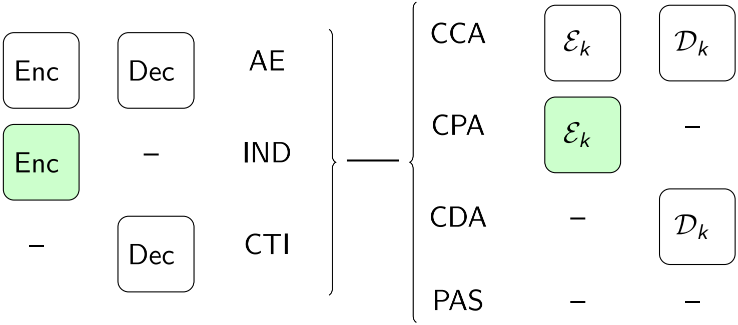
A piecewise name scheme for AE notions



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

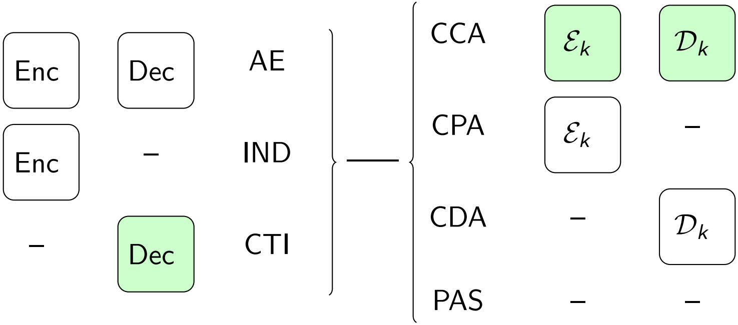
A piecewise name scheme for AE notions



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

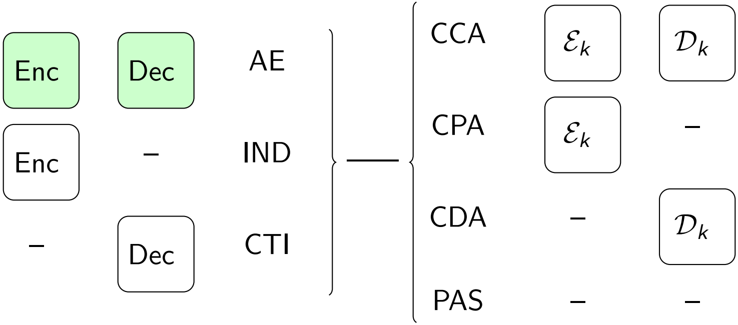
A piecewise name scheme for AE notions



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

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 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:

$$\Lambda \quad : \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{C} \quad \rightarrow \quad \{\top\} \quad \cup \quad \mathsf{L}$$

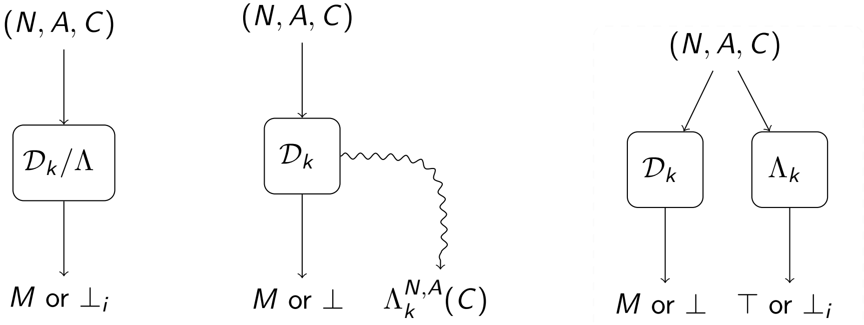
(Where an output of $\top$ corresponds to a valid message)

Modelling Decryption Leakage

So, our leakage functions looks like:

$$\Lambda \quad : \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{C} \;\rightarrow\; \{\top\} \;\cup\; \mathsf{L}$$

