On Some Constructions for Authenticated Encryption with Associated Data

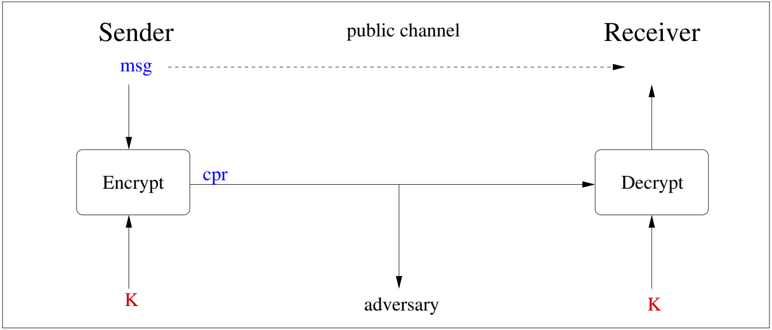
Palash Sarkar

Applied Statistics Unit
Indian Statistical Institute, Kolkata
India
palash@isical.ac.in

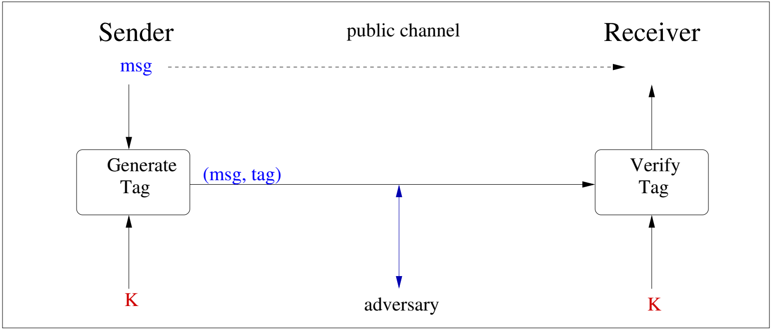
(Partially based on joint work with Debrup Chakraborty)

Directions in Authenticated Ciphers – DIAC 2012,
6th July 2012

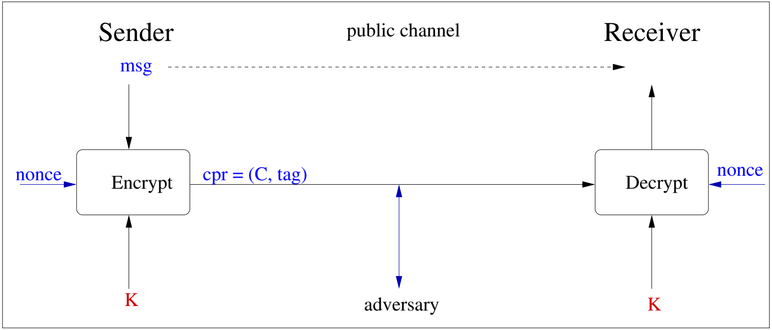
Encryption


Sender
public channel
Receiver
msg
Encrypt
cpr
Decrypt
K
adversary
K

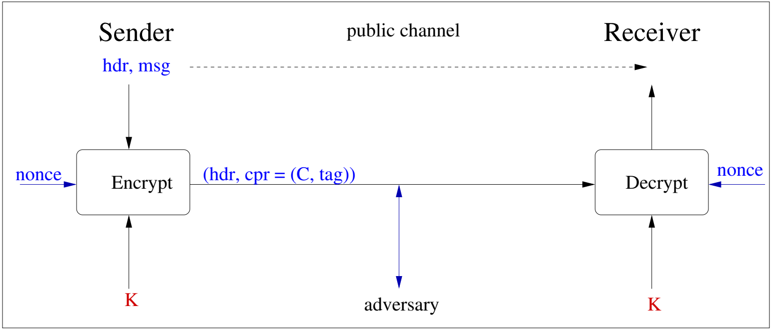
Authentication



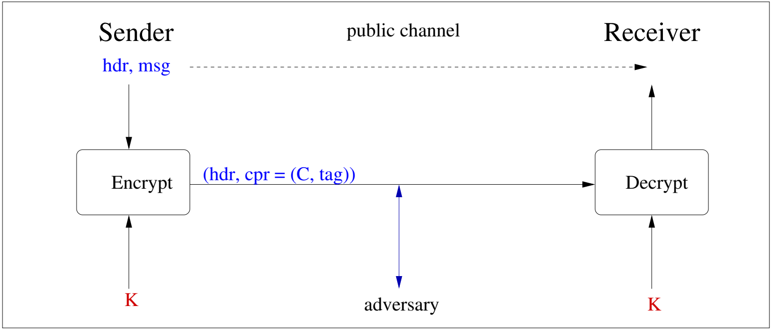
Authenticated Encryption (AE)



AE with Associated Data (AEAD)



Deterministic AEAD (DAEAD)



Construction Approaches

We will consider:

- Single-pass block cipher modes of operations.
  - From tweakable block ciphers.
  - From (plain) block ciphers.

