DryGASCON

Lightweight Cryptography Standardization Process round 1 submission[*]

Sébastien Riou

No institute given.

1 Introduction

This paper describes DryGASCON, an AEAD and hash algorithm based on DrySponge and ASCON [DEMS16].

DrySponge is a new construction derived from the Duplex Sponge construction [BDPA11]. It is designed to produce efficient implementations of AEAD and hash protected against side channels and fault attacks. It differs from the Duplex Sponge construction in the way it merge the input with the state and in the way it extract the output from the state, separating the concerns of cryptographic strength and robustness against physical attacks. This separation allows to prevent most of physical attacks at algorithmic level rather than at implementation level.

The ASCON permutation is limited to 128 bit security. To reach 256 bit security, a generalized variant, GASCON, is described.

Finally two fully defined variants named DryGASCON128 and DryGASCON256 are described and cryptanalysed.

The primary member of this submission is DryGASCON128.

2 Notation

2.1 Elementary Operation

$\overline{x}$ is the bitwise complement of x

$\oplus$ is the bitwise xor

$\&$ is the bitwise and

$|$ is the bitwise or

$/$ integer division

$a \ll$ is a shifted left by b bit, the lsb being filled with 0

$a \ggg$ is a rotated right by b bit

$a||b$ is the concatenation of a and b, a being the least significant part of the result

$a == b$ is the comparison of a and b. It returns 1 if they are equal, 0 overwise

$\lfloor x \rfloor$ rounding down

$\lceil x \rceil$ rounding up

[*]update of March 28th 2019

2.2 Block vs integers notation

In this document a block is a multi byte values interpreted with the little endian convention. We write block names using this style: `BLOCKNAME`. We write integer variable names using this style: *intname*. We write block values in raw hexadecimal strings:

```
Integer value: 0xffeeddccbbaa99887766554433221100
Block notation:  00112233445566778899aabbccddeeff
```

3 The DrySponge construction

The DrySponge construction aims at preventing most of physical attacks at the algorithmic level.

Like the original sponge construction, the security claims depends on the size of an internal storage called the capacity, noted c. Absorbing and squeezing phases can be freely interleaved and therefore most constructions built on top of the original sponge are applicable to DrySponge.

Unlike the original sponge contruction, domain separation is done via a dedicated input in the F function noted DS. The state include an optional x part which help separating the concerns of cryptographic strength and leakage resilience. A key setup function KS is generating the initial values of c and x. The final squeezing phase use a dedicated function noted G.

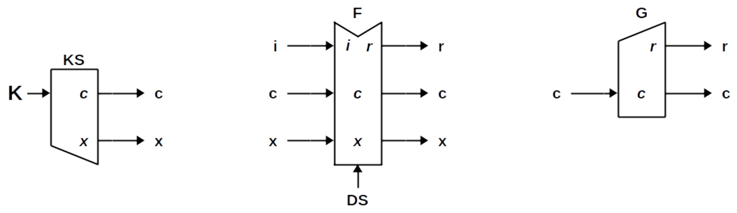

Figure 1: KS, F and G functions

Table 1 defines the relations between the widths of i, r, c and x:

Table 1: F and G inputs/outputs widths

$i.width$	≥ 1
$r.width$	$i.width$
$c.width$	$\geq r.width$
$x.width$	≥ 0
$DS.width$	≥ 1

Remark 1. The DS input is meant to carry information related to padding and transitions from one phase of a protocol to the next. It is a least 1 bit wide. DS is not represented in diagrams when its value is 0, which by convention means the continuation of the current phase.

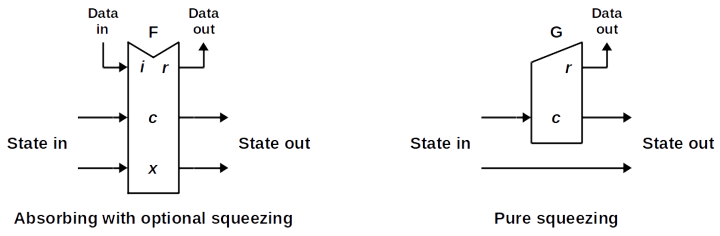

Figure 2: DrySponge absorb and squeez operations

3.1 Conceptual description of F and G functions

The main idea of DrySponge is to split the function F in three conceptual phases:

- Mix: It is the only one which process the input bits. It shall transform the capacity bits without injecting any known difference to them. It shall be a bijection of the capacity bits tweaked by the input bits.
- Core: It process only secret data (the capacity bits). It is responsible for the cryptographic strength of the algorithm. It is done iteratively, the number of iteration is a design parameter noted *rounds*.
- Accumulate: It is producing the output bits labelled as rate. It decorellates the output from the capacity bits which shall remain secret. It shall be purely linear.

The Core and Accumulate phases are grouped inside the G function. This allows to avoid the overhead of the Mix phase when there is no input to process (i.e. the squeez phases).

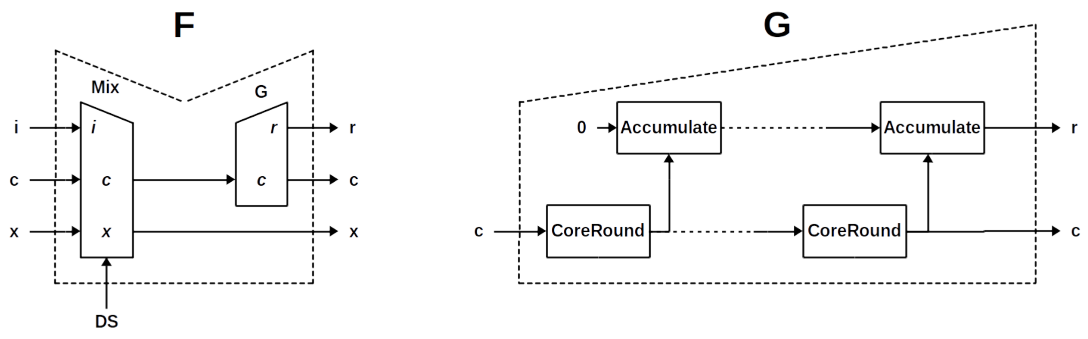

Figure 3: F and G functions details

3.2 F Description

The F function takes as input the state (c, x), the data input i and a domain separator DS. It outputs the modified state (c, x) and the data output r. Internally it uses a Mix function to inject i and DS into (c, x) and uses the G function to produce the output values.

The F function has the following design parameters

- $i.width$: the width of r and i.
- $c.width$: the width of c.
- $x.width$: the width of x.
- Mix: a function transforming $c.width+x.width$ bits according to $i.width+DS.width$ bits.

Algorithm 1 F function

```
Input: $(c, x)$ the secret state, $i$ the data input, $DS$ the domain separator.
Output: $(c, x)$, $r$
1: $(c, x) \leftarrow Mix(c, x, i, DS)$
2: $(c, r) \leftarrow G(c)$
3: return $(c, x, r)$
```

3.3 Mix Description

The Mix phase can be done in many ways, for that reason it is a design parameter of the DrySponge construction. A concrete instanciation is given in section 7.

The inputs and outputs are the following:

- It shall have an input i of width $i.width$.
- It shall have an input DS of width $DS.width$.
- It shall have an input c of width $c.width$.
- It shall have an output c of width $c.width$.
- It may have an input x of width $x.width$.
- It may have an output x of width $x.width$.

The constraints are the following:

- For a given (c, x) input, all possible i and DS shall result in a different (c, x) output.
- It shall not be possible to inject a known difference into (c, x) by manipulating i or DS.
- Bits from (c, x) shall be manipulated by large units to complicate side channel attacks. Ideally 32 bits or more.

The x input/output is there to allow algorithms based on the selection of a secret value according to the inputs i and DS. It is permitted to require some conditions on x, such as 'every byte shall be different'. x may be updated by Mix or remain unmodified.

3.4 G Description

The G function has the following design parameters

- *r.width*: the width of r.
- *c.width*: the width of c.
- *CoreRound*: a permutation or transformation operating on *c.width* bits.
- *rounds*: the number of calls to *CoreRound*.
- *Accumulate*: a purely linear function producing *r.width* bits from two inputs of *r.width* bits and *c.width* bits.

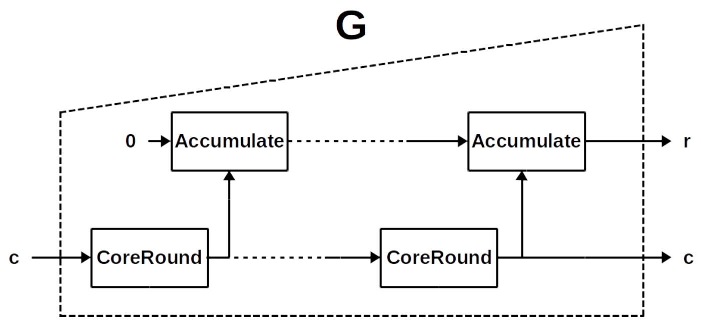

Figure 4: G function components

Algorithm 2 G function

```
Input: $c$ the capacity part of the state
Output: $c$, $r$
1: $r \leftarrow 0$
2: for $j \in 0, \cdots, rounds - 1$ do
3:     $c \leftarrow CoreRound(c)$
4:     $r \leftarrow Accumulate(r, c)$
5: end for
6: return $(c, r)$
```

3.4.1 CoreRound Description

The *CoreRound* function can be done in many ways, for that reason it is a design parameter of the DrySponge construction. A concrete instanciation is given in section 6.

CoreRound shall be a permutation or a transformation which include one or more non linear layer(s) and one or more linear layer(s). When iterated sufficiently it shall provide full diffusion: the toggling of one input bit shall toggle half of output bits on average. More generally *rounds* calls to *CoreRound* shall result in a cryptographically strong random permutation or random function.

3.4.2 Accumulate Description

The *Accumulate* function is simply the bitwise xor of all its inputs. The design parameter *AccumulateFactor* defines the relation between its outputs width, *r.width*, and its input width.

Algorithm 3 Accumulate function

```
Input: $c$ the capacity part of the state, $r$
Output: $r$
1: $nw \leftarrow r.width/32$
2: for $k \in 0, \cdots, AccumulateFactor$ do
3:     $cpart \leftarrow Sel(c, r.width, k)$
4:     for $i \in 0, \cdots, nw$ do
5:         $Sel(r, 32, i) \leftarrow Sel(r, 32, i) \oplus Sel(cpart, 32, (i + k) \mod nw)$
6:     end for
7: end for
8: return $r$
```

Remark 2. Other purely linear function could be used such as CRC.

Algorithm 4 Sel function

```
Input: $i$: multi bit input, $width$: number of bits of the output, $index$: scaled index
Output: $part$: bits from $i$ from $width * index$ to $width * (index + 1) - 1$
1: $part \leftarrow 0$
2: for $j \in 0, \cdots, width - 1$ do
3:     $bit \leftarrow (i \gg (width * index + j))\ \&\ 1$
4:     $part \leftarrow part \oplus (bit \ll j)$
5: end for
6: return $part$
```

4 AEAD mode of operation

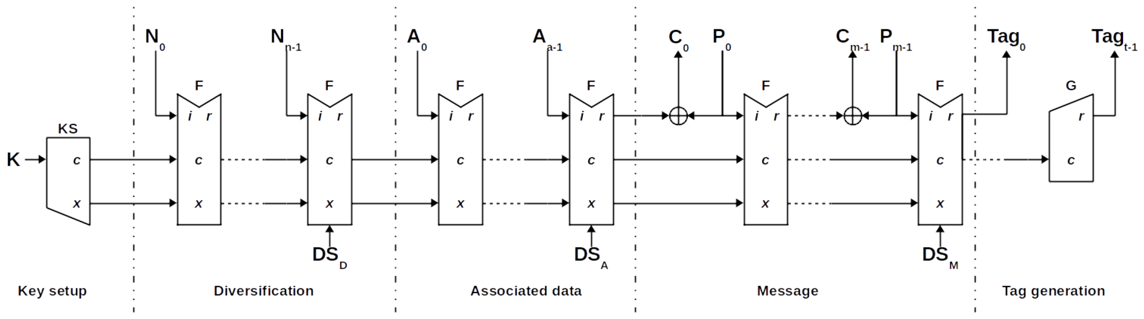

Figure 5: AEAD mode of operation

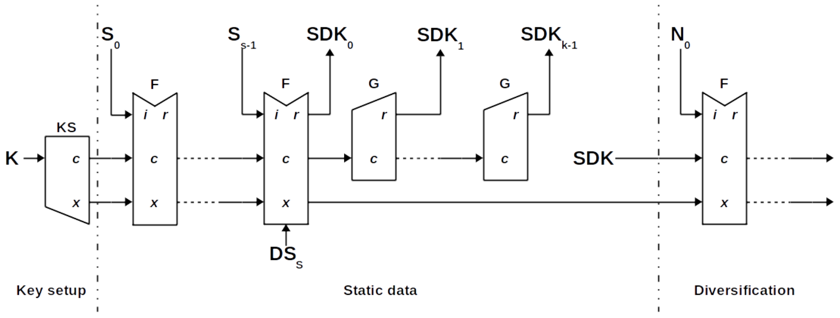

Figure 6: Optional Static Data phase

The AEAD mode use six different kind of data blocks:

1. Static data: optional data which can be precomputed. The blocks of static data are noted $\mathtt{S}_0$ to $\mathtt{S}_{s-1}$. The blocks of statically derived key are noted $\mathtt{SDK}_0$ to $\mathtt{SDK}_{k-1}$.
2. The blocks used in the diversification phase are noted $\mathtt{N}_0$ to $\mathtt{N}_{d-1}$. They are collectively referred as `NONCE`.
3. The associated data blocks are noted $\mathtt{A}_0$ to $\mathtt{A}_{a-1}$.
4. The plaintext blocks are noted $\mathtt{P}_0$ to $\mathtt{P}_{m-1}$.
5. The ciphertext blocks are noted $\mathtt{C}_0$ to $\mathtt{C}_{m-1}$.
6. The blocks generated in the tag generation phase are noted $\mathtt{Tag}_0$ to $\mathtt{Tag}_{t-1}$. They are collectively referred as `TAG`.

For a given security level, the secret key `K` is accepted in 3 different lengths. The 'small' key profile allows to use the minimum number of bits for the key. The 'fast' key profile allows a faster key setup and additional robustness against side channel attacks. The 'full' key profile provide the highest robustness against side channel attacks. Table 2 gives the relation between the target security level, noted *seclevel*, and the key width depending on the key profile.

Table 2: Key profiles and associated key widths

Profile	K width
Small	$seclevel$
Fast	$seclevel + x.width$
Full	$c.width + x.width$

4.1 Domain separators

The different phases of processing are separated by calls to F with non null domain separator.

Each domain separator has three fields:

- *pad*: 1 bit indicating if current input block is padded or not
- $final$: 1 bit indicating if tag generation starts next or not
- *domain*: 2 bits encoding the current domain:
  - 10: Static data
  - 01: Diversification
  - 10: Associated data
  - 11: Message

If static data length is 0, this phase is skipped entirely. If static data length is not 0 and is not a multiple of $i.width$, $DS_S.pad = 1$ and the static data are padded. $DS_S.pad = 0$ otherwise. The static data is never followed by finalization so $DS_S.final = 0$.

The length of `NONCE` is a design parameter which shall be enforced by implementation. $DS_D.pad = 0$ in all cases. If `NONCE` length is not a multiple of $i.width$ it is simply padded with 0 bits.

If associated data length is 0, this phase is skipped entirely. If associated data length is not 0 and is not a multiple of $i.width$, $DS_A.pad = 1$ and the associated data are padded. $DS_A.pad = 0$ otherwise.

If message length is 0, this phase is skipped entirely. If message length is 0 or not a multiple of $i.width$, $DS_M.pad = 1$ and the plaintext is padded. $DS_M.pad = 0$ otherwise.

