Beetle Family of Lightweight and Secure Authenticated Encryption Ciphers

Avik Chakraborti [1], Nilanjan Datta [2], Mridul Nandi [3] and Kan Yasuda [1]

1. NTT Secure Platform Laboratories, Japan
2. Indian Institute of Technology, Kharagpur, India
3. Indian Statistical Institute, Kolkata, India

CHES, 2018

Sep 11, 2018

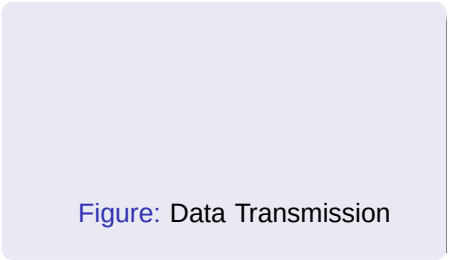

Figure: Data Transmission

A symmetric encryption scheme AE = (K ; E; D)

E : K M N A ! C

D : K C N A ! M [f?g

C set of tagged ciphertexts ((C; T) pair)

? : special symbol to denote reject

Nonce

Arbitrary number used only once for each encryption

Useful as initialization vectors. Example: Counter

Associated Data

Header of the Message (not encrypted but authenticated)

Example: IP Address

Why AE?

In practice both privacy and authenticity are desirable

A doctor wishes to send medical information about Alice to the medical database. Then

- We want data privacy to ensure Alice's medical records remain con dential
- We want integrity to ensure the person sending the information is really the doctor and the information was not modi ed in transit

We refer to this as authenticated encryption

Privacy

We want IND-CPA

Integrity

Adversary's goal: Receiver accepts a forged tuple $(C, T); N; A)$

INT-CTXT: Any forged tuple is rejected with high probability

Goal - IND-CPA + INT-CTXT

Adversary A runs in time t

A makes q_e enc queries ($_e$ enc blocks)

q_f o ine permutation queries to f or f^{-1} (simply f)

q_d forge queries ($_d$ forge blocks)

$$\mathrm{Adv}_E^{AE}(A) = {}_A((f\ ; E_K ; D_K);(f\ ;\$;?))$$

$\$$ returns a random string from the range set of E_K

$?$ oracle always returns $?$ (reject always)

$$\mathrm{Adv}_E^{AE}((q_e;q_f;q_d);(\ _e;\ _d);t) = \max_A \mathrm{Adv}_E^{AE}(A)$$

Designing Highly Secure Lightweight AE

- The mode should be very light
- It should provide su cient security level
- It should achieve better area-security trade-o among the existing designs

Several Ways of Designing AE

- Blockcipher(BC) based
- Streamcipher (SC) based
- Permutation based (Sponge) etc.

Our target: Highly Secure Lightweight AE

Best Choice: Sponge Based

- Sequential nonce-based AE
- b-bit state: r-bit rate + c-bit capacity (b = r + c)
- r-bit: process then feedback, c-bit: direct feedback

Introduced as a hash mode with Keccak[a] hash (SHA-3)

[a]Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche, Keccak, In EUROCRYPT 2013

Sponge based AE designed in Duplex mode[a]

c=2-bit AE security

[a]Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the sponge: Single-pass authenticated encryption and other applications, SAC 2011

Improved Bound (Jovanovic et al's Result)

Showed $\min\{b/2, c\}$-bit AE security[a] of Duplex sponge

Assumed number of decryption blocks $2^{c/2}$ (Impractical in real life)

Essentially $c/2$-bit security remains (considering decryption blocks)

[a] Philipp Jovanovic, Atul Luykx, and Bart Mennink, Beyond $2^{c/2}$ security in sponge- based authenticated encryption modes, ASIACRYPT 2014

Main Challenge of This Work

Main Di culty: Ciphertext is injected directly to the permutation

Can we stop that and increase the security adding simple tweaks in the design?

Possible Options for Feedback

Message Feedback: Current $M[i]$ is the feedback $X[i]$ for the next primitive call

Ciphertext Feedback: Current $C[i]$ is the feedback $X[i]$

Output Feedback: Previous primitive output $Y[i-1]$ is the feedback $X[i]$

Combined Feedback

Exactly one of $M[i]$, $C[i]$, $Y[i-1]$ can not compute $X[i]$. Adversary can not control $X[i]$ (by enc/ dec queries). Introduced in COFB[a]

[a] Avik Chakraborti, Tetsu Iwata, Kazuhiko Minematsu and Mridul Nandi, Blockcipher Based Authenticated Encryption: How small can we go?, CHES 2017

Beetle: Uses Combined Feedback in the rst bit

State Size

It needs only ab bits for storing the permutation f state

E ect of Combined Feedback

Each f output is processed with M using a combined feedback

(X ; C) = (Y ; M): X is in uenced by both Y and M

High security bound: due to feedback function, hard to forge

$\text{Const}_M = 1$ if $M \neq \varepsilon$ and n divides $|M|$, $\text{Const}_M = 2$ else