Construction Approaches

We will consider:

- Single-pass block cipher modes of operations.
    - From tweakable block ciphers.
    - From (plain) block ciphers.
- Stream cipher with IV and a hash function (with provably low collision and differential probabilities).

Construction Approaches

We will consider:

- Single-pass block cipher modes of operations.
  - From tweakable block ciphers.
  - From (plain) block ciphers.
- Stream cipher with IV and a hash function (with provably low collision and differential probabilities).

Other approaches:

- Direct construction of an integrated primitive: PHELIX, SOBER, AEGIS, ...
- From permutations (Bertoni at al 2011).
- Generic conversion from AE to AEAD: AE+MAC (Rogaway 2002); AE+CRHF (Sarkar 2010).

Some AE(AD) Schemes from Block Ciphers

Two-pass: Cost per block (approx): **2[BC]** or **1[BC]+1[M]**.

- CCM: Counter + CBC-MAC; standardised by NIST (USA).
- GCM: Counter + (universal) hash; standardised by NIST (USA).
- CWC: Carter-Wegman + Counter Mode; EAX; CHM: CENC + hash; CCFB: between one and two-pass.

Some AE(AD) Schemes from Block Ciphers

Two-pass: Cost per block (approx): **2[BC]** or **1[BC]+1[M]**.

- CCM: Counter + CBC-MAC; standardised by NIST (USA).
- GCM: Counter + (universal) hash; standardised by NIST (USA).
- CWC: Carter-Wegman + Counter Mode; EAX; CHM: CENC + hash; CCFB: between one and two-pass.

Single-pass: Cost per block (approx): **1[BC]+**SOMETHING.

- Constructions having associated (US) patents:
  - IACBC, IAPM: (Jutla, 2001);
  - XCBC, XECB: (Gligor-Donescu, 2001);
  - OCB: (Rogaway et al, 2001; Rogaway 2004; Krovetz-Rogaway, 2011).

Some AE(AD) Schemes from Block Ciphers

Two-pass: Cost per block (approx): **2[BC]** or **1[BC]+1[M]**.

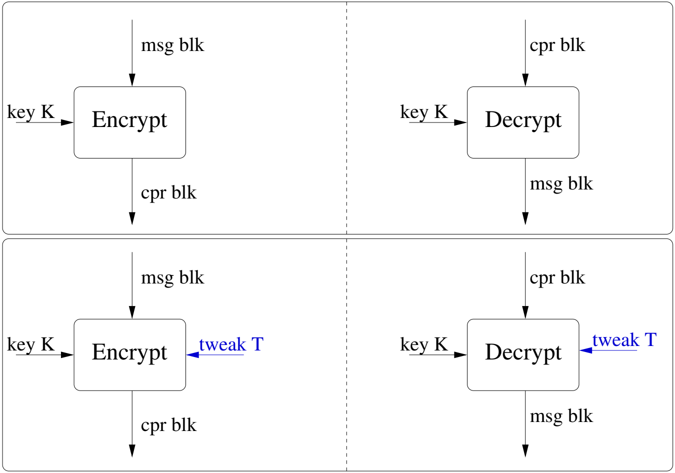
- CCM: Counter + CBC-MAC; standardised by NIST (USA).
- GCM: Counter + (universal) hash; standardised by NIST (USA).
- CWC: Carter-Wegman + Counter Mode; EAX; CHM: CENC + hash; CCFB: between one and two-pass.

Single-pass: Cost per block (approx): **1[BC]+**SOMETHING.

- Constructions having associated (US) patents:
  - IACBC, IAPM: (Jutla, 2001);
  - XCBC, XECB: (Gligor-Donescu, 2001);
  - OCB: (Rogaway et al, 2001; Rogaway 2004; Krovetz-Rogaway, 2011).
- Constructions without assoicated patents:
  - Chakraborty-Sarkar (2006, 2008); Sarkar (2010).

AE(AD) from Tweakable Block Ciphers

(Tweakable) Block Ciphers



- Non-secret tweak allows flexibility in designing applications.
- Formalised by Liskov-Rivest-Wagner (2002).

TBC and Modes of Operations

Rogaway (2004).

- Provides efficient construction of a TBC family.
- Introduces a technique for using a TBC family to construct different modes of operations.

TBC and Modes of Operations

Rogaway (2004).

- Provides efficient construction of a TBC family.
- Introduces a technique for using a TBC family to construct different modes of operations.