The padding is the same for static data, associated data and message: a single 1 bit is appended and then followed by the smallest number of 0 bits needed to reach a length multiple of $i.width$.

Table 3 list few possibility of phases combination and the resulting number of calls to F in AEAD mode. We denote k the number $\lceil seclevel/r.width \rceil$.

Table 3: AEAD phases combination (non exhaustive)

s	a	m	phases	DS with final=1	calls to F
0	0	0	D,T	DS_D	$d + 1$
>0	0	0	S,D,T	DS_D	$s + k + d$
0	>0	0	D,A,T	DS_A	$d + a + 1$
0	0	>0	D,M,T	DS_M	$d + m + 1$
>0	>0	>0	S,D,A,M,T	DS_M	$s + k + d + a + m$

4.2 Number of rounds

The AEAD mode use a higher number of rounds for the diversification phase adn other phases. The rational is that the diversification phase can be entered in always the same state for a given key. We note *DRounds* the number of rounds for the diversification phase and *Rounds* the number of rounds for the other phases.

4.3 Algorithms

4.3.1 Design parameters

- Design parameters:
  - *i.width*: F function's i input width.
  - *r.width*: F and G functions' r input width.
  - *c.width*: F and G functions' c input/output width
  - *x.width*: F function's x input/output width
  - d: Width of `NONCE` in number of blocks of *i.width*
  - t: Width of `TAG` in number of blocks of *r.width*
  - *AccumulateFactor*: parameter for the *Accumulate* function
  - KS: Key setup function
  - *Mix*: Mix function
  - *CoreRound*: Core round function
  - *DRounds*: Number of rounds in the diversification phase
  - *Rounds*: Number of rounds in other phases

4.3.2 Padding Algorithm

Algorithm 5 Padding

```
Input: $\mathtt{B}_i$: one block or less
Output: $padded$: 1 if pading occured; $\mathtt{B}_o$: exactly one block
 1: $\mathtt{B}_o \leftarrow \mathtt{B}_i$
 2: $padded \leftarrow 0$
 3: if $length(\mathtt{B}_i) < i.width$ then
 4:     $padded \leftarrow 1$
 5:     for $i \in 0, \cdots, i.width - 2$ do
 6:         $\mathtt{B}_o \leftarrow \mathtt{B}_o || 0$
 7:     end for
 8:     $\mathtt{B}_o \leftarrow \mathtt{B}_o || 1$
 9: end if
10: return $(padded, \mathtt{B}_o)$
```

4.3.3 DomainSeparator Algorithm

Algorithm 6 DomainSeparator

```
Input: $padded, domain, finalize$
Output: $ds$
 1: $ds \leftarrow padded + (finalize \ll 1) + (domain \ll 2)$
 2: return $ds$
```

4.3.4 Absorb Algorithm

Algorithm 7 Absorb

```
Input: $(c, x, r)$: current state,$\mathtt{B}$: input blocks to process;$domain$,$finalize$
Output: $(c, x, r)$: new state
1: $b \leftarrow \left\lceil \frac{length(\mathtt{B})}{i.width} \right\rceil$
2: if $b > 0$ then
3:     for $i \in 0, \cdots, b-2$ do
4:         $(c, x, r) \leftarrow F(c, x, \mathtt{B}_i, 0)$
5:     end for
6:     $(padded, lastblock) \leftarrow Padding(\mathtt{B}_{b-1})$
7:     $ds \leftarrow DomainSeperator(padded, domain, finalize)$
8:     $(c, x, r) \leftarrow F(c, x, \mathtt{B}_{b-1}, ds)$
9: end if
10: return $(c, x, r)$
```

4.3.5 Squeez Algorithm

Algorithm 8 Squeez

```
Input: $(c, r)$: Current state, $remaining$: desired data length
Output: $\mathtt{B}$: output data
1: $len \leftarrow r.width$
2: while $remaining > 0$ do
3:     if $remaining < r.width$ then
4:         $len \leftarrow remaining$
5:         $r \leftarrow Sel(r, len, 0)$
6:     end if
7:     $\mathtt{B} \leftarrow \mathtt{B}||r$
8:     $remaining \leftarrow remaining - len$
9:     if $remaining > 0$ then
10:        $(c, r) \leftarrow G(c)$
11:    end if
12: end while
13: return $\mathtt{B}$
```

4.3.6 Encrypt Algorithm

- Inputs:
  - `K`: secret key
  - `S`: optional static data
  - `NONCE`: nonce
  - `A`: optional associated data
  - `P`: optional plain text data
- Outputs:
  - `C`: cipher text
  - `TAG`: authentication tag

Algorithm 9 AEAD.Encrypt

```
Input: K,S,NONCE,A,P
Output: C, TAG
1: $(c, x) \leftarrow KS(\mathtt{K})$
2: $(dss, dsd, dsa, dsm) \leftarrow (2, 1, 2, 3)$
3: $s \leftarrow \left\lceil \frac{length(\mathtt{S})}{i.width} \right\rceil$
4: $a \leftarrow \left\lceil \frac{length(\mathtt{A})}{i.width} \right\rceil$
5: $m \leftarrow \left\lceil \frac{length(\mathtt{P})}{i.width} \right\rceil$
6: $finalize \leftarrow 0$
7: if $s > 0$ then
8:     $(c, x, r) \leftarrow Absorb(c, x, r, \mathtt{S}, dss, finalize)$
9:     $\mathtt{SDK} \leftarrow Squeez(c, r, c.width)$
10:    $c \leftarrow \mathtt{SDK}$
11: end if
12: $finalize \leftarrow (a + m == 0)$
13: $(c, x, r) \leftarrow Absorb(c, x, r, \mathtt{NONCE}, dsd, finalize)$
14: $finalize \leftarrow m == 0$
15: $(c, x, r) \leftarrow Absorb(c, x, r, \mathtt{A}, dsa, finalize)$
16: if $m > 0$ then
17:     for $i \in 0, \cdots, m - 2$ do
18:         $\mathtt{C}_i \leftarrow \mathtt{P}_i \oplus r$
19:         $(c, x, r) \leftarrow F(c, x, \mathtt{P}_i, 0)$
20:     end for
21:     $\mathtt{C}_{m-1} \leftarrow Sel(\mathtt{P}_{m-1} \oplus r, length(\mathtt{C}) \mod r.width, 0)$
22:     $(padded, lastblock) \leftarrow Padding(\mathtt{P}_{m-1})$
23:     $finalize \leftarrow 1$
24:     $DS_M \leftarrow DomainSeperator(padded, dsm, finalize)$
25:     $(c, x, r) \leftarrow F(c, x, \mathtt{P}_{m-1}, DS_M)$
26: end if
27: $\mathtt{TAG} \leftarrow Squeez(c, r, t)$
28: return $(\mathtt{C}, \mathtt{TAG})$
```

4.3.7 Decrypt and Verify Algorithm

- Inputs:
  - `K`: secret key
  - `S`: optional static data
  - `NONCE`: nonce
  - `A`: optional associated data
  - `C`: optional cipher text data
  - `TAG`: authentication tag
- Outputs:
  - `P`: plain text data

Algorithm 10 AEAD.Decrypt

```
Input: K,S,NONCE,A,C,TAG
Output: P or $Failure$
1: $(c, x) \leftarrow KS(\mathtt{K})$
2: $(dss, dsd, dsa, dsm) \leftarrow (2, 1, 2, 3)$
3: $s \leftarrow \left\lceil \frac{length(\mathtt{S})}{i.width} \right\rceil$
4: $a \leftarrow \left\lceil \frac{length(\mathtt{A})}{i.width} \right\rceil$
5: $m \leftarrow \left\lceil \frac{length(\mathtt{C})}{i.width} \right\rceil$
6: $finalize \leftarrow 0$
7: if $s > 0$ then
8:     $(c, x, r) \leftarrow Absorb(c, x, r, \mathtt{S}, dss, finalize)$
9:     $\mathtt{SDK} \leftarrow Squeez(c, r, c.width)$
10:    $c \leftarrow \mathtt{SDK}$
11: end if
12: $finalize \leftarrow (a + m == 0)$
13: $(c, x, r) \leftarrow Absorb(c, x, r, \mathtt{NONCE}, dsd, finalize)$
14: $finalize \leftarrow m == 0$
15: $(c, x, r) \leftarrow Absorb(c, x, r, \mathtt{A}, dsa, finalize)$
16: if $m > 0$ then
17:     for $i \in 0, \cdots, m - 2$ do
18:         $\mathtt{P}_i \leftarrow \mathtt{C}_i \oplus r$
19:         $(c, x, r) \leftarrow F(c, x, \mathtt{P}_i, 0)$
20:     end for
21:     $\mathtt{P}_{m-1} \leftarrow Sel(\mathtt{C}_{m-1} \oplus r, length(\mathtt{P}) \mod r.width, 0)$
22:     $(padded, lastblock) \leftarrow Padding(\mathtt{P}_{m-1})$
23:     $finalize \leftarrow 1$
24:     $DS_M \leftarrow DomainSeperator(padded, dsm, finalize)$
25:     $(c, x, r) \leftarrow F(c, x, \mathtt{P}_{m-1}, DS_M)$
26: end if
27: if $\mathtt{TAG} == Squeez(c, r, t)$ then
28:     return P
29: end if
30: return $Failure$
```

4.4 Security claims

Table 4 present the security claims in AEAD mode:

Table 4: AEAD mode security claims

Requirement	security level in bits
Confidentiality of plaintext	$2^{c.width/2}$
Integrity of plaintext	$min(2^{\texttt{TAG}.width}, 2^{c.width/2})$
Integrity of associated data	$min(2^{\texttt{TAG}.width}, 2^{c.width/2})$
Integrity of static data	$min(2^{\texttt{TAG}.width}, 2^{c.width/2})$
Integrity of `NONCE`	$min(2^{\texttt{TAG}.width}, 2^{c.width/2})$

Assumptions:

- `NONCE` is not reused for a given key.
- no more than $2^{\texttt{TAG}.width/2}$ calls to G are made for a given key.
- no structural attack with complexity less than $2^{c.width/2}$ exists on F and G.

We claim that a careful implementation in hardware can achieve robustness against side channel attacks and fault attacks [1] without the addition of any implementation level randomization.

4.5 Classic security analysis

The security analysis shall be done on the concrete instantiation since the cryptographic properties of the F and G functions depend completly on abstract functions Mix, $CoreRound$ which are considered as design parameters. This section is therefore limited to the generic security aspects of the DrySponge construction: what can be said if the abstract sub functions fulfills their contract.

4.5.1 Access to inputs and outputs

In the AEAD mode of operation, F's i input is always controllable by the attacker and (c, x) is always secret. DS input is somewhat controllable: the attacker can modify it but not independantly of other inputs. r is observable only during the message and `TAG` generation phases.

4.5.2 Differentials and Linear propagation

Provided that the Mix function fulfils its contract, the construction prevent the propagation of known differentials to (c, x) and therefore to the G function.

A differential attack on the G function is nevertheless possible by working backwards from differences on the r output. The goal would be to deduce a known difference on c at G input stage and then deduce some bits of c by going backwards again, up to i through the Mix function.

In conclusion, despite the shielding from the Mix function, the G function shall be designed to resist the 'backward differential attack' described above.

4.6 General purpose countermeasures against physical attacks

This section discuss aspects of the DrySponge construction which enhance robustness against most physical attacks.

[1]except safe error attacks

4.6.1 Key extra width

The secret key is allowed to be wider than mathematically required to achieve the target security level. This provide security margin against physical attacks simply because there are more bits to guess.

4.6.2 Precomputation of static data phase

The processing of static data before `NONCE` produce the statically derived key SDK. If a side channel or fault attack based on `NONCE` variation is practical it can recover only SDK, not the original key. A master key can therefore be deployed safely on a lot of device simply by adding a public device ID as static data. Furthermore this does not cost any runtime as this can be precomputed. Note that storing such precomputed state instead of the key also protects against any kind of key recovery attack: the key is simply not in the device. Such mechanism can be added at protocol levev however inserting it within the AEAD description makes it more likely to be used in practice.

4.7 Security against Side channel attacks

4.7.1 Attacks in the chosen plaintext or chosen ciphertext scenario

An attacker who controls the input i may attempt a DPA or template attack to recover c as it is manipulated by the Mix function. Those attacks cannot be discussed without a concrete specification of the Mix function, refer to section 7.

4.7.2 DPA in the known ciphertext scenario

An attacker who observes r may attempt a DPA to recover c in the last rounds of G. Since the final r value is produced by the xor of several unknown variable values, such DPA is not possible.

4.7.3 Template attack in the known ciphertext scenario

An attacker who observes r may attempt a template attack to recover c. Such template attack is considered unlikely to succeed as the final r value is produced by the xor of several unknown variable values as wide as r. If an implementer wish to add a countermeasure, the bitwise xor is trivial to protect by using boolean masking.

4.7.4 Blind Side-Channel Analysis

Linge and al. introduced the concept of joint distribution analysis [LDLL13]. This methods allows to perform so called 'blind side channel analysis' i.e. side channel attack without access to the input nor the output of the targeted algorithm. This method is well suited to target the core phase of the DrySponge construction and it cannot be prevented at algorithmic level. As a result, the implementation shall be protected against such attack.

Note that simple countermeasures like shuffling and dummy operations are sufficient to prevent that kind of attacks. Another way to make such attack unpractical is to do the implementation in such a way that secret values are always manipulated by wide units such as 32 or 64 bits. As the attacker needs to go through all possible values for its targeted intermediate, a width of 32 bit already provides a large security margin. To the best of our knownledge side channel attacks with more than 100 millions traces have not been attempted. As a result, costly countermeasures such as masking can be avoided on the $CoreRound$ function.

4.8 Security against fault attacks

4.8.1 Introduction

Over the recent years many new fault attacks have been published. We address all but one at the protocol level. The one not addressed is called "safe error attack". We cannot address it at the algorithm level because it target the storage of key bits (or intermediate value bits): For example if one knows how to force a given key bit to 0, it is trivial to check wherever forcing this bit is corrupting AEAD decryption tag. If the tag is not corrupted, the attacker knows the bit was already 0. If the tag is corrupted, the attacker knows that the bit was not 0, so it was 1. The attack is therefore linked to a particular implementation and the thorough characterization of it by the attacker, the algorithm level has no way to prevent this. In the remaining of this document, "fault attacks" shall mean "published fault attacks except safe error attacks".

4.8.2 Leveraging AEAD

In AEAD encryption, fault attacks are prevented by deriving a unique key for each output. To learn the key successfully, the attacker would need a method which give the full key after a single invocation. To the best of our knowledge, such attack does not exist.

In AEAD decryption, the same invocation can be repeated over and over since the attacker control the nonce. As a result, the argumentation given for AEAD encryption does not hold. DFA is prevented by ensuring that all outputs (plain text blocks) have to be correct in order to get the correct tag. Note that this is not the case for all AEAD algorithms, for example AES-GCM and ASCON do not have this property (a fault in plaintext can be cancelled out via modification of the ciphertext).

Statistical Fault Attacks (SFA), introduced by Fuhr and al. in [FJLT13], are prevented in the same way DFA is prevented.

Statistical Inefective Fault attacks (SIFA), introduced by Dobraunig and al. in [DEK$^+$18], are still possible so we provide an algorithmic countermeasures to make them impractical.

4.8.3 Fighting SIFA

SIFA aims at producing a bias in an internal variable which propagates to the output. Unlike DFA, SIFA works by observing correct outputs only, it is therefore difficult to prevent at implementation level. In DrySponge, due to the additive nature of the computation of r, any bias introduced in one round is masked by the other rounds. As a result SIFA attacks are not possible on the outputs of the DrySponge construction.

SIFA is also possible on the input side as described in [DMMP18]. In DrySponge such attack is prevented by the Mix function as a bias in c does not propagate back to i.