$(X;C) = (Y;M) = (_1(Y;M); Y M)$, where

$X = _1(Y;M) := G Y M, \quad C = I Y M$

Both G and $G + I$: Full rank matrix $6= I$

During decryption: $X = (G + I) Y + C$

Distinction of G and I makes combined feedback

$G : y = (y_1; y_2) ! (y_2; y_2 y_1)$ where $y_1; y_2 2 f 0; 1g^{r=2}$

E cient to implement ($r=2$-bit left shift + $r=2$-bit XOR)

$$G_{r r} = \begin{matrix} 0 & I \\ I & I \end{matrix}$$

Recommended Versions

Beetle[Light+]: Lightweight

Beetle[Secure+]: High security level

Underlyingf

For Beetle[Light+]: PHOTON[a] P_{144} with $b = 144; r = 64; c = 80$

For Beetle[Secure+]:PHOTON P_{256} with $b = 256; r = 128; c = 128$

[a] Jian Guo, Thomas Peyrin, and Axel Poschmann, The PHOTON family of lightweight hash functions, CRYPTO 2011

AE Security Bound

Nonce-respecting adversary

minf b/2, c - logr, rg bit AE security

Beetle[Light+] has 64-bit security

Beetle[Secure+] has 121-bit security

Table: Comparative Study on the State size and Security Trade-o . Assume c = b=2

Design	State size	Security
Beetle	b	b=2 logb=4
SpongeAE	b	b=4

Advantages

- Very low state size of b (b: state size)
- Very exible mode (any permutation f can be used)
- Inverse free
- Simple linear feedback
- Very lightweight and consumes low hardware area

Limitations

- Both the encryption and decryption are completely serial

a block AD, m block M (= 64-bit blocks, i.e, 8 byte blocks)

cycle count = 13(a + m) + 12 (In this calculation, we assume a = m)

cpb = $\frac{\text{cycle count}}{\text{len}}$, len is length of M in bytes (actually $\frac{13}{8}\frac{2m+12}{m} = 3:25 + \frac{1:5}{m}$)

Table: Clock cycles per message byte for Beetle[Light+] with = 64.

	Message length (Bytes)										
	8	16	24	32	64	128	256	512	1024	2048	16384
cpb	3.4375	3.3437	3.3125	3.2969	3.2734	3.2617	3.2559	3.2529	3.2514	3.2507	3.2500

- Serial processing of data
- Round-based architecture of PHOTON P_{144} permutation
- Processes 64 bits per 12 clock cycles
- Uses very low storage registers (only bit)
- Minimum hardware area among all the known implementations

Table: Beetle[Light+]. Not compatible with CAESAR API

Platform	# Slice Registers	# LUTs	# Slices	Frequency (MHZ)	Gbps	Mbps/ LUT	Mbps/ Slice
Vertex 6	185	616	252	381.592	1.879	3.050	7.369
Virtex 7	185	608	312	425.595	2.095	3.445	6.715

Table: Beetle[Secure+]

Platform	# Slice Registers	# LUTs	# Slices	Frequency (MHZ)	Gbps	Mbps/ LUT	Mbps/ Slice
Vertex 6	281	998	434	256.000	2.520	2.525	5.806
Vertex 7	305	1101	512	303.965	2.993	2.718	5.846

We admit our implementation does not follow CAESAR API

Scheme	Underlying Primitive	Security (in Bits)	# LUTs	# Slices	Gbps	Mbps/ LUT	Mbps/ Slice
Beetle[Light+]	Sponge(144, 64)	64	616	252	1.879	3.050	7.369
Ketje-JR	Sponge(200, 16)	96	1236	412	2.832	2.292	6.875
ASCON-128	Sponge(320, 64)	128	1274	451	3.118	2.447	6.914
JAMBU-SIMON96	BC(64)	48	1035	386	0.931	0.899	2.411
CLOC-TWINE80	BC(80)	32	1689	532	0.343	0.203	0.645
SILC-LED80	BC(80)	32	1684	579	0.245	0.145	0.422
SILC-PRESENT80	BC(80)	32	1514	548	0.407	0.269	0.743
COFB-AES	BC(128)	58	1075	442	2.850	2.240	6.450

Scheme	Underlying Primitive	Security (in Bits)	# LUTs	# Slices	Gbps	Mbps/ LUT	Mbps/ Slice
Beetle[Secure+]	Sponge(256, 128)	121	998	434	2.520	2.525	5.806
ASCON-128	Sponge(320, 64)	128	1274	451	3.118	2.447	6.914
NORX	Sponge(1024, 768)	128	5495	1724	24.524	4.463	9.139
Ketje-SR	Sponge(400, 32)	128	1903	613	5.772	3.033	9.416
Riverkeyak	Sponge(800, 544)	128	6234	1751	7.417	1.190	4.236
Lakekeyak	Sponge(1600, 1344)	128	19860	7130	12.603	0.635	1.768
Gibbon	Sponge(280, 40)	120	1807	653	1.280	0.708	1.960
Hanuman	Sponge(280, 40)	120	1769	626	0.693	0.392	1.107
ICEPOLE128a	Sponge(1280, 1024)	128	5734	1995	44.464	7.754	22.288

Beetle : Permutation based AE mode

Security level: min{b/2, c- log r, rg

Low area AE and can be used in low resource embedded devices