Chakraborty-Sarkar (2006, 2008).

- A new TBC family obtained by generalising Rogaway's construction.
  - Can be instantiated over $GF(2^n)$ or $\mathbb{Z}_{2^n}$.
- Provides two techniques for constructing modes of operations.
  - The first technique generalises Rogaway's work.
  - A second new technique.
- Provides a family of modes of operations for MAC, AE and AEAD.
  - Only one of each kind was known earlier.

From BC to TBC (Generalising Rogaway 2004)

XE Construction (tweakable PRP): $\widetilde{E}_K^{N,l}(M) = E_K(M + \Delta)$.

XEX Construction (tweakable SPRP): $\widetilde{E}_K^{N,l}(M) = E_K(M + \Delta) - \Delta$.

where $\Delta = f_l(\mathcal{N})$ and $\mathcal{N} = E_K(N)$.

- $\mathbf{f_1}, \mathbf{f_2}, \ldots$ is a masking sequence.
- (N, l) is the tweak; tweak space is $\{0, 1\}^n \times \{1, 2, \ldots, 2^n - 2\}$.
- Addition (and subtraction) is over a ring **R.**

From BC to TBC (Generalising Rogaway 2004)

XE Construction (tweakable PRP): $\widetilde{E_K}^{N,l}(M) = E_K(M + \Delta)$.

XEX Construction (tweakable SPRP): $\widetilde{E_K}^{N,l}(M) = E_K(M + \Delta) - \Delta$.

where $\Delta = f_l(\mathcal{N})$ and $\mathcal{N} = E_K(N)$.

- $\mathbf{f_1}, \mathbf{f_2}, \ldots$ is a masking sequence.
- (N, l) is the tweak; tweak space is $\{0,1\}^n \times \{1, 2, \ldots, 2^n - 2\}$.
- Addition (and subtraction) is over a ring **R**.

The generalisation arises from the notion of masking sequence and working over **R**.

Masking Sequence: Definition

$f_1, f_2, \ldots, f_m$ is an (n, m, μ) masking sequence if: $(f_s : \{0,1\}^n \rightarrow \{0,1\}^n)$

$$
\begin{array}{lcl}
\mathrm{Prob}[f_s(\mathcal{N}) = \alpha] & \leq & \frac{1}{\mu} \\
\mathrm{Prob}[f_s(\mathcal{N}) = \mathcal{N} + \alpha] & \leq & \frac{1}{\mu} \\
\mathrm{Prob}[f_s(\mathcal{N}) = f_t(\mathcal{N}) + \alpha] & \leq & \frac{1}{\mu} \\
\mathrm{Prob}[f_s(\mathcal{N}) = f_t(\mathcal{N}') + \alpha] & \leq & \frac{1}{\mu}
\end{array}
$$

where

- $\mathcal{N}$ and $\mathcal{N}'$ are randomly and independently chosen from $\{0,1\}^n$.
- α is any fixed n-bit string.

Instantiations of **R**

R as $GF(2^n)$:

- Define $f_i(\mathcal{N}) = \mathcal{N}G^i$ where G is an $n \times n$ binary matrix whose characteristic polynomial is primitive over $GF(2)$.
- $f_1, f_2, \ldots, f_{2^n-2}$ is an $(n, 2^n - 2, 2^n)$ masking sequence.
- Efficient instantiations of G: powering method, (word oriented) LFSR, CA.

Instantiations of **R**

R as $GF(2^n)$:

- Define $f_i(\mathcal{N}) = \mathcal{N}G^i$ where G is an $n \times n$ binary matrix whose characteristic polynomial is primitive over $GF(2)$.
- $f_1, f_2, \ldots, f_{2^n-2}$ is an $(n, 2^n - 2, 2^n)$ masking sequence.
- Efficient instantiations of G: powering method, (word oriented) LFSR, CA.

R as Z_{2^n}:

- Let $p = 2^n + \delta$ be a prime, with δ as small as possible, eg: $p = 2^{128} + 51$.
- Define $f_i(\mathcal{N}) = ((i + 1)\mathcal{N} \bmod p) \bmod 2^n$.
- $f_1, f_2, \ldots, f_{2^n-2}$ is an $(n, 2^n - 2, 2^{n-1}/(\delta + 1))$ masking sequence.

Instantiations of **R**

R as $GF(2^n)$:

- Define $f_i(\mathcal{N}) = \mathcal{N}G^i$ where G is an $n \times n$ binary matrix whose characteristic polynomial is primitive over $GF(2)$.
- $f_1, f_2, \ldots, f_{2^n-2}$ is an $(n, 2^n - 2, 2^n)$ masking sequence.
- Efficient instantiations of G: powering method, (word oriented) LFSR, CA.