5 Hash mode of operation

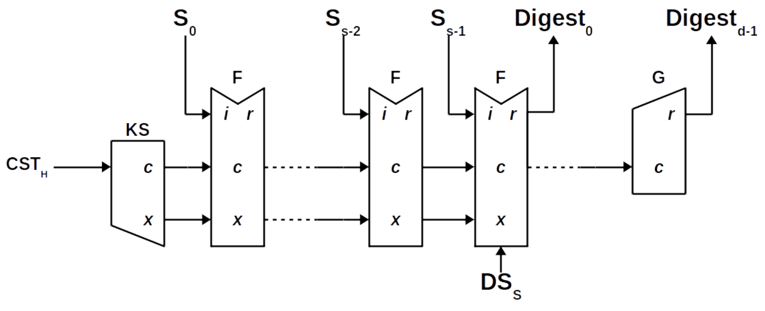

Figure 7: Hash mode of operation

Table 5 defines the constant used in hash mode:

Table 5: Hash constant CST_H as byte array

	byte index mod 8							
	0	1	2	3	4	5	6	7
0	0x24	0x3f	0x6a	0x88	0x85	0xa3	0x08	0xd3
1	0x13	0x19	0x8a	0x2e	0x03	0x70	0x73	0x44
2	0xa4	0x09	0x38	0x22	0x29	0x9f	0x31	0xd0
3	0x08	0x2e	0xfa	0x98	0xec	0x4e	0x6c	0x89
4	0x45	0x28	0x21	0xe6	0x38	0xd0	0x13	0x77
5	0xbe	0x54	0x66	0xcf	0x34	0xe9	0x0c	0x6c
6	0xc0	0xac	0x29	0xb7	0xc9	0x7c	0x50	0xdd
7	0x3f	0x84	0xd5	0xb5	0xb5	0x47	0x09	0x17
8	0x92	0x16	0xd5	0xd9	0x89	0x79	0xfb	0x1b
9	0xd1	0x31	0x0b	0xa6	0x98	0xdf	0xb5	0xac
10	0x2f	0xfd	0x72	0xdb	0xd0	0x1a	0xdf	0xb7
11	0xb8	0xe1	0xaf	0xed	0x6a	0x26	0x7e	0x96
12	0xba	0x7c	0x90	0x45	0xf1	0x2c	0x7f	0x99
13	0x24	0xa1	0x99	0x47	0xb3	0x91	0x6c	0xf7
14	0x08	0x01	0xf2	0xe2	0x85	0x8e	0xfc	0x16
15	0x63	0x69	0x20	0xd8	0x71	0x57	0x4e	0x69

Remark 3. CST_H is the 128 most significant bytes of the fractional part of π. It is truncated to $seclevel + x.width$ before entering the KS function where $seclevel$ is the desired security level in bits.

The padding is the same as in the AEAD mode: a single 1 bit is appended and then followed by the smallest number of 0 bits needed to reach a length multiple of $i.width$.

The *Rounds* parameter for all calls to F and G is identical and equal to the parameter *Rounds* defined for AEAD mode.

5.1 Security claims

Table 6 present the security claims in hash mode:

Table 6: Hash mode security claims

Requirement	security level in bits
Integrity of data	$min(2^{\texttt{DIGEST}.width/2}, 2^{c.width/2})$

Assumptions:

- no more than $2^{\texttt{DIGEST}.width/4}$ calls to G are made for a given message.
- no structural attack with complexity less than $2^{c.width/2}$ exists on F and G.

6 $GASCON$ function

This section describes $GASCON$, a concrete instanciation of the $CoreRound$ function. It is used in the named instances of this submission. As the name suggest, it is based on the ASCON permutation round [DEMS16]. '$GASCON$' stands for 'Generalized ASCON', its purpose is to support wider state size than the original ASCON permutation function.

6.1 Specification

6.1.1 Input, output and design parameters

- Inputs:
  - c: input state
  - $round$: round number
- Outputs:
  - c: output state
- Design parameters:
  - $cwords$: number of 64 bit words.
    - $cwords \geq 5$
    - $cwords \equiv 1 \mod 2$

We note the fully parameterized function as $GASCON_{CyRx}$ with x replaced by $rounds$ and y replaced by $cwords$. For example $GASCON_{C5R12}$ is the instance equivalent to the original ASCON permutation round.

The design parameters fix the width of c and internal constants according to the following equations.

$$c.width = cwords * 64 \tag{1}$$

$$mid = (cwords - 1)/2 \tag{2}$$

6.1.2 Algorithm

Algorithm 11 GASCON

```
Input: $c$,$round$
Output: $c$
1: $Sel(c, 64, mid) \leftarrow Sel(c, 64, mid) \oplus (((0xF - round) \ll 4) \mid round)$
2: $c \leftarrow sbox(c)$
3: $c \leftarrow linlayer(c)$
4: return $c$
```

Algorithm 12 sbox

```
Input: $c$
Output: $c$
 1: for $i \in 0, \cdots, mid$ do
 2:     $d \leftarrow 2 * i$
 3:     $s \leftarrow (cwords + d - 1) \mod cwords$
 4:     $Sel(c, 64, d) \leftarrow Sel(c, 64, d) \oplus Sel(c, 64, s)$
 5: end for
 6: for $i \in 0, \cdots, cwords - 1$ do
 7:     $s \leftarrow (i + 1) \mod cwords$
 8:     $ws \leftarrow Sel(c, 64, s)$
 9:     $wi \leftarrow Sel(c, 64, i)$
10:     $Sel(t, 64, i) \leftarrow (\sim wi)\ \&\ ws$
11: end for
12: for $i \in 0, \cdots, cwords - 1$ do
13:     $s \leftarrow (i + 1) \mod cwords$
14:     $Sel(c, 64, i) \leftarrow Sel(c, 64, i) \oplus Sel(t, 64, s)$
15: end for
16: for $i \in 0, \cdots, mid$ do
17:     $s \leftarrow 2 * i$
18:     $d \leftarrow (s + 1) \mod cwords$
19:     $Sel(c, 64, d) \leftarrow Sel(c, 64, d) \oplus Sel(c, 64, s)$
20: end for
21: $Sel(t, 64, mid) \leftarrow \sim Sel(t, 64, mid)$
22: return $c$
```

Algorithm 13 linlayer

```
Input: $c$
Output: $c$
1: $lut0 \leftarrow$ [19,61,1,10, 7,31,53, 9,43]
2: $lut1 \leftarrow$ [28,38,6,17,40,26,58,46,50]
3: for $j \in 0, \cdots, cwords - 1$ do
4:     $w \leftarrow Sel(c, 64, j)$
5:     $r0 \leftarrow lut0(j)$
6:     $r1 \leftarrow lut1(j)$
7:     $Sel(c, 64, j) \leftarrow Sel(c, 64, j) \oplus BiRotR(w, r0) \oplus BiRotR(w, r1)$
8: end for
9: return $c$
```

Remark 4. The five first values in $lut0$ and $lut1$ match the rotation of ASCON's $\sum_i$ except 38 and 40. The remaining values have been selected from π decimal digits. The numbers already present in the list and 32 have been discarded. The numbers which would result in two even shifts of a word have been discarded, so $lut1$ is always odd where $lut0$ is even.

Algorithm 14 BiRotR: 64 bit rotation in bit interleaved format

```
Input: $in$: 64 bit value in bit interleaved format, $shift$
Output: $out$: $in$ rotated right by $shift$
1: $i0 \leftarrow in \,\&\, \text{0xFFFFFFFF}$
2: $i1 \leftarrow in \gg 32$
3: $shift2 = \lfloor shift/2 \rfloor$
4: if $shift \,\&\, 1$ then
5:     $t \leftarrow i1 \ggg shift2$
6:     $i1 \leftarrow i0 \ggg ((shift2 + 1) \mod 32)$
7:     $i0 \leftarrow t$
8: else
9:     $i0 \leftarrow i0 \ggg shift2$
10:    $i1 \leftarrow i1 \ggg shift2$
11: end if
12: return $(i1 \ll 32) \mid i0$
```

Remark 5. The fact that for each word has exactly one rotation with an odd shift ensure that a difference contained in half an input word will be propagated to the other half of the corresponding output word. We refer to this property as the 'intra word diffusion' property.

6.1.3 Comparison with original ASCON permutation

The differences between $GASCON_{C5R12}$ and the original ASCON are listed below:

- byte order is little endian
- Each 64 bit word is in bit interleaved representation
- Shift 39 replaced by 38
- Shift 41 replaced by 40
- Constants for round addition does not depend on the total number of rounds (the first round of $GASCON_{C5R12}$ use the same constant as the first round of $GASCON_{C5R11}$ or $GASCON_{C5R7}$).

$GASCON_{C5R12}$ is therefore closely related to ASCON permutation round function. It follows that all cryptanalysis result on ASCON permutation [DEMS15] may be reused to study $GASCON$.

Remark 6. The best differentials and some other properties are different but any tool used to cryptanalyse ASCON can be trivially adapted to cryptanalyse $GASCON_{C5Rx}$.

7 $MixSX32$ function

This section describes $MixSX32$, a function used to build a concrete instanciation of the Mix function. It is used in the named instances of this submission. $MixSX32$ stands for 'Mix by Selecting and Xoring 32 bit values'.

7.1 Specification

7.1.1 Input, output and design parameters

- Inputs:
  - c: secret data
  - x: secret data which constrained: each 32 bit word shall be unique within x.
  - d: input data
- Outputs:
  - c: secret data output
- Design parameters:
  - $cwords64$
  - $xwords32$

The design parameters fix the width of c, x and d according to the following equations. Equation 5 gives the width of the internal variable idx used in algorithm 15.

$$c.width = cwords64 * 64 \tag{3}$$
$$x.width = xwords32 * 32 \tag{4}$$
$$idx.width = \log_2(xwords32) \tag{5}$$
$$d.width = cwords64 * idx.width \tag{6}$$

7.1.2 Algorithm

Algorithm 15 MixSX32

```
Input: $c$,$x$,$d$
Output: $c$
1: for $j \in 0, \cdots, cwords64 - 1$ do
2:     $idx \leftarrow Sel(d, idx.width, j)$
3:     $xw \leftarrow Sel(x, 32, idx)$
4:     $Sel(c, 32, 2 * j) \leftarrow Sel(c, 32, 2 * j) \oplus xw$
5: end for
6: return $c$
```

7.2 Correctness

In the section we check that $MixSX32$ is a suitable Mix function. For that purpose we set $d = i||DS$.

The constraint on x ensures that xw is different for each possible value of idx. It follows that for given inputs c and x, $MixSX32$ output is different for every d input (and therefore every i and DS).

As long as c and x are secret, $MixSX32$ does not allow to know the difference of two outputs for any pair of input (d_1, d_2).

Finally $MixSX32$ manipulate c and x only by 32 bits units, it therefore meets all the requirements for the Mix functions.

7.3 Security analysis

The $MixSX32$ function is not supposed to bring any cryptographic strength in the DrySponge construction. We just list some properties in this section.

$MixSX32$ is linear with respect to c:

$$MixSX32(c1 \oplus c2 \oplus c3, x, d) = MixSX32(c1, x, d) \oplus MixSX32(c2, x, d) \oplus MixSX32(c3, x, d) \tag{7}$$

$MixSX32$ is linear with respect to x:

$$MixSX32(c, x1 \oplus x2 \oplus x3, d) = MixSX32(c, x1, d) \oplus MixSX32(c, x2, d) \oplus MixSX32(c, x3, d) \tag{8}$$

Remark 7. $MixSX32$ allow to inject the same difference in two or more 64 bits words of c.

7.4 Protection against side channel attacks

The $MixSX32$ transform secret data according to the inputs d which is controllable by the attacker. It is therefore a target for DPA and shall be implemented with care. Fortunately the chosen transformation, composed only of selection and xor of 32 bit values is easy to protect against side channels both in software and in hardware.

The setup is defavorable for the attacker: only $idx.width$ controllable bits impact each 64 bit word of c. Practical values for $idx.width$ ranges from 2 to 5. The attacker is therefore able to get only a limited number of traces with different values for a given 64 bit word of c. In addition the values being xored are both unknown. It follows that a DPA attack targeting c or x is not practical.

Since the xor is a linear operator and the operands are 32 bits wide, 'blind side channel analysis' is not applicable to target c or x.

One could attempt a template attack on each $idx.width$ bits subsets of a 32 bit word in c. For each subset the remaining bits in the same word would be ignored and would therefore generate noise in measurements. Crucially, unlike noise from all other parts of the target circuit, this noise depends on the specific value of the targeted word in c. As a result the noise is not uniform and does not cancel out over a large number of measurements. It follows that such template attack is unlikely to succeed. In any case such attack does not recover the upper half of each 64 bit words of c since they are not touched at all by $MixSX32$.

Finally an attacker can set out to guess the value of d by template attack. Such attack does not reveal the key but gives information about the plaintext in AEAD mode. Hardware implementation can reduce the likelyhood of success by processing several words in parallel. In software implementation that point shall be carefully evaluated, it is likely to require randomization of words indexes in c and x and randomization of the order of processing of the words.

7.5 Protection against fault attacks

As explained in section 4.8 faults attacks only SIFA is relevant for the Mix function. A bias in the output of $MixSX32$ can be mapped to particular bits of d however this does not give any information about the related word in c. The only information leaked is that the related words in x have an unknown number of bits which match.

8 *Mix*128 function

This section describes *Mix*128, the *Mix* function used in the named instances of this submission. It is built on top of *MixSX*32 and *GASCON* functions.

8.1 Specification

8.1.1 Input, output and design parameters

- Inputs:
  - c: secret data
  - x: secret data which constrained: each 32 bit word shall be unique within x.
  - i: 128 bit input data
- Outputs:
  - c: secret data output
- Design parameters:
  - $cwords64$
  - $xwords32$

The design parameters fix the width of c, x and internal constants according to the following equations.

$$c.width = cwords64 * 64 \tag{9}$$
$$x.width = xwords32 * 32 \tag{10}$$
$$idx.width = \log_2(xwords32) \tag{11}$$
$$d.width = cwords64 * idx.width \tag{12}$$
$$mixrounds = \left\lceil \frac{i.width + 4}{d.width} \right\rceil \tag{13}$$

8.1.2 Algorithm

Algorithm 16 Mix128

```
Input: $c$,$x$,$i$,$DS$
Output: $c$
1: $i \leftarrow i \oplus (DS \ll 128)$
2: for $j \in 0, \cdots, mixrounds - 2$ do
3:     $d \leftarrow Sel(i, d.width, j)$
4:     $c \leftarrow MixSX32(c, x, d)$
5:     $c \leftarrow GASCON(c, 0)$
6: end for
7: $d \leftarrow Sel(i, d.width, mixrounds - 1)$
8: $c \leftarrow MixSX32(c, x, d)$
9: return $c$
```

8.2 Correctness

This function rely on $MixSX32$ for processing i and DS piece by piece. As a result it meets the requirements to not insert known difference in c and to manipulate c and x by 32 bit units.

$Mix128$ output is different for each i and DS. This is garanteed by the combination of two properties:

- $GASCON$ intra word diffusion ensures that a difference in the lower 32 bit of a word propagates to the upper 32 bit of at least one output word.
- $MixSX32$ modifies only the lower half of each 64 bit word.

As a result, a difference introduced by a call to $MixSX32$ cannot be cancelled out by a subsequent call to $MixSX32$ if a call to $GASCON$ is interleaved.

8.3 Security analysis

The $Mix128$ function is not supposed to bring any cryptographic strength in the DrySponge construction. We just list some properties in this section.

Remark 8. $Mix128$ allow to inject the same difference in two or more 64 bits words of c in all calls to $MixSX32$ since x is never updated.

9 $KSneq32$ function

This section describes $KSneq32$, a concrete instanciation of the KS function. It meets the requirement on $Mix128$ input and it is used in the named instances of this submission.