(Where an output of $\top$ corresponds to a valid message)

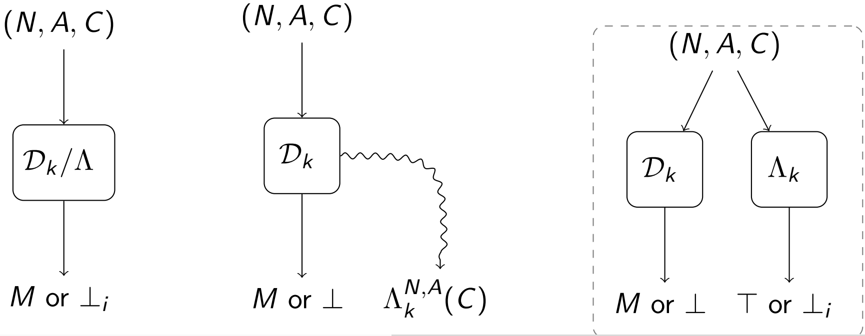


Modelling Decryption Leakage

So, our leakage functions looks like:

$$\Lambda \quad : \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{C} \rightarrow \{\top\} \cup \mathsf{L}$$

(Where an output of $\top$ corresponds to a valid message)



Oracles

Thus our oracles have the syntax:

$$
\begin{aligned}
\mathrm{Enc}, \mathcal{E} &: \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{M} \rightarrow \mathsf{C} \\
\mathrm{Dec}, \mathcal{D} &: \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{C} \rightarrow \mathsf{M} \cup \{\bot\} \\
\Lambda &: \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{C} \rightarrow \{\top\} \cup \mathsf{L}
\end{aligned}
$$

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

Enc Dec $\mathcal{E}_k$ $\mathcal{D}_k$ Λ_k

Oracles

Thus our oracles have the syntax:

$$
\begin{array}{rl}
\mathrm{Enc}, \mathcal{E} & : \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{M} \rightarrow \mathsf{C} \\
\mathrm{Dec}, \mathcal{D} & : \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{C} \rightarrow \mathsf{M} \cup \{\bot\} \\
\Lambda & : \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{C} \rightarrow \{\top\} \cup \mathsf{L}
\end{array}
$$

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

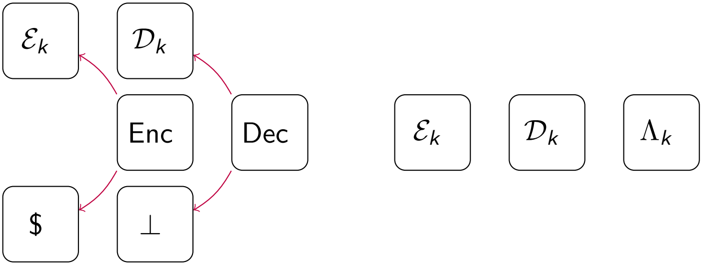
Enc Dec $\mathcal{E}_k$ $\mathcal{D}_k$ Λ_k

Oracles

Thus our oracles have the syntax:

$$\begin{array}{rlcccc} \mathrm{Enc}, \mathcal{E} & : \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{M} & \rightarrow & \mathsf{C} & & \\ \mathrm{Dec}, \mathcal{D} & : \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{C} & \rightarrow & \mathsf{M} & \cup & \{\perp\} \\ \Lambda & : \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{C} & \rightarrow & \{\top\} & \cup & \mathsf{L} \end{array}$$

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



Oracles

Thus our oracles have the syntax:

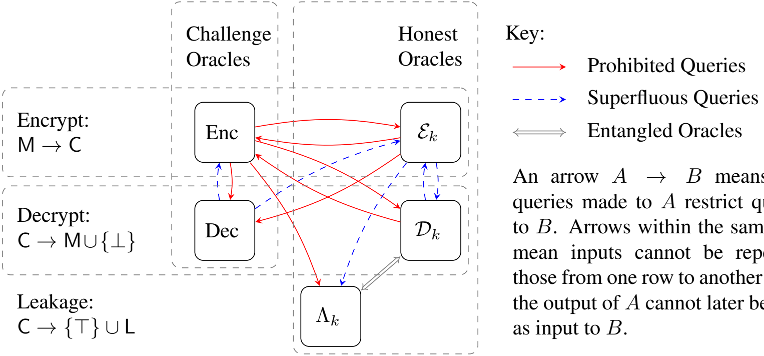
$$
\begin{aligned}
\mathrm{Enc}, \mathcal{E} &: \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{M} \rightarrow \mathsf{C} \\
\mathrm{Dec}, \mathcal{D} &: \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{C} \rightarrow \mathsf{M} \cup \{\perp\} \\
\Lambda &: \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{C} \rightarrow \{\top\} \cup \mathsf{L}
\end{aligned}
$$

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

$\boxed{\mathsf{Enc}}$ $\boxed{\mathsf{Dec}}$ $\boxed{\mathcal{E}_k}$ $\boxed{\mathcal{D}_k}$ $\boxed{\Lambda_k}$

We extend our *power* terminology with the addition of an *s* for *subtle*

Disallowed Queries



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 $24 = 3 * 2^3$ 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

The *effective games* are: AE–PAS, IND–PAS, IND–CDA, CTI–CPA, CTI–PAS and their subtle variants.

Effective Games

So, there are a total of $24 = 3 * 2^3$ 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

The *effective games* are: AE–PAS, IND–PAS, IND–CDA, CTI–CPA, CTI–PAS and their subtle variants.

Effective Games

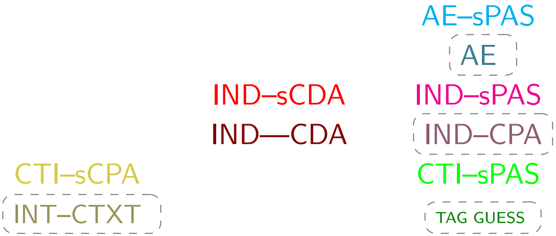
So, there are a total of $24 = 3 * 2^3$ security games, some of which are equivalent:

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

The *effective games* are: AE–PAS, IND–PAS, IND–CDA, CTI–CPA, CTI–PAS and their subtle variants.

Effective Games

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



The *effective games* are: AE–PAS, IND–PAS, IND–CDA, CTI–CPA, CTI–PAS and their subtle variants.

SAE: Subtle Authenticated Encryption

SAE := 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

Error Simulatability

"Leakage should not give out useful information"

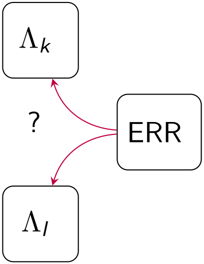
A new goal: Error Simulatability

Error Simulatability: A means not an end

Error Simulatability

"Leakage should not give out useful information"

A new goal: Error Simulatability

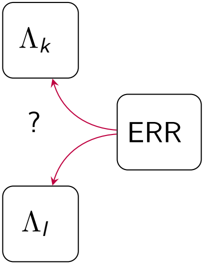


Error Simulatability: A means not an end

Error Simulatability

"Leakage should not give out useful information"

For example: ERR–PAS

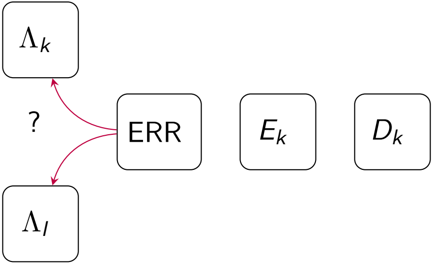


Error Simulatability: A means not an end

Error Simulatability

"Leakage should not give out useful information"