R as Z_{2^n}:

- Let $p = 2^n + \delta$ be a prime, with δ as small as possible, eg: $p = 2^{128} + 51$.
- Define $f_i(\mathcal{N}) = ((i + 1)\mathcal{N} \bmod p) \bmod 2^n$.
- $f_1, f_2, \ldots, f_{2^n-2}$ is an $(n, 2^n - 2, 2^{n-1}/(\delta + 1))$ masking sequence.

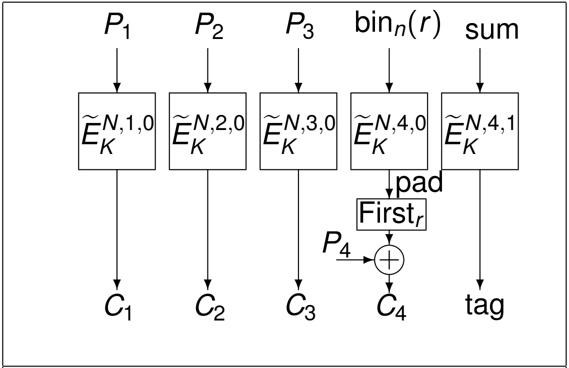
Rogaway (2004): **R** as $GF(2^n)$ with the powering construction.

From TBC to AE

XEX-TBC $\widetilde{E}$ with tweak space $\{0,1\}^n \times \{1,2,\ldots,2^{n/2}\} \times \{0,1\}$.

From TBC to AE

XEX-TBC $\widetilde{E}$ with tweak space $\{0,1\}^n \times \{1,2,\ldots,2^{n/2}\} \times \{0,1\}$.



N is used as a nonce; $r = \text{len}(P_4)$;

$\text{sum} = P_1 + P_2 + P_3 + (C_4 || 0^*) + \text{pad}$

Figure: Rogaway's 2004 TBC-to-AE construction lifted to **R**.

From TBC to AE

Required tweak space: $\{0,1\}^n \times \{1,2,\ldots,2^{n/2}\} \times \{0,1\}$.

Tweak space of XEX-TBC: $\{0,1\}^n \times \{1,2,\ldots,2^n-2\}$.

From TBC to AE

Required tweak space: $\{0,1\}^n \times \{1,2,\ldots,2^{n/2}\} \times \{0,1\}$.

Tweak space of XEX-TBC: $\{0,1\}^n \times \{1,2,\ldots,2^n-2\}$.

Injective Map $\phi : \{1,2,\ldots,2^{n/2}\} \times \{0,1\} \to \{1,2,\ldots,2^n-2\}$.

From TBC to AE

Required tweak space: $\{0,1\}^n \times \{1,2,\ldots,2^{n/2}\} \times \{0,1\}$.

Tweak space of XEX-TBC: $\{0,1\}^n \times \{1,2,\ldots,2^n-2\}$.

Injective Map $\phi : \{1,2,\ldots,2^{n/2}\} \times \{0,1\} \to \{1,2,\ldots,2^n-2\}$.

- Linear Separation: $\phi(i,b) = i + Lb$ where L is an appropriately chosen "large" integer.
  - **R** as $GF(2^n)$: L is the discrete log of $(x+1)$ (Rogaway 2004).
  - **R** as Z_{2^n}: $L = 2^{n/2}$.
- Interleaved Separation: $\phi(i,b) = 2i + b$.
  - Avoids the (design time) discrete log computation.

From TBC to AE

Required tweak space: $\{0,1\}^n \times \{1,2,\ldots,2^{n/2}\} \times \{0,1\}$.

Tweak space of XEX-TBC: $\{0,1\}^n \times \{1,2,\ldots,2^n - 2\}$.

Injective Map $\phi : \{1,2,\ldots,2^{n/2}\} \times \{0,1\} \to \{1,2,\ldots,2^n - 2\}$.

- Linear Separation: $\phi(i,b) = i + Lb$ where L is an appropriately chosen “large” integer.
  - **R** as $GF(2^n)$: L is the discrete log of $(x+1)$ (Rogaway 2004).
  - **R** as Z_{2^n}: $L = 2^{n/2}$.
- Interleaved Separation: $\phi(i,b) = 2i + b$.
  - Avoids the (design time) discrete log computation.

Variations of the above technique provides constructions for MAC and AEAD.

AE(AD) from (Plain) Block Ciphers

AE Functions

Random function $f : \mathcal{N} \times \mathcal{X} \to \mathcal{X} \times \{0,1\}^t$; $f(N, X) = (Y, \mathrm{tag})$.
(Randomness arising from uniform random K.)