9.1 Specification

9.1.1 Input, output and design parameters

- Inputs:
  - K: Secret key
  - $width$: Width of K
- Outputs:
  - c: initial value of the capacity part of the state
  - x: initial value of x part of the state
- Design parameters:
  - $minwith$: minimum width of K. $minwidth \equiv 0 \mod 32$
  - $c.with$: width of c. $c.width \equiv 0 \mod 32$
  - $x.width$:
    - $x.width \equiv 0 \mod 32$
    - $x.width < c.width$

The design parameter fix internal constants according to the following equations.

$$(x.width) \mapsto xwords = x.width/32 \tag{14}$$

$$(minwidth, x.width) \mapsto fastwidth = minwidth + x.width \tag{15}$$

$$(c.width, x.width) \mapsto fullwidth = c.width + x.width \tag{16}$$

9.1.2 Algorithm

Algorithm 17 KSneq32

```
Input: $K$ secret key, $width \in [minwidth, fastwidth, fullwidth]$
Output: $(c, x)$
 1: if $width == fullwidth$ then
 2:   $c \leftarrow Sel(K, c.width, 0)$
 3:   $x \leftarrow K \gg c.width$
 4: else
 5:   $kwords \leftarrow minwidth/32$
 6:   for $i \in 0, \cdots, c.width/32$ do
 7:     $Sel(c, 32, i) \leftarrow Sel(K, 32, i \mod kwords)$
 8:   end for
 9:   if $width == fastwidth$ then
10:     $x \leftarrow K \gg minwidth$
11:   else
12:     repeat
13:       $c \leftarrow GASCON(c, 0)$
14:       $match = 0$
15:       for $i \in 0, \cdots, xwords - 1$ do
16:         for $j \in i + 1, \cdots, xwords$ do
17:           if $Sel(c, 32, i) == Sel(c, 32, j)$ then
18:             $match = 1$
19:           end if
20:         end for
21:       end for
22:     until $match == 0$
23:     $x \leftarrow Sel(c, x.width, 0)$
24:     $Sel(c, x.width, 0) \leftarrow Sel(K, x.width, 0)$
25:   end if
26: end if
27: return $(c, x)$
```

Remark 9. The instance of $GASCON$ function to use shall be the same selected in the G function.

Remark 10. Since $x.width < c.width$, c after line 24 is garanteed to be different from c in the first iteration of line 13. It follows that $x \neq c$.

10 Specification of DryGASCON128

It process blocks of 128 bits and target a security level of 128 bits.

10.1 Supported key sizes

Table 7: DryGASCON128 key sizes

Profile	size
Small	128
Fast	256
Full	448

10.2 Security Claims for AEAD mode

Table 8: DryGASCON128 AEAD mode security claims

Requirement	security level in bits
Confidentiality of plaintext	128
Integrity of plaintext	128
Integrity of associated data	128
Integrity of static data	128
Integrity of `NONCE`	128

Conditions:

- `NONCE` is not reused for a given key.
- no more than 2^{64} bytes are encrypted with a given key (including `NONCE` bytes).

10.3 Security claims for hash mode

Table 6 present the security claims in hash mode:

Table 9: DryGASCON128 Hash mode security claims

Requirement	security level in bits
Integrity of data	128

Conditions:

- Message length is less than 2^{64} bytes.

10.4 Parameters

Table 10: DryGASCON128 parameters

Name	value
$i.width$	128
$r.width$	128
$c.width$	320
$x.width$	128
d	1
t	1
$AccumulateFactor$	2
KS	$KSneq32$
Mix	$Mix128$
$CoreRound$	$GASCON_{C5}$
$DRounds$	11
$Rounds$	7

10.4.1 Sbox

The table below gives the values in hexadecimal. The number in front of each row indicate the value of the most significant bit of the input. The column number gives the value of the four least significant input bits. For example S(0x01) is 0x0f and S(0x10) is 0x1a.

Table 11: DryGASCON128 sbox

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	04	0f	1b	01	0b	00	17	0d	1f	1c	02	10	12	11	0c	1e
1	1a	19	14	06	15	16	18	0a	05	0e	09	13	08	03	07	1d

Let $x4, x3, x2, x1, x0$ and $y4, y3, y2, y1, y0$ be the 5-bit input and output of the S-box, where $x4$ refers to the most significant bit or the first register word of the S-box. Then the algebraic normal form (ANF) of the S-box is given by:

$$y0 = x4x1 + x3 + x2x1 + x2 + x1x0 + x1 + x0 \quad (17)$$

$$y1 = x4 + x3x2 + x3x1 + x3 + x2x1 + x2 + x1 + x0 \quad (18)$$

$$y2 = x4x3 + x4 + x2 + x1 + 1 \quad (19)$$

$$y3 = x4x0 + x4 + x3x0 + x3 + x2 + x1 + x0 \quad (20)$$

$$y4 = x4x1 + x4 + x3 + x1x0 + x1 \quad (21)$$

10.5 Sbox security analysis

DryGASCON128 sbox is the same as ASCON except the order of the input bits. The differential distribution and linear approximation have been computed. The associated tables and 'heat' maps are given in appendix 14.1.

In [DEMS16], the following has been established:

- Its algebraic degree is 2.
- The maximum differential probability of the S-box is 2^{-2}.
- Its differential branch number is 3.
- The maximum linear probability of the S-box is 2^{-2}.

- Its linear branch number is 3.
- This S-box is weak in terms of algebraic attacks.

In [Tez16], the undisturbed bits have been established:

Table 12: DryGASCON128 sbox undisturbed bits

input difference	output difference		input difference	output difference
00001	?1???		10000	?10??
00010	1???1		10001	10??1
00011	???0?		10011	0???0
00100	??110		10100	0?1??
00101	1????		10101	????1
00110	????1		10110	1????
00111	0??1?		10111	????0
01000	??11?		11000	??1??
01011	???1?		11100	??0??
01100	??00?		11110	?1???
01110	?0???		11111	?0???
01111	?1?0?			

Using the tool "SET" [PBJ$^+$], we computed the following results:

- S-box is balanced.
- Nonlinearity is 8.
- Corelation immunity is 0.
- Absolute indicator is 32.
- Sum of square indicator is 8192.
- Algebraic degree is 2.
- Algebraic immunity is 2.
- Transparency order is 4.258.
- Propagation characteristic is 0.
- Strict Avalanche Criterion is not satisfied.
- Number of fixed points is 0.
- Number of opposite fixed points is 0.
- Composite algebraic immunity is 2.
- Delta uniformity is 8.
- Confusion coefficient variance is 0.501560.

Remark 11. The tool evaluate only the sbox, the metrics related to differential cryptanalysis and DPA have been omited since they also depends on other aspects of the cipher.

10.6 Linear cryptanalysis

10.6.1 Comparison between $GASCON_{C5R11}$ and ASCON

Using the "lineartrails" tool [DEM15], we have verified the equivalence between the round function of ASCON and $GASCON_{C5R11}$ function. In both cases, the best unconstrained linear characteristic on 3 rounds has 13 active S-boxes and a bias of 2^{-15}. Tables 13 and 14 show the unconstrained linear characteristic in both cases.

Table 13: ASCON unconstrained linear characteristic for 3 rounds

Round	State				
0	`...........4....`	`.4.8.2...1...48.`	`.4.8.2...1...481`	`...........44...`	`............4...`
1	`................`	`................`	`................`	`.....2........81`	`.....2........81`
2	`................`	`................`	`................`	`................`	`...............1`
3	`................`	`................`	`fc1.c53d1c96ecd5`	`a423953bbde612d6`	`................`

Table 14: $GASCON_{C5R11}$ unconstrained linear characteristic for 3 rounds

Round	State				
0	`1......8.21.1.21`	`................`	`................`	`.............18.`	`1......8.21.11a.`
1	`................`	`................`	`................`	`.......8..1....1`	`.......8..1....1`
2	`................`	`................`	`................`	`................`	`...............1`
3	`................`	`e37c4f1b6e8d53e6`	`e.8629e8e4b766af`	`................`	`................`

10.6.2 Full F function

The $Mix128$ function and the x secret state are unique features of the DrySponge construction. The $Mix128$ does 13 calls to $MixSX32$ interleaved with 12 calls to $GASCON_{C5R11}$. We choose to focus on the last call to $MixSX32$ and the following call to the G function.

The $MixSX32$ function can be seen as a secret transformation mapping 2 bit of input to 32 bit. Depending on x, this transformation may be linear or non linear. We make the assumption that x is such that the transformation is linear. Note that this is unlikely if x is taken at random: there are less than 2^{96} such values distributed over a total of 2^{128} values. It follows that this assumption has a probability of 2^{-32} to hold.

Even with this assumption, the linear cryptanalysis is constrained on the input side. Usable characteristics are limited to the lower half of each 64 bit word of c.

During the rounds the *Accumulate* function xor two bits of c with one bit of r, it follows that all the bits which contribute to the same bit in r are a set linked by a contraint. A linear characteristic shall include the entire set.

Using the "lineartrails" tool [DEM15], we found the following linear characteristic.

Table 15: $r = 3$: 46 active S-boxes, bias 2^{-73}.

Round	State				
0	`.286.514........`	`1......2........`	`1.....82........`	`88.6192.........`	`4a8.3428........`
1	`c42.2.8...42.3..`	`81.4..2.4.....1.`	`4.....1.8......4`	`42.42.....4....4`	`.3....3......3..`
2	`8..........2....`	`8.....2.........`	`..2...1.8.....2.`	`..2.....8..2....`	`4.....3.........`
3	`d9286e927421566c`	`12c662bb1c7bd2a3`	`2.8296eae4b6218b`	`6ace2837da33.b2e`	`................`

Table 16: $r = 4$: 87 active S-boxes, bias 2^{-140}.

Round	State				
0	`.4.45888........`	`5..2.2.1........`	`1292.321........`	`.9.88...........`	`8c2cc.84........`
1	`cb213a4c8.2.1324`	`8..4.82.4.8...1.`	`c9..486a82..1.19`	`...421.8..4.1...`	`c8.85.4d8.4d...c`
2	`812.2..4..2..22.`	`832....4..21..27`	`812.....c..8.227`	`.2212......5...1`	`8224....c..1.2..`
3	`8..........1....`	`8......4...4....`	`8.......8..5....`	`...........1....`	`........8.2.....`
4	`a3145c645.da.6ba`	`21146a9fd7b588e3`	`eafc8629bd18b766`	`85d5e9382c3f68ce`	`................`

Table 17: $r = 5$: 108 active S-boxes, bias 2^{-173}.

Round	State				
0	.8288.1.........	6..2.14.........	4.4....1........	.a1....2........	.131441a........
1	f763c4.54e.7.45c	5c5262caa4.41145	3493a4.5f34d4125	8752f2c52c4d466b	2c538.44c5445..7
2	8816a..4414.421.	48..a.85c.614.25	c81...85c.614225	6.1422..82214224	e.1622.19361.234
3	88..a........2..	8...8........2.4	88..8...8......4	.8..a........2..	....
4	8...............	....	........8.......	....	.8..............
5	eb5ad6bd35ef7ad6	c2ca4d1289716d.2	7f25bb358e.8629e	1.babd27c587ed19	....

Remark 12. As x is secret, the attacker shall search the 2^{96} possibilities for which the $MixSX32$ is linear with respect to the user input.

In conclusion 4 rounds of $GASCON_{C5R11}$ is sufficient to resist linear cryptanalysis. The analysis revealed that the $Mix128$ and the *Accumulate* functions increase the difficulty to find good characteristics. We conclude that linear cryptanalysis is not practical on $DryGASCON128$.

10.7 Differential cryptanalysis

10.7.1 Probability 1 differentials

Using knownledge of the undisturbed bits of the sbox, truncated differentials with probability 1 can be constructed. In [Tez16] several such differentials are described on ASCON. In particular it is shown that controlling the input of a single sbox lead to a differential up to 'round 3.5', meaning round 3 after the sbox layer. On $GASCON_{C5R11}$ the best differential with probability 1 reach round 3 before the sbox layer, table 18 show one case (the dots marks the bits with unknown difference). We verified by computer search that controlling 2 sboxes does not allow longer differentials.

Table 18: 3 round differential with probablility 1

Step	State
	0000000100000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
Input	0000000000000000000000000000000000000000000000000000000000000000
	0000000100000000000000000000000000000000000000000000000000000000
	0000000100000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000.00000000000000000000000000000000000000000000000000000000
S1	0000000.00000000000000000000000000000000000000000000000000000000
	0000000.00000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000.000000000000.0000000000000000000.00000000000000000000000
L1	0000.00.000000000000000000000000000000.0000000000000000000000000
	00.0000.000000000000000000000000000000000000000000000000000000.0
	0000000000000000000000000000000000000000000000000000000000000000
	00.0.00.000000000000.00000000000000000.0.000000000000000000000.0
	00.0.00.000000000000.00000000000000000.0.000000000000000000000.0
S2	00.0.00.000000000000.00000000000000000.0.000000000000000000000.0
	00.0.00.000000000000.00000000000000000.0.000000000000000000000.0
	00.0000.000000000000.0000000000000000000.000000000000000000000.0
	00.0.0..000000000000...00.000.0.000000.0.0.00000.0000000.0.00..0
	...0.00..0.0000.0.00.00000000000000000.0..0.00.0000000.0.00000000.0
L2	0..0.0...00000000.00.000000000..0.0.0..0.0000000000.0000000.00.0
	.0.0.00.0000000.0000.0.000000...0.0.00.0.00.0000000000000.0.00.0
	.0.00.0.000000.0000..000000.0000000.0000.0.00000.000.000000000.0
	...0.....0.000..0.0....00.0.0...0.0.0..0.0..0000.00...00....0..0
	...0.....0.000..0.0....00.0.0...0.0.0..0.0..0000.00...00....0..0
S3	...0.....0.000..0.0..0.0000.0...0.0.0..0.0..0000.00...000.0.00.0
	...0.....0.000..0.0....00.0.0...0.0.0..0.0..0000.00...00....0..0
	...0.....0.000..0.0....00.0.0...0.0.0..0.0..0000.00...00....0..0
	.........0....................................0...0............0
	.........0......0.......0...........0.......00...0..............
L3	.........0...0....0....0....................0..0.......0........
	.......................0..........0.0..0.0.......0..............
	............................0...............0.0.................

10.7.2 Differentials on F

We used the open-source tool CryptoSMT [Ste] and found the minimum weight for a differential on 3 rounds to be 39. The result is shown in table 19. In this search we omited the Mix function to keep the search space in a reasonable size. The input difference is constrained to the 32 least significant bits of each 64 bit words s0 to s4 which represent the capacity. The rate output is represented by the words r0 and r1.

Table 19: 3 round differential of weight 39

Rounds	s0	s1	s2	s3	s4	r0	r1	w
0	0x0000000000000000	0x0000000000800000	0x0000000000800000	0x0000000000000000	0x0000000000000000	0x0000000000000000	0x0000000000000000	-1
1	0x0000200000800200	0x0000000000000000	0x0000000000000000	0x0000000000000000	0x0000000000000000	0x0000200000800200	0x0000000000000000	-2
2	0x0000000008800010	0x0500240000c08210	0x0000000000000000	0x0000000000000000	0x0000000000000000	0x0000200008000210	0x0500240000c08210	-5
3	0x9100002000820300	0x100124800cc21108	0x05c0618805d8b652	0x0d2c456188838624	0x07c82c7100018618	0x94d8967205ae4471	0x9d8286a409c2f290	-

We conclude that $GASCON_{C5R11}$ is sufficienty robust against differential cryptanalysis.

11 Specification of DryGASCON256

It process blocks of 128 bits and taget a security level of 256 bits.

11.1 Supported key sizes

Table 20: DryGASCON256 key sizes

Profile	size
Small	256
Fast	384
Full	704

11.2 Security claims for AEAD mode

Table 21: DryGASCON256 AEAD mode security claims

Requirement	security level in bits
Confidentiality of plaintext	256
Integrity of plaintext	256
Integrity of associated data	256
Integrity of static data	256
Integrity of `NONCE`	256

Conditions:

- `NONCE` is not reused for a given key.
- no more than 2^{128} bytes are encrypted with a given key (including `NONCE` bytes).