For example: ERR–CCA



Decomposing SAE

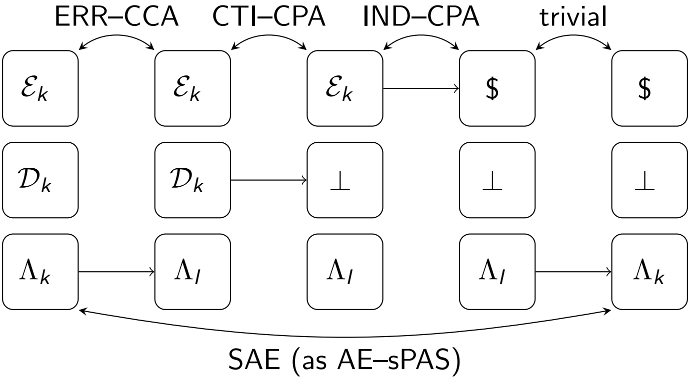
SAE decomposes in an intuitive manner

SAE $\iff$ ERR–CCA + CTI–CPA + IND–CPA

Decomposing SAE

SAE decomposes in an intuitive manner

SAE $\iff$ ERR–CCA + CTI–CPA + IND–CPA



Comparison of Strengthened AE notions

Syntactic Choices

	$\mathcal{D}_k$	Λ_k
$C = \mathcal{E}_k(M)$	$M \in \mathsf{M}$	$\top$
$c \in \mathsf{C} \setminus \mathrm{im}(\mathcal{E}_k)$	$\bot$	$\bot_i \in \mathsf{L}$

- BDPS: L, M disjoint
- RUP: $\mathsf{L} = \mathsf{M}$, add V
- RAE[τ]: L, M disjoint

Syntactic Choices

	$\mathcal{D}_k$	Λ_k
$C = \mathcal{E}_k(M)$	$M \in \mathsf{M}$	$\top$
$c \in \mathsf{C} \setminus \mathrm{im}(\mathcal{E}_k)$	$\perp$	$\perp_i \in \mathsf{L}$

D_k

- BDPS: L, M disjoint
- RUP: $\mathsf{L} = \mathsf{M}$, add V
- RAE[τ]: 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

Our Notion	BDPS Notion
IND–CPA	IND\$–CPA
IND–sCCA	IND\$–CCA
IND–sCPA	IND\$–CVA
CTI–CPA	INT–CTXT*
CTI–sCPA	INT–CTXT
AE	
SAE	≈IND\$–CCA3

RUP: Release of Unverified Plaintext

- Nonce-based definitions, relations and separations.
- Provisioned for the leakage of a candidate plaintext.
- Models Decrypt-then-authenticate (eg MtE,M&E).
- Observes that if Λ_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

	$\mathcal{D}_k$	Λ_k
$C = \mathcal{E}_k(M)$	$M \in \mathsf{M}$	$\top$
$c \in \mathsf{C} \setminus \mathrm{im}(\mathcal{E}_k)$	$\perp$	$\perp_i \in \mathsf{L}$

D_k

- BDPS: L, M disjoint
- RUP: $\mathsf{L} = \mathsf{M}$, add V
- RAE[τ]: L, M disjoint

Syntactic Choices

	$\mathcal{D}_k$	Λ_k
$C = \mathcal{E}_k(M)$	$M \in \mathsf{M}$	$\top$
$c \in \mathsf{C} \setminus \mathrm{im}(\mathcal{E}_k)$	$\perp$	$\perp_i \in \mathsf{L}$

V_k

D_k

- BDPS: L, M disjoint
- RUP: $\mathsf{L} = \mathsf{M}$, add V
- RAE[τ]: 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

Recent Literature	Our Notion	BDPS Notion	RUP Notion
IND–CPA	IND–CPA	IND\$–CPA	IND–CPA
	IND–sCCA	IND\$–CCA	
	IND–sCPA	IND\$–CVA	
INT–CTXT	CTI–CPA	INT–CTXT*	INT–CTXT
	CTI–sCPA	INT–CTXT	INT–RUP
AE	AE		AE
	SAE	≈IND\$–CCA3	RUPAE