AE Functions

Random function $f : \mathcal{N} \times \mathcal{X} \to \mathcal{X} \times \{0,1\}^t$; $f(N, X) = (Y, \text{tag})$.
(Randomness arising from uniform random K.)

- $f_N^{\text{main}}(X) \triangleq Y$, a length preserving permutation.
- $\widetilde{f}$: authentication function associated with f.
  $\widetilde{f}(N, Y) \triangleq \text{tag}$ if $f(N, X) = (Y, \text{tag})$ for some $X \in \mathcal{X}$;

AE Functions

Random function $f : \mathcal{N} \times \mathcal{X} \to \mathcal{X} \times \{0,1\}^t$; $f(N, X) = (Y, \text{tag})$.
(Randomness arising from uniform random K.)

- $f_N^{\text{main}}(X) \triangleq Y$, a length preserving permutation.
- $\widetilde{f}$: authentication function associated with f.
  $\widetilde{f}(N, Y) \triangleq \text{tag}$ if $f(N, X) = (Y, \text{tag})$ for some $X \in \mathcal{X}$;
- AE-privacy of f: follows from
  - PRF-property of f^{main} against nonce-respecting adversaries.
- AE-auth of f: follows from
  - AE-privacy of f;
  - PRF-property of $\widetilde{f}$.

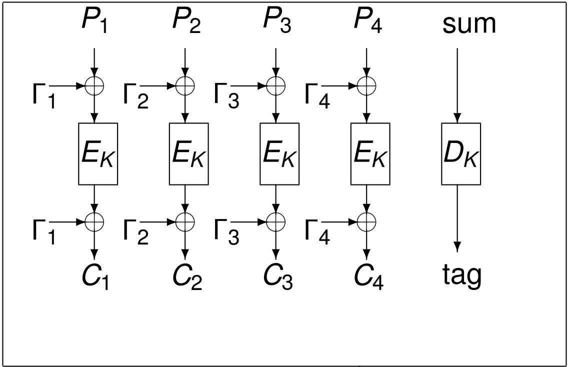
AE Functions

Random function $f : \mathcal{N} \times \mathcal{X} \rightarrow \mathcal{X} \times \{0,1\}^t$; $f(N, X) = (Y, \text{tag})$.
(Randomness arising from uniform random K.)

- $f_N^{\text{main}}(X) \triangleq Y$, a length preserving permutation.
- $\widetilde{f}$: authentication function associated with f.
  $\widetilde{f}(N, Y) \triangleq \text{tag}$ if $f(N, X) = (Y, \text{tag})$ for some $X \in \mathcal{X}$;
- AE-privacy of f: follows from
  - PRF-property of f^{main} against nonce-respecting adversaries.
- AE-auth of f: follows from
  - AE-privacy of f;
  - PRF-property of $\widetilde{f}$.

$\widetilde{f}$ itself can serve as a standalone MAC function.

PAE (Sarkar 2010): Only Full Blocks

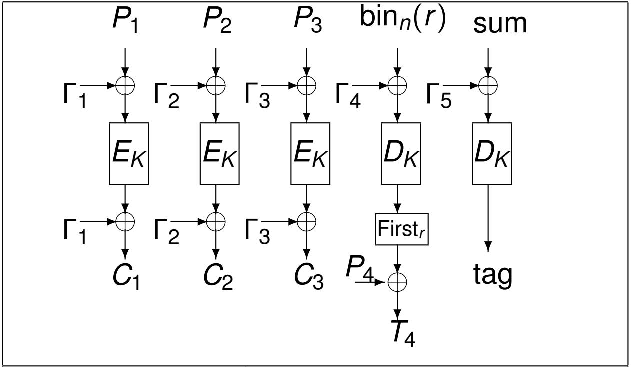


N is a nonce; $\Gamma_i = \psi^i(D_K(N))$;

sum $= P_1 \oplus P_2 \oplus P_3 \oplus C_4$.

$\psi : \mathbb{F}_{2^n} \rightarrow \mathbb{F}_{2^n}$ is a linear map whose minimal polynomial over $\mathbb{F}_2$ is primitive.

PAE (Sarkar 2010): Last Block is Partial



$C_4 = T_4 || (10^{n-r-1})$, sum $= P_1 \oplus P_2 \oplus P_3 \oplus C_4$

PAEAD (Sarkar 2010)

$\text{PAEAD.Encrypt}_{E_K,\text{fStr}}(N, H, P)$
1. if H is null, return $\text{PAE.Encrypt}_{E_K}(N, P)$;
2. $(C, \text{tag}_1) = \text{PAE.Encrypt}_{E_K}(N, P)$;
3. $\upsilon = D_K(\text{fStr})$;
4. $\text{tag}_2 = \text{iPMAC}_{D_K}(\upsilon \| H)$;
5. return $(C, \text{tag}_1 \oplus \text{tag}_2)$.