11.3 Security claims for hash mode

Table 6 present the security claims in hash mode:

Table 22: DryGASCON256 Hash mode security claims

Requirement	security level in bits
Integrity of data	256

Conditions:

- Message length is less than 2^{128} bytes.

11.4 Parameters

Table 23: DryGASCON256 parameters

Name	value
$i.width$	128
$r.width$	128
$c.width$	576
$x.width$	128
d	1
t	1
$AccumulateFactor$	4
KS	$KSneq32$
Mix	$Mix128$
$CoreRound$	$GASCON_{C9}$
$DRounds$	12
$Rounds$	8

11.4.1 SBox

Table 24 gives the values in hexadecimal. The number in front of each row indicate the value of the most significant bit of the input. The column number gives the value of the four least significant input bits. For example S(0x01) is 0x93 and S(0x10) is 0x2c.

Table 24: DryGASCON256 sbox

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	10	93	11f	9d	1f	9c	113	91	2a	a9	127	a5	27	a4	129	ab
1	2c	af	123	a1	23	a0	12f	ad	1a	99	117	95	17	94	119	9b
2	f8	7b	1f7	75	f7	74	1fb	79	ca	49	1c7	45	c7	44	1c9	4b
3	cc	4f	1c3	41	c3	40	1cf	4d	f2	71	1ff	7d	ff	7c	1f1	73
4	e0	63	1ef	6d	ef	6c	1e3	61	da	59	1d7	55	d7	54	1d9	5b
5	dc	5f	1d3	51	d3	50	1df	5d	ea	69	1e7	65	e7	64	1e9	6b
6	38	bb	137	b5	37	b4	13b	b9	0a	89	107	85	07	84	109	8b
7	0c	8f	103	81	03	80	10f	8d	32	b1	13f	bd	3f	bc	131	b3
8	1b1	1b2	be	1bc	1be	1bd	b2	1b0	18b	188	86	184	186	185	88	18a
9	18d	18e	82	180	182	181	8e	18c	1bb	1b8	b6	1b4	1b6	1b5	b8	1ba
a	179	17a	76	174	176	175	7a	178	14b	148	46	144	146	145	48	14a
b	14d	14e	42	140	142	141	4e	14c	173	170	7e	17c	17e	17d	70	172
c	161	162	6e	16c	16e	16d	62	160	15b	158	56	154	156	155	58	15a
d	15d	15e	52	150	152	151	5e	15c	16b	168	66	164	166	165	68	16a
e	199	19a	96	194	196	195	9a	198	1ab	1a8	a6	1a4	1a6	1a5	a8	1aa
f	1ad	1ae	a2	1a0	1a2	1a1	ae	1ac	193	190	9e	19c	19e	19d	90	192
10	1d2	1d1	1dc	de	1dd	1de	1d0	d2	1e8	1eb	1e4	e6	1e5	1e6	1ea	e8
11	1ee	1ed	1e0	e2	1e1	1e2	1ec	ee	1d8	1db	1d4	d6	1d5	1d6	1da	d8
12	13a	139	134	36	135	136	138	3a	108	10b	104	06	105	106	10a	08
13	10e	10d	100	02	101	102	10c	0e	130	133	13c	3e	13d	13e	132	30
14	122	121	12c	2e	12d	12e	120	22	118	11b	114	16	115	116	11a	18
15	11e	11d	110	12	111	112	11c	1e	128	12b	124	26	125	126	12a	28
16	1fa	1f9	1f4	f6	1f5	1f6	1f8	fa	1c8	1cb	1c4	c6	1c5	1c6	1ca	c8
17	1ce	1cd	1c0	c2	1c1	1c2	1cc	ce	1f0	1f3	1fc	fe	1fd	1fe	1f2	f0
18	33	b0	3d	1bf	3c	bf	31	1b3	09	8a	05	187	04	87	0b	189
19	0f	8c	01	183	00	83	0d	18f	39	ba	35	1b7	34	b7	3b	1b9
1a	fb	78	f5	177	f4	77	f9	17b	c9	4a	c5	147	c4	47	cb	149
1b	cf	4c	c1	143	c0	43	cd	14f	f1	72	fd	17f	fc	7f	f3	171
1c	e3	60	ed	16f	ec	6f	e1	163	d9	5a	d5	157	d4	57	db	159
1d	df	5c	d1	153	d0	53	dd	15f	e9	6a	e5	167	e4	67	eb	169
1e	1b	98	15	197	14	97	19	19b	29	aa	25	1a7	24	a7	2b	1a9
1f	2f	ac	21	1a3	20	a3	2d	1af	11	92	1d	19f	1c	9f	13	191

11.5 Security analysis

11.5.1 SBox

The differential distribution and linear approximation have been computed. The associated 'heat' maps are given in appendix 14.1.

The differential branch number is 3 and the linear branch number is 3. The associated tables are too large to fit in a page so instead 'heat' maps are given in appendix 14.1 in figure 10 and 11.

Using the tool "SET" [PBJ+], we computed the following results:

- S-box is balanced.
- Nonlinearity is 128.
- Corelation immunity is 0.
- Absolute indicator is 512.
- Sum of square indicator is 33554432.
- Algebraic degree is 2.
- Algebraic immunity is 2.
- Transparency order is 7.667.
- Propagation characteristic is 0.
- Strict Avalanche Criterion is not satisfied.
- Number of fixed points is 0.
- Number of opposite fixed points is 2.
- Composite algebraic immunity is 2.
- Delta uniformity is 128.
- Confusion coefficient variance is 1.275545.

11.6 Linear cryptanalysis

Using the same methodology as in section 10.6.2, we found the following linear characteristics.

Table 25: $r = 3$: 59 active S-boxes, bias 2^{-60}.

Round	State								
0	....	....	....	....	.a9ee62f........	.a9ee62f........	....	....	....
1	....	....	....	11c.24.9.4235a43	....	....	....	....	....
2	....	....	....	218211ad........	2b9291ada41..191	....	....	....	....
3	....	....	....	....	....	95d.c689cb3d4d26	81c7a5fd9b7.3bc6	....	....

Table 26: $r = 5$: 112 active S-boxes, bias 2^{-274}.

Round	State								
0	4..24.11........	...11.1.........	...4.86.........	.....8..........	8.2.14.5........	8...1..1........	2..442c8........	..88.2.8........	48...841........
1	aed986549b.7f47a	ec88c637f46bc.45	7249417.e.82121f	56c3732.c226.215	a54d26571.13644c	b4cf221124183.4c	99b37776a.d77518	a383d.36fb.78516	ae51d22471c8.47c
2	.c8.a.44..1964.2	3.22a...a4.84..2	2c.2a..422.d6.42	2d22a..4a6.d4.42	212....484.42..2	21a.8.4484.42..2	...28..4.418....	1182..4...1.2.4.	3.8.8.44.418.4..
3	....8......8....	....8......8....	.1.......4..2...	.1.......4..2...	....	....	....	....8......8....	.1.......4..2...
4	....	.........4......	....	....	....	....	....	....	.........4......
5	ca5364de25.9b26b	....	....	....	....	....	....	....	....

Table 27: $r = 6$: 151 active S-boxes, bias 2^{-398}.

Round	State								
0	14...53.........	24...11.........	8.28..2.........	8.28.4..........	48...42.........	48....2.........	...88...........	....	3...............
1	.1.8.4.22.1.....	....	12d8.....43.....	.21.84..2..8....	....	....	....	.1.8.4.2.1....4.	2.488.12.4811...
2	....84.4.3....c.	18464.49653819a4	9.27e4222.88c124	38c495282d2.96b4	8.86a7cb7b9cc574	3.a394.8243ca464	2.e2...12db...49	2...8..2.32.8.44	b88123ca.52c78f9
3	5.d5e23.23.84c..	998ee295171ffc18	a.5f7.74.2.6945c	a4d77.b..786dc54	849a9881558cc2.8	a49ad8f15515ce.4	8148181.518b36.c	3999b87.73922e.c	c95.e.e5211.6c1.
4	....c.44..1.c.1.	2..a8.84.28.c.1.	...a8..4....c...	...2c..4..8.c...	...84.....8.....	...84.....8.....	....c.8...1.....	....c.8..21.....	2...4.c4.21.....
5	....8...........	....8...........	....4...........	....4...........	....	....	....	....8...........	....4...........
6	4d64a13637b294d9	54f99ba3278df1be	....	....	....	....	....	....	....

11.7 Differential cryptanalysis

11.7.1 Probability 1 differentials

Using computer search, we compute differentials with probablity 1 for every input impacting a single sbox. On $GASCON_{C9R12}$ the best differential with probability 1 reach round 3 after the sbox layer, table 28 show one case (the dots marks the bits with unknown difference). We verified by computer search that controlling 2 sboxes does not allow longer differentials, the best result is actually lower, reaching round 3 before the sbox layer.

Table 28: 3.5 round differential with probablility 1

Step	State
Input	0000000000000000000000000000000000000000000000000000000000000000
	0100000000000000000000000000000000000000000000000000000000000000
	0100000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
S1	0.00000000000000000000000000000000000000000000000000000000000000
	0.00000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0.00000000000000000000000000000000000000000000000000000000000000
L1	0.00000000000000000.00000000000000000000000000000000000.00000000
	0.000000000000.0000000000000000000.00000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0.000000.0000000000000000000000000000000000.00000000000000000000
S2	0.000000.00000.0000.00000000000000.00000000.00000000000.00000000
	0.000000.00000.0000.00000000000000.00000000.00000000000.00000000
	0.000000000000.0000000000000000000.00000000000000000000000000000
	0.000000000000.0000000000000000000.00000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0.000000.0000000000000000000000000000000000.00000000000000000000
	0.000000.0000000000.00000000000000000000000.00000000000.00000000
	0.000000.00000.0000.00000000000000.00000000.00000000000.00000000
L2	...00.00.00000.0000.00000..0000000.0.0000.0.00000000.00.00000..0
	..00.000.0000..0000.0.000.0.000000.0.0000.0.000.0000.00..0000000
	0..00000000.00.000000000000000.0.0.0000000000.00000000000000000.
	0.0000000.0000.00000000000.0.00000.00.000000000000000000.0000.00
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0.00.000.00.00000.00000000000000000000.0000.0..00000000000000000
	0.00000..0.000000.0.00000000.000.00.0000000.00.00000.00.0000.000
	0..00000.0000...000.0..000.0000000.000000.0.000000.0000..0000..0
S3	...0..0..0..0...0.0.0..00....0.0.0...0000.0.0...00.0.00..000....
	...0..00..0.0...000.0..00....0.0.0.0..000.0.0.0.00.0.00..0000...
	...0.000..0.0..0000.0.000....0.0.0.0..000.0.0.0.0000.00..0000.0.
	...0.000..0.0..0000.0.000....0.0.0.0..000.0.0.0.0000.00..0000.0.
	0.00.000..0.00.00.00000000.0.00000.00..0000.0..000000000.0000.00
	0.00.00.....00.00.0.000000.0.000.0..0..0000.0..00000.00..000..00
	0..0.00..0..0...0.0.0..000.0.000.0..00.00.0.0..000.0.00..000...0
	...0..0..0..0...0.0.0..00..0.000.0...0.00.0.0..000.0.00..000...0
	...0..0..0.00...0.0.0..00....000.0...0000.0.00..00.0.00..000...0
L3	........................0...............0..........0.....0......
	..........0.0...0......0.....0.........00..........0.0...0......
	...0...0....0....0............................0.0..0..0.......0.
	..........0......00..............0.0...0......0.......0.....0.0.
	0....0.0....0..0..00..0.00.0..0..0.0....0...0..00..000.....00.00
	0.00........0...0.0..0..00..........0...00........0............0
	0......................0...0........0...0.0..............0......
	......0.....0.....0........0.0.....................0.....00.....
	......0.........0.0....0......0..0......0....0.....0......0.....
S4	....0............
	....
	....
	....
	....
	....
	....
	....
	....0............

11.7.2 Differentials on *F*

Using the same methodology as in section 10.7.2, we found the following differentials.

Table 29: 2 round differential of weight 8

Rounds	State					r0	r1	w
0	0x0000000000000000	0x0000000000000000	0x0000000000000000	0x0000000000000000				
	0x0000000000000000	0x0000000000000004	0x0000000000000004	0x0000000000000000	0x0000000000000000	0x0000000000000000	0x0000000000000000	-1
1	0x0000000000000000	0x0000000000000000	0x0000000000000000	0x0000000000000000				
	0x4000000000004004	0x0000000000000000	0x0000000000000000	0x0000000000000000	0x0000000000000000	0x0000000000000000	0x4000000000004004	-2
2	0x0000000000000000	0x0000000000000000	0x0000000000000000	0x0000000000000000				
	0x000000000c000004	0x000600000020c006	0x0000000000000000	0x0000000000000000	0x0000000000000000	0x000600000020c006	0x400000000c004000	-

12 Advantages and limitations

12.1 Advantages

12.1.1 Efficient protection against physical attacks

As they do not require implementation level countermeasure, DryGASCON128 and DryGASCON256 are expected to shine in benchmarks of secure hardware implementation, i.e. implementations protected against both side channels and fault attacks. This should be the case for secure software implementations on 32bit and 64bit platforms.

12.1.2 No randomness needed

DryGASCON does not need randomness to achieve security against side channel attacks. In contrast, state of the art countermeasures such as Threshold implementation [NRR06], Domain Oriented Masking [GMK16], Unified Masking Approach [GM17], all require randomness. This is not desirable for many reasons, in particular the generation of random numbers consumes power and area.

12.1.3 Achieving DPA resistance without patents

The general principles to achieve DPA resistance at implementation level have been patented. In particular the patents cover signal path equalization, noise generation, time jitter, randomization of execution order, masking (non exhaustive list). DryGASCON achieves DPA resistance at algorithmic level and therefore does not need those patented countermeasures to be useful in real world applications.

12.1.4 Ease of implementation and security verification

The implementation of security countermeasures and their verification dominates the effort to develop a hardware crypto accelerator. From past experience in the smart card industry, we estimate the ratio at 1 to 9: what takes 1 month to implement without security in mind takes at least 9 month when the goal is to resist side channel and fault attacks. The budget is impacted even more as additional skills are required as well as silicon respins. All those issues are sidelined by DryGASCON as the countermeasures are built in the algorithm.

12.2 Limitations

12.2.1 Overhead compared to unprotected implementations

For applications not exposed to physical attacks, it is clear that this submission is not the smallest nor the most efficient one. We know for sure that it is less efficient than ASCON given that the DryGASCON128 add operations on top of what is done in ASCON.

We think that in practice most low cost devices are exposed to physical attacks. If something is worth encrypting then the key encrypting it worth protection and as a result the real cost need to be measured on implementations protected against both side channels and fault attacks. Currently many consider only side channel attacks to be relevant in most cases. We share this point of view however it is clear that if side channel attacks are not longer effective fault attacks will be used.

Finally we note that DryGASCON128 could easily support a high efficiency mode based on the $GASCON$ round function. Hardware manufacturer could then offer accelerators that allow to choose between security and efficiency on the fly. That would allow protocols to choose the secure mode for establishing a session key and use the efficient mode for bulk transfers.

12.2.2 Software implementation on 8 and 16 bit platforms

In this case the algorithmic level countermeasures do not protect against side channel attacks. The reason is that all transfers and operations on those machines are subject to template attacks.

13 Implementations and Benchmarks

This section show implementation figures for DryGASCON on various platforms. We expect that it is possible to optimize the results much more. We also expect most performance optimizations to be neutral from side channel and fault resistance point of view.