RUP: A strengthened definition for AE

RUPAE	:=	CTI–sCPA	+	DI	+	IND–CPA
	$\iff$	CTI–sCPA	+	ERR–CPA	+	IND–CPA
	$\iff$	SAE				

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

RUP: A strengthened definition for AE

$$\text{RUPAE} := \overbrace{\text{CTI–sCPA}}^{\text{INT–RUP}} + \overbrace{\text{DI}}^{\text{PA2}} + \text{IND–CPA}$$

$$\iff \text{CTI–sCPA} + \text{ERR–CPA} + \text{IND–CPA}$$

$$\iff \text{SAE}$$

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

RUP: A strengthened definition for AE

$$
\begin{aligned}
\text{RUPAE} \quad &:= \quad \overbrace{\text{CTI–sCPA}}^{\text{INT–RUP}} + \overbrace{\text{DI}}^{\text{PA2}} + \text{IND–CPA} \\
&\iff \quad \text{CTI–sCPA} + \text{ERR–CPA} + \text{IND–CPA} \\
&\iff \quad \text{SAE}
\end{aligned}
$$

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

Syntactic Choices

	$\mathcal{D}_k$	Λ_k
$C = \mathcal{E}_k(M)$	$M \in \mathrm{M}$	$\top$
$c \in \mathrm{C} \setminus \mathrm{im}(\mathcal{E}_k)$	$\bot$	$\bot_i \in \mathrm{L}$

D_k

- BDPS: L, M disjoint
- RUP: $\mathrm{L} = \mathrm{M}$, add V
- RAE[τ]: L, M disjoint

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 (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 $\mathcal{E}_k$

- 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 (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

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

RAE: Robust Authenticated Encryption

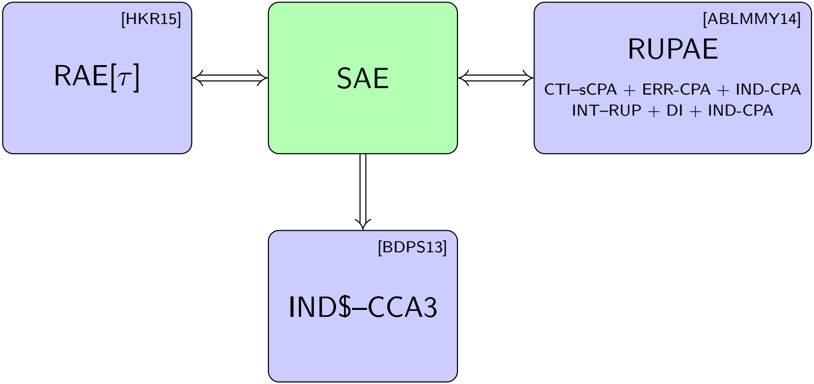
- Nonce-based model
- Accurately models Decrypt-then-Decode (eg Encode-then-encipher)
- Allows leakage to be any element of the Leakage space

  that is not of valid length
- *Variable Length stretch*
- Attainable rather than ideal security model

RAE[τ] := Restriction of RAE to user-independent τ

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

Comparison of Robust AE notions



Conclusions

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
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
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
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
http://eprint.iacr.org/2015/895; or, *http://ia.cr/2015/895*

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; or, *http://ia.cr/2015/895*

Thank you for your time

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

Thank you for your time

Any Questions

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

Quick Shortcuts

Comparison with past works

Recent Literature	Our Notion	BDPS Notion	RUP Notion
IND–CPA	IND–CPA	IND$–CPA	IND–CPA
	IND–sCCA	IND$–CCA	
	IND–sCPA	IND$–CVA	
INT–CTXT	CTI–CPA	INT–CTXT*	INT–CTXT
	CTI–sCPA	INT–CTXT	INT–RUP
AE	AE		AE
	SAE	≈IND$–CCA3	RUPAE