- fStr is a fixed string without any secrecy requirement.
- iPMAC is a MAC algorithm which is also given in Sarkar (2010).

Advantages

- Single-pass with efficient masking:
  - Word-oriented LFSR based masking should be faster than competitors on 32-bit machines.
  - Support for AES-NI and 128-bit instructions has changed the game for Intel processors.

Advantages

- Single-pass with efficient masking:
  - Word-oriented LFSR based masking should be faster than competitors on 32-bit machines.
  - Support for AES-NI and 128-bit instructions has changed the game for Intel processors.
  - But, 98% of the CPU market consists of embedded CPUs (Christof Paar, Indocrypt 2011).

Advantages

- Single-pass with efficient masking:
  - Word-oriented LFSR based masking should be faster than competitors on 32-bit machines.
  - Support for AES-NI and 128-bit instructions has changed the game for Intel processors.
  - But, 98% of the CPU market consists of embedded CPUs (Christof Paar, Indocrypt 2011).
- Reconfigurable: easy to change the masking functions.
  - Simply choose another suitable ψ.
  - Yields a large family of schemes enjoying the same security promise.
  - Provides an opportunity to combine "provable security" with "security-by-obscurity".

Advantages (contd.)

- Versatility: A single module (hardware/software) can be used for different tasks.
  - Authentication.
  - Authenticated encryption.
  - Authenticated encryption with associated data.

Advantages (contd.)

- Versatility: A single module (hardware/software) can be used for different tasks.
  - Authentication.
  - Authenticated encryption.
  - Authenticated encryption with associated data.
- Simplified Decryption: The decryption algorithms of PAE and PAEAD do not require $E_K()$.
  - Leads to smaller hardware for lower-level operatives who only need to decrypt the encrypted instructions that are received.

Advantages (contd.)

- Versatility: A single module (hardware/software) can be used for different tasks.
  - Authentication.
  - Authenticated encryption.
  - Authenticated encryption with associated data.
- Simplified Decryption: The decryption algorithms of PAE and PAEAD do not require $E_K()$.
  - Leads to smaller hardware for lower-level operatives who only need to decrypt the encrypted instructions that are received.
- Simplified Encryption: Obtained from a variant PAE-1 (and also PAEAD-1).
  - The encryption algorithm requires only $E_K()$; leads to smaller hardware for devices which only need to transmit encrypted information.

AE(AD) from Stream Ciphers With IV

Components

Stream cipher with IV: $\text{SC}_K : \{0,1\}^n \rightarrow \{0,1\}^L$.

- L long enough to encrypt practical-sized messages.
- Modelled as a PRF.

Components

Stream cipher with IV: $\mathrm{SC}_K : \{0,1\}^n \rightarrow \{0,1\}^L$.

- L long enough to encrypt practical-sized messages.
- Modelled as a PRF.

Hash function: a keyed family $\{\mathrm{Hash}_\tau\}$; τ is the hash key.

- Low collision probability. For all distinct x and x'
  $\Pr_\tau[\mathrm{Hash}_\tau(x) = \mathrm{Hash}_\tau(x')]$ is low.
- Low differential probability. For all distinct x and x' and any y,
  $\Pr_\tau[\mathrm{Hash}_\tau(x) \oplus \mathrm{Hash}_\tau(x') = y]$ is low.

Components

Stream cipher with IV: $\mathrm{SC}_K : \{0,1\}^n \to \{0,1\}^L$.

- L long enough to encrypt practical-sized messages.
- Modelled as a PRF.

Hash function: a keyed family $\{\mathrm{Hash}_\tau\}$; τ is the hash key.

- Low collision probability. For all distinct x and x' $\Pr_\tau[\mathrm{Hash}_\tau(x) = \mathrm{Hash}_\tau(x')]$ is low.
- Low differential probability. For all distinct x and x' and any y, $\Pr_\tau[\mathrm{Hash}_\tau(x) \oplus \mathrm{Hash}_\tau(x') = y]$ is low.

Type-I hash function: key is a short fixed length string.

- Example: polynomial hashing.

Type-II hash function: key is as long as the message (or longer).