13.1 Hardware implementations

13.1.1 Xilinx Zynq-7000 FPGA

DryGASCON128 F and G functions have been implemented in verilog HDL and synthesized using Vivado tool. (File drygascon128.v in the implementation package) The architecture is among the simplest that can be done:

- $Mix128$ process 10 input bits per cycle
- G function executes 1 call to $GASCON_{C5R11}$ per cycle
- No pipelining

The only compromise on simplicity is the interface: rather than using fully parallel I/Os we implemented a 32 bit oriented interface. We made this choice in order to provide an accelerator block which is almost ready for integration in a microcontroller system.

Table 30: Performance of DryGASCON128 on FPGA

	7 rounds	11 rounds
G execution time	8 cycles	not used
F execution time	20 cycles	24 cycles

Taking into account the cycles needed for I/O operations, the digest for a message up to 16 bytes takes 65 cycles.

The implementation has been synthesized succesfully at 150MHz. As the timing report indicates that the net delays dominates over the logic delays, no optimization attempt have been made.

The utilization report indicates that 625 flipflops have been used. The bulk of them are accounted for by the state (576 bits) and a 32 bit register which has been implemented for the data output.

Listing 1: Extract of utlization report

```
+----------+------+---------------------+
| Ref Name | Used | Functional Category |
+----------+------+---------------------+
| LUT6     |  922 |                 LUT |
| LUT5     |  678 |                 LUT |
| FDRE     |  624 |        Flop & Latch |
| LUT3     |  267 |                 LUT |
| MUXF7    |  150 |               MuxFx |
| LUT4     |  136 |                 LUT |
| LUT2     |   53 |                 LUT |
| MUXF8    |   10 |               MuxFx |
| CARRY4   |    3 |          CarryLogic |
| FDSE     |    1 |        Flop & Latch |
+----------+------+---------------------+
```

The full utilization and timing reports are in appendix 14.2.

13.1.2 Lattice ICE40 HX8K FPGA

The basic implementation used on Xilinx FPGA reach a much lower frequency on the ICE40 HX8K. The open source tools icestorm [WL] reach only 21MHz. To improve the result, two enhancement have been done:

- ACC_PIPE: Compute the accumulate function one cycle later than the core round. This shorten the critical path at the expense of 1 additional cycle per invocation of F or G.
- MIX_SHIFT_REG: Use a shift register to set the input of $MixSX32$ (instead of muxes in the basic implementation).

Using icestorm, the implementation 'drygascon128_ACC_PIPE_MIX_SHIFT_REG.v' reaches 41MHz. Lattice's LSE tool reaches 103MHz.

The implementation is small enough to be fitted in the HX8K along within 'Murax' , an open source SOC based on the VexRiscv implementation of the RV32IMC CPU [Pap]. The accelerator is mapped as an APB3 peripheral. The embedded software allows to specify the inputs of F and to get the results via UART. The resulting bitfile can be downloaded on the 'ICE40 hx8k breakout' board to be used as a low cost platform for physical security evaluations.

13.2 Software implementations

13.2.1 ARM Cortex M0

DryGASCON128 F and G functions in ARM cortex M0 assembly with the objective to balance code size and performances. The code is provided in appendix

Table 31: Memory footprint of DryGASCON128 F and G on ARM Cortex M0

Code size	980 bytes
Stack size	52 bytes

Table 32: Performance of DryGASCON128 F and G on ARM Cortex M0

	7 rounds	11 rounds
G execution time	1.7 KCycles	not used
F execution time	4.9 KCycles	5.8 KCycles

In AEAD mode, the total number of clock cycles for a 80 bytes message and no associated data is 31212 (throughput is 390 cycles per bytes). We obtained this number by executing C code compiled using GCC 6.3.1 with -Os -fPIC flags on an STM32L011K4 device.

This compares advantageously to the 31 KCycles for 64 bytes of message and 16 bytes of associated data reported in [AFM18] for a masked implementation of ASCON. From the numbers the results seems similar but they are not. It is clear that DryGascon128 performs significantly better once the following is taken into account:

- Only the initialization and the finalization have been protected against DPA.
- The target device was a Cortex M3 (the instruction set is significantly more efficient).
- No protection against fault attacks have been implemented.

The authors justify their choice of protecting against DPA only the initialization and the finalization: they argue that a DPA on an arbitrary block allows only to recover n bits of state where n is the block size. We note that the possibility of using the new knowledge to mount further DPA attack has not been discussed.

13.2.2 RISC-V RV32E

The benchmark has been done on the GAP8 SOC from Greenwaves [gre]. The SOC contains 9 RISC-V RV32IMC cores 'RI5CY' from the 'PULP' platform [pul]. Even though the target contained 32 registers we limited our code to the 16 registers available on RV32E. The only departure from a pure RV32E implementation is the use of the p.ror instruction. This instruction is not a standard RV32 instruction, it was considered as part of the 'bit manipulation' extension before the working group disolved.

Table 33: Memory footprint of DryGASCON128 F and G on RISC-V RV32E

Code size	1372 bytes
Stack size	60 bytes

Table 34: Performance of DryGASCON128 F and G on RISC-V RV32E

	7 rounds	11 rounds
G execution time	1.3 KCycles	not used
F execution time	4.0 KCycles	4.7 KCycles

The following listings show the details of this benchmark. This shows that the performances could be improved further by carefully reordering the instructions to avoid stalls and by avoiding memory accesses to the external memory.

CYCLES Number of cycles the core was active (not sleeping)

INSTR Number of instructions executed

LD_STALL Number of load data hazards

JMP_STALL Number of jump register data hazards

IMISS Number of cycles waiting for instruction fetches, i.e. number of instructions wasted due to non-ideal caching

LD Number of data memory loads executed. Misaligned accesses are counted twice

ST Number of data memory stores executed. Misaligned accesses are counted twice

JUMP Number of unconditional jumps (j, jal, jr, jalr)

BRANCH Number of both taken and not taken branches

TAKEN_BRANCH Number of taken branches

RVC Number of compressed instructions executed

LD_EXT Number of memory loads to EXT executed. Misaligned accesses are counted twice. Every non-L1 access is considered external

ST_EXT Number of memory stores to EXT executed. Misaligned accesses are counted twice. Every non-L1 access is considered external

LD_EXT_CYC Number of cycles used for memory loads to EXT. Every non-L1 access is considered external

ST_EXT_CYC Number of cycles used for memory stores to EXT. Every non-L1 access is considered external

TCDM_CONT Number of cycles wasted due to L1/log-interconnect contention

Listing 2: Performance counters for 7 rounds *G* function

```
      CYCLES:   1317
       INSTR:   1114
    LD_STALL:     70
   JMP_STALL:      1
       IMISS:      0
          LD:     92
          ST:     85
        JUMP:      7
      BRANCH:      7
TAKEN_BRANCH:      1
         RVC:    513
      LD_EXT:      7
      ST_EXT:      1
  LD_EXT_CYC:     87
  ST_EXT_CYC:      4
   TCDM_CONT:      0
```

Listing 3: Performance counters for 7 rounds *F* function

```
      CYCLES:   3988
       INSTR:   3209
    LD_STALL:    159
   JMP_STALL:      1
       IMISS:    173
```

```
           LD:    208
           ST:    114
         JUMP:     21
       BRANCH:     21
 TAKEN_BRANCH:      2
          RVC:   1259
       LD_EXT:      7
       ST_EXT:      1
   LD_EXT_CYC:     86
   ST_EXT_CYC:      4
    TCDM_CONT:      0
```

Listing 4: Performance counters for 11 rounds F function

```
       CYCLES:   4689
        INSTR:   3817
     LD_STALL:    199
    JMP_STALL:      1
        IMISS:    175
           LD:    252
           ST:    150
         JUMP:     25
       BRANCH:     25
 TAKEN_BRANCH:      2
          RVC:   1535
       LD_EXT:     11
       ST_EXT:      1
   LD_EXT_CYC:    137
   ST_EXT_CYC:      4
    TCDM_CONT:      0
```

References

[AFM18] Alexandre Adomnicai, Jacques J.A. Fournier, and Laurent Masson. Masking the lightweight authenticated ciphers acorn and ascon in software. Cryptology ePrint Archive, Report 2018/708, 2018. https://eprint.iacr.org/2018/708.

[BDPA11] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the sponge: single-pass authenticated encryption and other applications. Cryptology ePrint Archive, Report 2011/499, 2011. https://eprint.iacr.org/2011/499.

[DEK$^+$18] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Stefan Mangard, Florian Mendel, and Robert Primas. Sifa: Exploiting ineffective fault inductions on symmetric cryptography. Cryptology ePrint Archive, Report 2018/071, 2018. https://eprint.iacr.org/2018/071.

[DEM15] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. Heuristic tool for linear cryptanalysis with applications to caesar candidates. Cryptology ePrint Archive, Report 2015/1200, 2015. https://eprint.iacr.org/2015/1200.

[DEMS15] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Cryptanalysis of ascon. Cryptology ePrint Archive, Report 2015/030, 2015. https://eprint.iacr.org/2015/030.

[DEMS16] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2. Submission to the CAESAR competition: http://competitions.cr.yp.to/round3/asconv12.pdf, 2016.

[DMMP18] Christoph Dobraunig, Stefan Mangard, Florian Mendel, and Robert Primas. Fault attacks on nonce-based authenticated encryption: Application to keyak and ketje. Cryptology ePrint Archive, Report 2018/852, 2018. https://eprint.iacr.org/2018/852.

[FJLT13] Thomas Fuhr, Éliane Jaulmes, Victor Lomné, and Adrian Thillard. Fault attacks on aes with faulty ciphertexts only. *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 108–118, 2013.

[GM17] Hannes Gross and Stefan Mangard. Reconciling d+1 masking in hardware and software. Cryptology ePrint Archive, Report 2017/103, 2017. https://eprint.iacr.org/2017/103.

[GMK16] Hannes Gross, Stefan Mangard, and Thomas Korak. Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. Cryptology ePrint Archive, Report 2016/486, 2016. https://eprint.iacr.org/2016/486.

[gre] Greenwaves Technology Website. https://greenwaves-technologies.com/.

[LDLL13] Yanis Linge, Cecile Dumas, and Sophie Lambert-Lacroix. Using the joint distributions of a cryptographic function in side channel analysis. Cryptology ePrint Archive, Report 2013/859, 2013. https://eprint.iacr.org/2013/859.

[NRR06] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. 2006.

[Pap] Charles Papon. A fpga friendly 32 bit risc-v cpu implementation. https://github.com/SpinalHDL/VexRiscv.

[PBJ$^+$] Stjepan Picek, Lejla Batina, Domagoj Jakobović, Barış Ege, and Marin Golub. S-box, set, match: A toolbox for s-box analysis. https://hal.inria.fr/hal-01400936.

[pul] PULP platform. http://pulp-platform.org.

[Ste] Stefan Kölbl. CryptoSMT: An easy to use tool for cryptanalysis of symmetric primitives. https://github.com/kste/cryptosmt.

[Tez16] Cihangir Tezcan. Truncated, impossible, and improbable differential analysis of ascon. Cryptology ePrint Archive, Report 2016/490, 2016. https://eprint.iacr.org/2016/490.

[WL] Clifford Wolf and Mathias Lasser. Project icestorm. http://www.clifford.at/icestorm/.

14 Appendix

14.1 S-box tables

Table 35: DryGASCON128 sbox differential distribution table

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
0	32	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
1	.	.	.	8	.	.	.	.	.	.	.	8	.	.	.	.	.	.	8	.	.	.	.	.	.	.	8	.	.	.	.	.
2	.	.	.	.	.	.	.	.	.	.	.	.	4	4	4	4	.	.	.	.	.	.	.	.	.	.	.	.	4	4	4	4
3	.	.	.	.	2	2	2	2	.	.	.	.	2	2	2	2	.	.	.	.	2	2	2	2	.	.	.	.	2	2	2	2
4	.	.	.	.	.	.	.	.	.	.	.	.	8	8	8	8	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
5	.	.	.	.	4	.	4	.	.	.	.	.	4	.	4	.	.	.	.	.	4	.	4	.	.	.	.	.	4	.	4	.
6	.	8	8	.	.	.	.	.	.	.	.	.	.	.	.	.	8	.	.	8	.	.	.	.	.	.	.	.	.	.	.	.
7	.	4	4	.	.	.	.	.	.	4	4	.	.	.	.	.	4	.	.	4	.	.	.	.	4	.	.	4	.	.	.	.
8	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	4	.	4	.	4	.	4	.	4	.	4	.	4	.	4
9	.	2	.	2	.	2	.	2	.	2	.	2	.	2	.	2	2	.	2	.	2	.	2	.	2	.	2	.	2	.	2	.
a	.	.	.	.	2	2	2	2	2	2	2	2	.	.	.	.	2	2	2	2	.	.	.	.	.	.	.	.	2	2	2	2
b	.	.	2	2	.	.	2	2	2	2	.	.	2	2	.	.	2	2	.	.	2	2	.	.	.	.	2	2	.	.	2	2
c	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
d	.	2	.	2	.	2	.	2	.	2	.	2	.	2	.	2	.	2	.	2	.	2	.	2	.	2	.	2	.	2	.	2
e	.	.	.	.	4	4	.	.	4	4	.	.	.	.	.	.	4	4	.	.	.	.	.	.	.	.	.	.	4	4	.	.
f	.	.	2	2	.	.	2	2	.	.	2	2	.	.	2	2	.	.	2	2	.	.	2	2	.	.	2	2	.	.	2	2
10	.	.	.	4	.	.	.	4	.	.	.	4	.	.	.	4	.	.	4	.	.	.	4	.	.	.	4	.	.	.	4	.
11	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	8	.	.	.	8	.	.	.	8	.	.	.	8	.	.
12	.	2	2	.	2	.	.	2	.	2	2	.	2	.	.	2	2	.	.	2	.	2	2	.	2	.	.	2	.	2	2	.
13	.	4	4	.	.	.	.	.	.	.	.	.	4	.	.	4	.	.	.	.	.	4	4	.	4	.	.	4	.	.	.	.
14	.	.	.	4	.	4	.	.	.	.	.	4	.	4	.	.	.	4	.	.	.	.	.	4	.	4	.	.	.	.	.	4
15	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	4	4	.	4	.	.	4	.	4	4	.	4	.	.	4
16	.	2	2	.	.	2	2	.	.	2	2	.	.	2	2	.	2	.	.	2	2	.	.	2	2	.	.	2	2	.	.	2
17	.	4	4	.	.	.	.	.	.	.	.	.	.	4	4	.	.	.	.	.	4	.	.	4	4	.	.	4	.	.	.	.
18	.	4	.	4	.	4	.	4	.	.	.	.	.	.	.	.	4	.	4	.	4	.	4	.	.	.	.	.	.	.	.	.
19	.	.	.	.	8	.	8	.	8	.	8	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
1a	.	.	.	.	.	.	.	.	2	2	2	2	2	2	2	2	.	.	.	.	.	.	.	.	2	2	2	2	2	2	2	2
1b	.	.	.	.	2	2	2	2	2	2	2	2	.	.	.	.	.	.	.	.	2	2	2	2	2	2	2	2	.	.	.	.
1c	.	.	.	.	.	.	.	.	4	.	4	.	4	.	4	.	.	.	.	.	.	.	.	.	4	.	4	.	4	.	4	.
1d	.	.	.	.	4	4	4	4	4	4	4	4	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
1e	.	.	4	4	.	.	4	4	.	.	.	.	.	.	.	.	.	.	4	4	.	.	4	4	.	.	.	.	.	.	.	.
1f	.	.	.	.	4	4	.	.	4	4	.	.	.	.	.	.	.	.	.	.	4	4	.	.	4	4	.	.	.	.	.	.

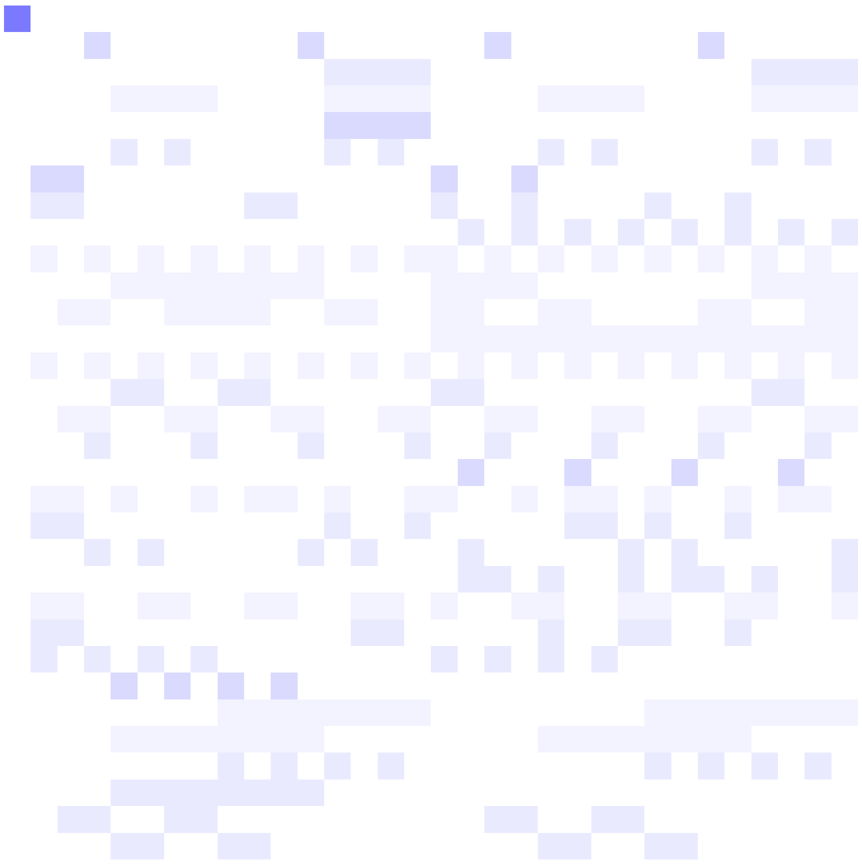
Figure 8: DryGASCON128 sbox differential distribution heat map

Table 36: DryGASCON128 sbox linear approximation table

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
0	16	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
1	.	.	.	4	.	4	-4	-4	.	.	.	-4	-8	4	-4	-4	.	.	4	.	.	-4	.	.	.	.	-4	.	.	4	.	.
2	.	.	.	.	.	.	.	.	.	.	4	-4	.	.	-4	4	.	.	.	8	.	.	.	8	.	.	4	4	.	.	-4	-4
3	.	.	.	-4	.	4	-4	4	.	.	4	.	.	-4	.	.	.	.	4	.	.	-4	.	.	.	.	.	-4	-8	-4	4	-4
4	.	.	.	.	.	-4	.	-4	.	4	4	.	.	.	-4	4	.	.	.	-8	-4	.	4	.	4	.	.	-4	.	.	-4	-4
5	.	.	.	-4	.	.	-4	.	.	4	4	4	.	-4	.	.	.	.	4	.	4	4	-4	.	4	.	-4	4	.	4	-4	4
6	.	.	.	.	-8	4	.	-4	-8	-4	.	4	.	.	.	.	.	.	.	.	-4	.	-4	.	4	.	4	.	.	.	.	.
7	.	.	.	4	.	.	4	.	8	-4	.	.	.	-4	4	4	.	.	4	.	-4	-4	-4	.	4	.	.	.	.	4	.	.
8	.	.	.	.	.	.	.	.	.	4	4	.	-8	-4	4	.	.	.	.	.	.	.	.	.	.	4	4	.	8	-4	4	.
9	.	.	.	4	.	-4	-4	4	.	-4	4	4	.	.	.	-4	-8	.	-4	.	.	-4	.	.	.	4	.	.	.	.	-4	.
a	.	.	.	.	.	.	.	.	.	4	.	4	.	4	.	4	.	.	.	8	.	.	.	-8	.	4	.	-4	.	4	.	-4
b	.	.	.	-4	.	-4	-4	-4	.	-4	.	.	.	.	4	.	8	.	-4	.	.	-4	.	.	.	4	-4	4	.	.	.	-4
c	.	.	.	.	.	-4	.	-4	.	.	.	.	.	-4	.	-4	.	.	.	8	-4	.	4	.	4	-4	-4	-4	.	-4	.	4
d	.	8	.	4	.	.	-4	.	.	.	.	4	.	.	4	.	.	.	4	.	-4	4	4	.	-4	-4	.	4	.	.	.	-4
e	.	.	.	.	8	4	.	-4	.	.	-4	4	.	-4	-4	.	.	.	.	.	-4	.	-4	.	-4	4	.	.	.	-4	-4	.
f	.	8	.	-4	.	.	4	.	.	.	-4	.	.	.	.	-4	.	.	4	.	4	-4	4	.	4	4	4	.	.	.	-4	.
10	.	.	.	4	.	.	.	-4	.	4	4	-4	8	4	4	-4	.	.	4	.	.	.	-4	.	.	4	.	.	.	-4	.	.
11	.	.	-8	.	.	-4	-4	.	.	4	-4	.	.	.	.	.	.	8	.	.	.	-4	-4	.	.	-4	4	.	.	.	.	.
12	.	.	.	4	.	.	.	-4	.	4	.	.	.	-4	.	.	.	.	-4	.	.	.	4	.	.	4	4	4	-8	4	4	4
13	.	.	.	.	.	-4	4	.	.	4	.	4	.	.	-4	-4	.	-8	.	.	.	-4	-4	.	.	-4	.	4	.	.	4	-4
14	.	.	.	4	.	4	.	.	.	.	.	-4	.	-4	.	.	.	.	-4	.	4	.	.	-8	4	-4	.	4	.	-4	-4	-4
15	.	.	.	.	.	.	4	4	.	.	.	.	.	.	-4	-4	.	8	.	.	-4	4	.	.	4	4	-4	4	.	.	4	-4
16	.	.	.	4	-8	-4	.	.	.	.	-4	.	.	-4	-4	4	.	.	4	.	4	.	.	.	-4	4	-4	.	.	-4	.	.
17	.	.	8	.	.	.	4	-4	.	.	4	4	.	.	.	.	.	8	.	.	4	-4	.	.	-4	-4	.	.	.	.	.	.
18	.	-8	.	4	.	.	.	4	.	.	.	4	.	.	.	-4	8	.	4	.	.	.	4	.	.	.	4	.	.	.	-4	.
19	.	.	.	.	.	-4	4	.	.	.	.	.	-8	4	4	.	.	.	.	.	.	4	-4	.	.	.	.	.	-8	-4	-4	.
1a	.	8	.	4	.	.	.	4	.	.	4	.	.	.	-4	.	8	.	-4	.	.	.	-4	.	.	.	.	-4	.	.	.	4
1b	.	.	8	.	.	-4	-4	.	.	.	-4	-4	.	-4	.	-4	.	.	.	.	.	4	-4	.	.	.	4	-4	.	4	.	-4
1c	.	.	.	-4	.	-4	.	.	.	-4	4	-4	.	.	-4	.	.	.	4	.	-4	.	.	-8	-4	.	4	4	.	.	.	4
1d	.	.	8	.	.	.	-4	4	.	4	-4	.	.	4	.	4	.	.	.	.	-4	-4	.	.	4	.	.	4	.	-4	.	4
1e	.	.	.	-4	-8	4	.	.	8	4	.	.	.	.	.	-4	.	.	-4	.	-4	.	.	.	-4	.	.	.	.	.	-4	.
1f	.	.	.	.	.	.	-4	-4	8	-4	.	4	.	4	-4	.	.	.	.	.	4	4	.	.	4	.	4	.	.	-4	4	.

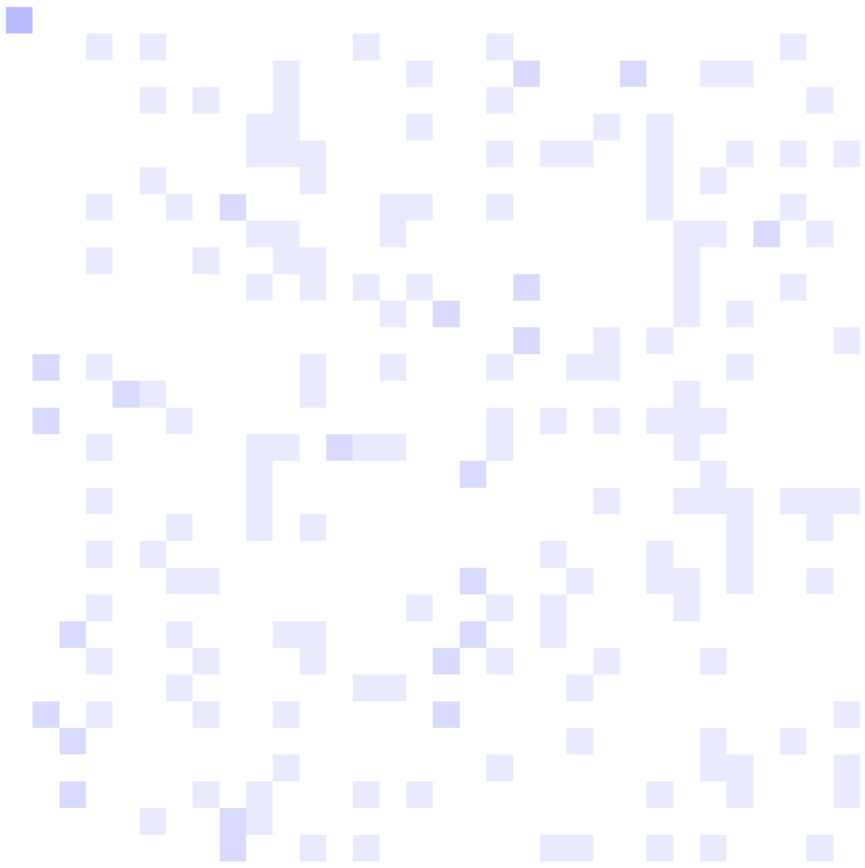
Figure 9: DryGASCON128 sbox linear approximation heat map

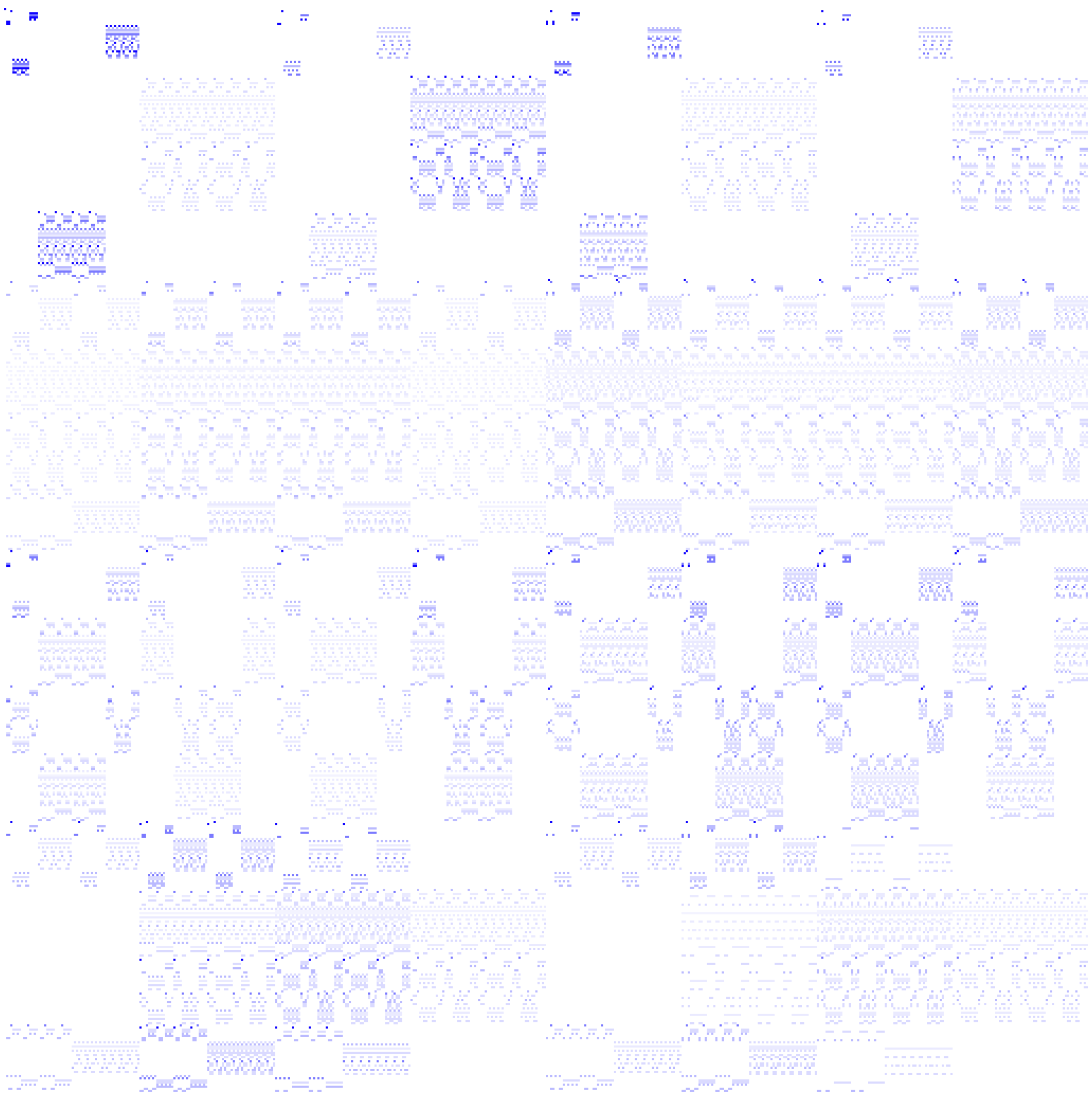

Figure 10: DryGASCON256 sbox differential distribution heat map

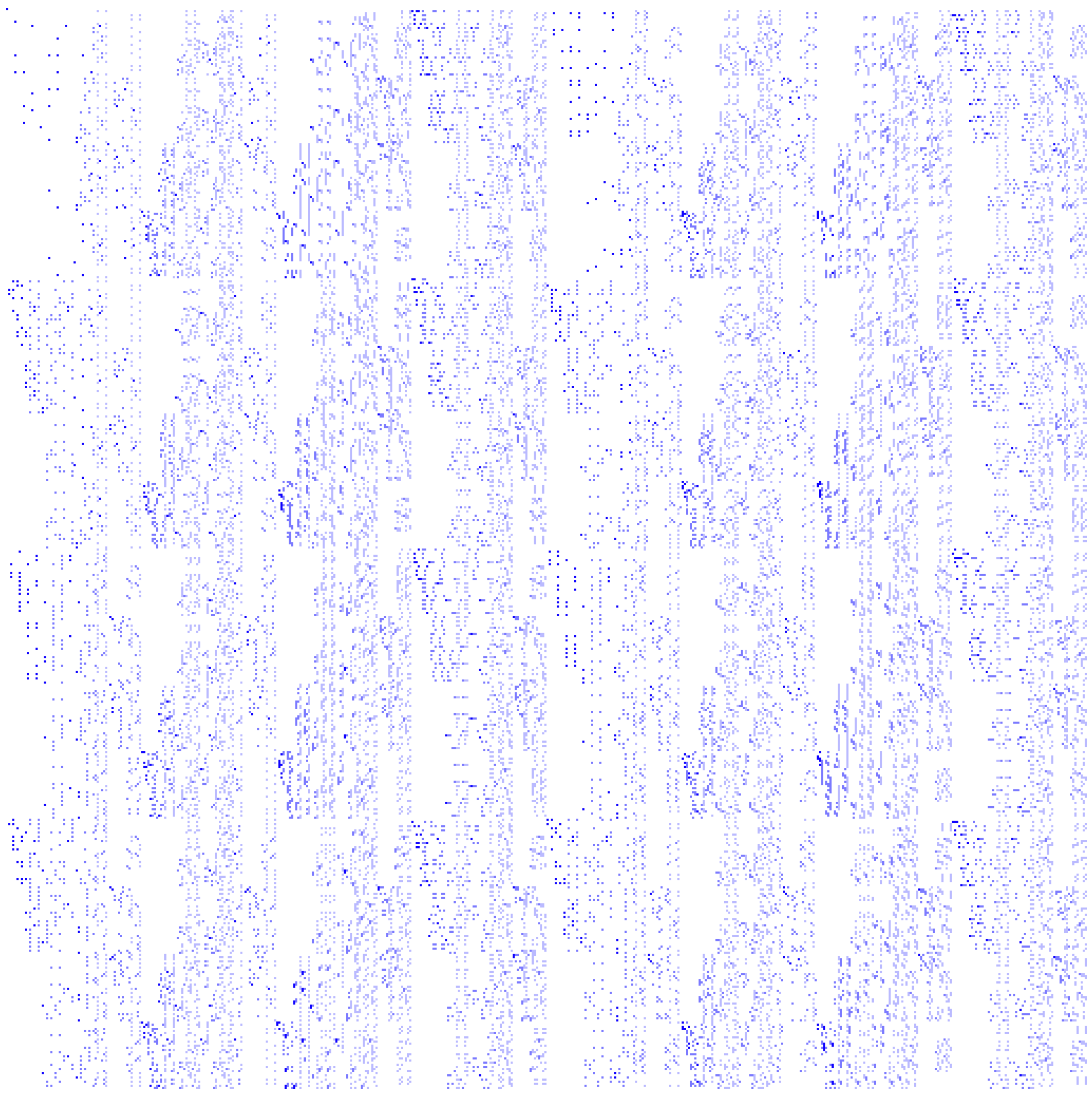
Figure 11: DryGASCON256 sbox linear approximation heat map