- Example: multilinear hash; UMAC.

$$
\begin{aligned}
&\text{AE-1.Encrypt}_{K,\tau}(N, M)\\
&\quad (R, Z) = \text{SC}_K(N);\\
&\quad C = M \oplus Z;\\
&\quad \text{tag} = \text{Hash}_\tau(C) \oplus R.
\end{aligned}
$$

AE

AE-1.Encrypt$_{K,\tau}(N, M)$	AE-2.Encrypt$_{K,K'}(N, M)$
$(R, Z) = \mathrm{SC}_K(N)$; $C = M \oplus Z$; $\mathrm{tag} = \mathrm{Hash}_\tau(C) \oplus R$.	$\tau = \mathrm{SC}_K(K')$; $(R, Z) = \mathrm{SC}_K(N)$; $C = M \oplus Z$; $\mathrm{tag} = \mathrm{Hash}_\tau(C) \oplus R$.

AE

AE-1.Encrypt$_{K,\tau}(N, M)$ $(R, Z) = \mathrm{SC}_K(N)$; $C = M \oplus Z$; tag $= \mathrm{Hash}_\tau(C) \oplus R$.	AE-2.Encrypt$_{K,K'}(N, M)$ $\tau = \mathrm{SC}_K(K')$; $(R, Z) = \mathrm{SC}_K(N)$; $C = M \oplus Z$; tag $= \mathrm{Hash}_\tau(C) \oplus R$.
AE-3.Encrypt$_{K,\tau}(N, M)$ $(R, Z) = \mathrm{SC}_K(N)$; $C = M \oplus Z$; tag $= \mathrm{Hash}_\tau(M) \oplus R$.	AE-4.Encrypt$_{K,K'}(N, M)$ $\tau = \mathrm{SC}_K(K')$; $(R, Z) = \mathrm{SC}_K(N)$; $C = M \oplus Z$; tag $= \mathrm{Hash}_\tau(M) \oplus R$.

AE

AE-1.Encrypt$_{K,\tau}(N, M)$ $(R, Z) = \mathrm{SC}_K(N)$; $C = M \oplus Z$; $\mathrm{tag} = \mathrm{Hash}_\tau(C) \oplus R$.	AE-2.Encrypt$_{K,K'}(N, M)$ $\tau = \mathrm{SC}_K(K')$; $(R, Z) = \mathrm{SC}_K(N)$; $C = M \oplus Z$; $\mathrm{tag} = \mathrm{Hash}_\tau(C) \oplus R$.
AE-3.Encrypt$_{K,\tau}(N, M)$ $(R, Z) = \mathrm{SC}_K(N)$; $C = M \oplus Z$; $\mathrm{tag} = \mathrm{Hash}_\tau(M) \oplus R$.	AE-4.Encrypt$_{K,K'}(N, M)$ $\tau = \mathrm{SC}_K(K')$; $(R, Z) = \mathrm{SC}_K(N)$; $C = M \oplus Z$; $\mathrm{tag} = \mathrm{Hash}_\tau(M) \oplus R$.

- AE-1: used by Bernstein during eSTREAM as a standard way of achieving AE; others from Sarkar (2011).
- AE-1, AE-3: suitable for Type-I hash functions; AE-2, AE-4: suitable for Type-II hash functions.
- AE-1, AE-2: hash the ciphertext; AE-3, AE-4: hash the message.

AEAD (Sarkar 2011)

AEAD-1.Encrypt$_{K,\tau}(H, N, M)$

$(R, Z) = \mathrm{SC}_K(N)$;

$C = M \oplus Z$;

tag $= \mathrm{Hash}_\tau(H, C) \oplus R$.

AEAD (Sarkar 2011)

AEAD-1	AEAD-2
$\text{AEAD-1.Encrypt}_{K,\tau}(H, N, M)$ $(R, Z) = \text{SC}_K(N);$ $C = M \oplus Z;$ $\text{tag} = \text{Hash}_\tau(H, C) \oplus R.$	$\text{AEAD-2.Encrypt}_{K,K'}(H, N, M)$ $\tau = \text{SC}_K(K');$ $(R, Z) = \text{SC}_K(N);$ $C = M \oplus Z;$ $\text{tag} = \text{Hash}_\tau(H, C) \oplus R.$

AEAD (Sarkar 2011)

$\text{AEAD-1.Encrypt}_{K,\tau}(H,N,M)$ $(R,Z) = \text{SC}_K(N)$; $C = M \oplus Z$; $\text{tag} = \text{Hash}_\tau(H,C) \oplus R$.	$\text{AEAD-2.Encrypt}_{K,K'}(H,N,M)$ $\tau = \text{SC}_K(K')$; $(R,Z) = \text{SC}_K(N)$; $C = M \oplus Z$; $\text{tag} = \text{Hash}_\tau(H,C) \oplus R$.
$\text{AEAD-3.Encrypt}_{K,\tau}(H,N,M)$ $(R,Z) = \text{SC}_K(N)$; $C = M \oplus Z$; $\text{tag} = \text{Hash}_\tau(H,M) \oplus R$.	$\text{AEAD-4.Encrypt}_{K,K'}(H,N,M)$ $\tau = \text{SC}_K(K')$; $(R,Z) = \text{SC}_K(N)$; $C = M \oplus Z$; $\text{tag} = \text{Hash}_\tau(H,M) \oplus R$.

AEAD (Sarkar 2011)

$\text{AEAD-5.Encrypt}_{K,\tau}(H, N, M)$

$V = \text{Hash}_{\tau}(H, N);$

$(R, Z) = \text{SC}_K(V);$

$C = M \oplus Z;$

$\text{tag} = \text{Hash}_{\tau}(C) \oplus R.$

AEAD (Sarkar 2011)

AEAD-5	AEAD-6
$\text{AEAD-5.Encrypt}_{K,\tau}(H, N, M)$ $V = \text{Hash}_\tau(H, N);$ $(R, Z) = \text{SC}_K(V);$ $C = M \oplus Z;$ $\text{tag} = \text{Hash}_\tau(C) \oplus R.$	$\text{AEAD-6.Encrypt}_{K,K'}(H, N, M)$ $(\tau_1, \tau_2) = \text{SC}_K(K');$ $V = \text{Hash}_{\tau_1}(H, N);$ $(R, Z) = \text{SC}_K(V);$ $C = M \oplus Z;$ $\text{tag} = \text{Hash}_{\tau_2}(C) \oplus R.$

AEAD (Sarkar 2011)

$\text{AEAD-5.Encrypt}_{K,\tau}(H,N,M)$ $V = \text{Hash}_\tau(H,N)$; $(R,Z) = \text{SC}_K(V)$; $C = M \oplus Z$; $\text{tag} = \text{Hash}_\tau(C) \oplus R$.	$\text{AEAD-6.Encrypt}_{K,K'}(H,N,M)$ $(\tau_1,\tau_2) = \text{SC}_K(K')$; $V = \text{Hash}_{\tau_1}(H,N)$; $(R,Z) = \text{SC}_K(V)$; $C = M \oplus Z$; $\text{tag} = \text{Hash}_{\tau_2}(C) \oplus R$.
$\text{AEAD-7.Encrypt}_{K,\tau}(H,N,M)$ $V = \text{Hash}_\tau(H,N)$; $(R,Z) = \text{SC}_K(V)$; $C = M \oplus Z$; $\text{tag} = \text{Hash}_\tau(M) \oplus R$.	$\text{AEAD-8.Encrypt}_{K,K'}(H,N,M)$ $(\tau_1,\tau_2) = \text{SC}_K(K')$; $V = \text{Hash}_{\tau_1}(H,N)$; $(R,Z) = \text{SC}_K(V)$; $C = M \oplus Z$; $\text{tag} = \text{Hash}_{\tau_2}(M) \oplus R$.

AEAD (Sarkar 2011)

Requires double-input hash function with low collision and differential probabilities.

- Efficient encoding methods have been proposed.
- Generic conversions from single-input to double-input (more generally, multiple-input) that is suitable for both Type-I and Type-II hash functions.
- Modifications of well-known hash functions such as Poly1305 and UMAC to handle double inputs.

Deterministic AEAD (Sarkar 2011)

$$
\begin{array}{|l|}
\hline
\text{DAEAD.Encrypt}_{K,\tau}(H, M) \\
\quad V = \text{Hash}_\tau(H, M); \\
\quad \text{tag} = \text{SC}_K(V); \\
\quad Z = \text{SC}_K(\text{tag}); \\
\quad C = M \oplus Z; \\
\quad \text{return } (C, \text{tag}). \\
\hline
\end{array}
$$

- Requires double-input hash functions.
- Suitable for Type-I hash functions.
- Extension to Type-II hash functions: Let K' be another n-bit key and produce τ as $\tau = \text{SC}_K(K')$.

"Provable" Security:

- All schemes described here have associated security proofs.
  - Make an appropriate idealised assumption on the underlying primitive (block or stream cipher).
  - Show that the adversary cannot do much more than try to defeat the assumption.
- Analysis is in the single-user setting.

Security

"Provable" Security:

- All schemes described here have associated security proofs.
  - Make an appropriate idealised assumption on the underlying primitive (block or stream cipher).
  - Show that the adversary cannot do much more than try to defeat the assumption.
- Analysis is in the single-user setting.

Multi-User Security:

- In the multi-user setting, an attack is successful if any one out of several keys is compromised.
- Using a single 128-bit key for the entire system may not offer 128-bit security (Chatterjee-Menezes-Sarkar, 2011).
- So, for attaining 128-bit security, the key length may possibly have to be greater than 128 bits.

Thank you for your attention!