14.2 FPGA synthesis reports

Listing 5: Utlization report

```
Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.
--------------------------------------------------------------------------------
| Tool Version : Vivado v.2018.1 (lin64) Build 2188600 Wed Apr  4 18:39:19 MDT 2018
| Date         : Mon Feb 18 22:46:04 2019
| Host         : lafite running 64-bit Ubuntu 16.04.4 LTS
| Command      : report_utilization -cells u_impl -file drygascon128_150
    ↪ MHz_utilization_report.txt
| Design       : wrapper
| Device       : 7z020clg400-3
| Design State : Routed
--------------------------------------------------------------------------------

Utilization Design Information

Table of Contents
-----------------
1. Slice Logic
1.1 Summary of Registers by Type
2. Slice Logic Distribution
3. Memory
4. DSP
5. IO and GT Specific
6. Clocking
7. Specific Feature
8. Primitives
9. Black Boxes
10. Instantiated Netlists

1. Slice Logic
--------------

+-------------------------+------+-------+-----------+-------+
|        Site Type        | Used | Fixed | Available | Util% |
+-------------------------+------+-------+-----------+-------+
| Slice LUTs              | 1647 |     0 |     53200 |  3.10 |
|   LUT as Logic          | 1647 |     0 |     53200 |  3.10 |
|   LUT as Memory         |    0 |     0 |     17400 |  0.00 |
| Slice Registers         |  625 |     0 |    106400 |  0.59 |
|   Register as Flip Flop |  625 |     0 |    106400 |  0.59 |
|   Register as Latch     |    0 |     0 |    106400 |  0.00 |
| F7 Muxes                |  150 |     0 |     26600 |  0.56 |
| F8 Muxes                |   10 |     0 |     13300 |  0.08 |
+-------------------------+------+-------+-----------+-------+


1.1 Summary of Registers by Type
--------------------------------

+-------+--------------+-------------+--------------+
| Total | Clock Enable | Synchronous | Asynchronous |
+-------+--------------+-------------+--------------+
| 0     |            _ |           - |            - |
| 0     |            _ |           - |          Set |
| 0     |            _ |           - |        Reset |
| 0     |            _ |         Set |            - |
| 0     |            _ |       Reset |            - |
| 0     |          Yes |           - |            - |
| 0     |          Yes |           - |          Set |
| 0     |          Yes |           - |        Reset |
| 1      |          Yes |         Set |             - |
| 624    |          Yes |       Reset |             - |
+--------+--------------+-------------+---------------+


2. Slice Logic Distribution
---------------------------

+-------------------------------------------+------+-------+-----------+-------+
|                 Site Type                 | Used | Fixed | Available | Util% |
+-------------------------------------------+------+-------+-----------+-------+
| Slice                                     |  453 |     0 |     13300 |  3.41 |
|   SLICEL                                  |  231 |     0 |           |       |
|   SLICEM                                  |  222 |     0 |           |       |
| LUT as Logic                              | 1647 |     0 |     53200 |  3.10 |
|   using O5 output only                    |    0 |       |           |       |
|   using O6 output only                    | 1238 |       |           |       |
|   using O5 and O6                         |  409 |       |           |       |
| LUT as Memory                             |    0 |     0 |     17400 |  0.00 |
|   LUT as Distributed RAM                  |    0 |     0 |           |       |
|   LUT as Shift Register                   |    0 |     0 |           |       |
| LUT Flip Flop Pairs                       |  584 |     0 |     53200 |  1.10 |
|   fully used LUT-FF pairs                 |    0 |       |           |       |
|   LUT-FF pairs with one unused LUT output |  579 |       |           |       |
|   LUT-FF pairs with one unused Flip Flop  |  584 |       |           |       |
| Unique Control Sets                       |   18 |       |           |       |
+-------------------------------------------+------+-------+-----------+-------+
* Note: Review the Control Sets Report for more information regarding control sets.


3. Memory
---------

+----------------+------+-------+-----------+-------+
|    Site Type   | Used | Fixed | Available | Util% |
+----------------+------+-------+-----------+-------+
| Block RAM Tile |    0 |     0 |       140 |  0.00 |
|   RAMB36/FIFO* |    0 |     0 |       140 |  0.00 |
|   RAMB18       |    0 |     0 |       280 |  0.00 |
+----------------+------+-------+-----------+-------+
* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate
    ↪  only one FIFO36E1 or one FIFO18E1. However, if a FIFO18E1 occupies a Block RAM
    ↪ Tile, that tile can still accommodate a RAMB18E1


4. DSP
------

+-----------+------+-------+-----------+-------+
| Site Type | Used | Fixed | Available | Util% |
+-----------+------+-------+-----------+-------+
| DSPs      |    0 |     0 |       220 |  0.00 |
+-----------+------+-------+-----------+-------+


5. IO and GT Specific
---------------------

+-----------------------------+------+-------+-----------+-------+
|          Site Type          | Used | Fixed | Available | Util% |
+-----------------------------+------+-------+-----------+-------+
| Bonded IOB                  |    0 |     0 |       125 |  0.00 |
| Bonded IPADs                |    0 |     0 |         2 |  0.00 |
| Bonded IOPADs               |    0 |     0 |       130 |  0.00 |
| PHY_CONTROL                 |    0 |     0 |         4 |  0.00 |
| PHASER_REF                  |    0 |     0 |         4 |  0.00 |
| OUT_FIFO                    |    0 |     0 |        16 |  0.00 |
| IN_FIFO                     |    0 |     0 |        16 |  0.00 |
| IDELAYCTRL                  |    0 |     0 |         4 |  0.00 |
| IBUFDS                      |    0 |     0 |       121 |  0.00 |
| PHASER_OUT/PHASER_OUT_PHY   |    0 |     0 |        16 |  0.00 |
| PHASER_IN/PHASER_IN_PHY     |    0 |     0 |        16 |  0.00 |
| IDELAYE2/IDELAYE2_FINEDELAY |    0 |     0 |       200 |  0.00 |
| ILOGIC                      |    0 |     0 |       125 |  0.00 |
| OLOGIC                      |    0 |     0 |       125 |  0.00 |
+-----------------------------+------+-------+-----------+-------+


6. Clocking
-----------

+------------+------+-------+-----------+-------+
|  Site Type | Used | Fixed | Available | Util% |
+------------+------+-------+-----------+-------+
| BUFGCTRL   |    0 |     0 |        32 |  0.00 |
| BUFIO      |    0 |     0 |        16 |  0.00 |
| MMCME2_ADV |    0 |     0 |         4 |  0.00 |
| PLLE2_ADV  |    0 |     0 |         4 |  0.00 |
| BUFMRCE    |    0 |     0 |         8 |  0.00 |
| BUFHCE     |    0 |     0 |        72 |  0.00 |
| BUFR       |    0 |     0 |        16 |  0.00 |
+------------+------+-------+-----------+-------+


7. Specific Feature
-------------------

+-------------+------+-------+-----------+-------+
|  Site Type  | Used | Fixed | Available | Util% |
+-------------+------+-------+-----------+-------+
| BSCANE2     |    0 |     0 |         4 |  0.00 |
| CAPTUREE2   |    0 |     0 |         1 |  0.00 |
| DNA_PORT    |    0 |     0 |         1 |  0.00 |
| EFUSE_USR   |    0 |     0 |         1 |  0.00 |
| FRAME_ECCE2 |    0 |     0 |         1 |  0.00 |
| ICAPE2      |    0 |     0 |         2 |  0.00 |
| STARTUPE2   |    0 |     0 |         1 |  0.00 |
| XADC        |    0 |     0 |         1 |  0.00 |
+-------------+------+-------+-----------+-------+


8. Primitives
-------------

+----------+------+---------------------+
| Ref Name | Used | Functional Category |
+----------+------+---------------------+
| LUT6     |  922 |                 LUT |
| LUT5     |  678 |                 LUT |
| FDRE     |  624 |        Flop & Latch |
| LUT3     |  267 |                 LUT |
| MUXF7    |  150 |               MuxFx |
| LUT4     |  136 |                 LUT |
| LUT2     |   53 |                 LUT |
| MUXF8    |   10 |               MuxFx |
| CARRY4   |    3 |          CarryLogic |
| FDSE     |    1 |        Flop & Latch |
+----------+------+---------------------+


9. Black Boxes
--------------

+----------+------+
| Ref Name | Used |
+----------+------+


10. Instantiated Netlists
-------------------------

+----------+------+
| Ref Name | Used |
+----------+------+
```

Listing 6: Timing report

```
Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.
-----------------------------------------------------------------------------------------------
    ↪
| Tool Version : Vivado v.2018.1 (lin64) Build 2188600 Wed Apr  4 18:39:19 MDT 2018
| Date         : Mon Feb 18 22:46:22 2019
| Host         : lafite running 64-bit Ubuntu 16.04.4 LTS
| Command      : report_timing_summary -file drygascon128_150MHz_timing_report.txt -cell
    ↪ u_impl
| Design       : wrapper
| Device       : 7z020-clg400
| Speed File   : -3  PRODUCTION 1.11 2014-09-11
-----------------------------------------------------------------------------------------------
    ↪

Timing Summary Report

-----------------------------------------------------------------------------------------------
    ↪
| Timer Settings
| --------------
-----------------------------------------------------------------------------------------------
    ↪

  Enable Multi Corner Analysis               :  Yes
  Enable Pessimism Removal                   :  Yes
  Pessimism Removal Resolution               :  Nearest Common Node
  Enable Input Delay Default Clock           :  No
  Enable Preset / Clear Arcs                 :  No
  Disable Flight Delays                      :  No
  Ignore I/O Paths                           :  No
  Timing Early Launch at Borrowing Latches   :  false

  Corner  Analyze    Analyze
  Name    Max Paths  Min Paths
  ------  ---------  ---------
  Slow    Yes        Yes
  Fast    Yes        Yes


-----------------------------------------------------------------------------------------------
    ↪
```

```
| Design Timing Summary
| ---------------------
------------------------------------------------------------------------------------------------
    ↪

WNS(ns)  TNS(ns)  TNS Failing Endpoints  TNS Total Endpoints  WHS(ns)  THS(ns)  THS Failing
    ↪ Endpoints
-------  -------  ---------------------  -------------------  -------  -------
    ↪ ---------------------
  0.131    0.000                      0                 1416    0.186    0.000
    ↪     0

THS Total Endpoints  WPWS(ns)  TPWS(ns)  TPWS Failing Endpoints  TPWS Total Endpoints
-------------------  --------  --------  ----------------------  --------------------
               1416     2.250     0.000                       0                   625

All user specified timing constraints are met.


------------------------------------------------------------------------------------------------
    ↪
| Clock Summary
| -------------
------------------------------------------------------------------------------------------------
    ↪

Clock  Waveform(ns)         Period(ns)      Frequency(MHz)
-----  ------------         ----------      --------------
clk    {0.000 3.750}        6.500           153.846


------------------------------------------------------------------------------------------------
    ↪
| Intra Clock Table
| -----------------
------------------------------------------------------------------------------------------------
    ↪

Clock  WNS(ns)  TNS(ns)  TNS Failing Endpoints  TNS Total Endpoints  WHS(ns)  THS(ns)
-----  -------  -------  ---------------------  -------------------  -------  -------
clk      0.131    0.000                      0                 1416    0.186    0.000

THS Failing Endpoints  THS Total Endpoints  WPWS(ns)  TPWS(ns)  TPWS Failing Endpoints  TPWS
    ↪ Total Endpoints
---------------------  -------------------  --------  --------  ----------------------
    ↪ --------------------
                    0                 1416     2.250     0.000                       0
    ↪          625


------------------------------------------------------------------------------------------------
    ↪
| Timing Details
| --------------
------------------------------------------------------------------------------------------------
    ↪


---------------------------------------------------------------------------------------------------
    ↪
From Clock:  clk
  To Clock:  clk
```

```
Setup :           0  Failing Endpoints,  Worst Slack       0.131ns,  Total Violation
    ↪    0.000ns
Hold  :           0  Failing Endpoints,  Worst Slack       0.186ns,  Total Violation
    ↪    0.000ns
PW    :           0  Failing Endpoints,  Worst Slack       2.250ns,  Total Violation
    ↪    0.000ns
-----------------------------------------------------------------------------------------------
    ↪


Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             0.131ns  (required time - arrival time)
  Source:                 u_impl/cnt_reg[2]/C
                            (rising edge-triggered cell FDRE clocked by clk  {rise@0.000ns
    ↪ fall@3.750ns period=6.500ns})
  Destination:            u_impl/c_reg[168]/CE
                            (rising edge-triggered cell FDRE clocked by clk  {rise@0.000ns
    ↪ fall@3.750ns period=6.500ns})
  Path Group:             clk
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            6.500ns  (clk rise@6.500ns - clk rise@0.000ns)
  Data Path Delay:        6.132ns  (logic 0.587ns (9.572%)  route 5.545ns (90.428%))
  Logic Levels:           2  (LUT5=1 LUT6=1)
  Clock Path Skew:        -0.051ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    3.525ns = ( 10.025 - 6.500 )
    Source Clock Delay      (SCD):    3.954ns
    Clock Pessimism Removal (CPR):    0.377ns
  Clock Uncertainty:      0.035ns  ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE
    Total System Jitter     (TSJ):    0.071ns
    Total Input Jitter      (TIJ):    0.000ns
    Discrete Jitter          (DJ):    0.000ns
    Phase Error              (PE):    0.000ns

    Location             Delay type                Incr(ns)  Path(ns)    Netlist Resource(s
    ↪ )
  -------------------------------------------------------------------
    ↪ -------------------
                         (clock clk rise edge)        0.000     0.000 r
    U7                                                0.000     0.000 r  clk (IN)
                         net (fo=0)                   0.000     0.000    clk
    U7                   IBUF (Prop_ibuf_I_O)         0.908     0.908 r  clk_IBUF_inst/O
                         net (fo=1, routed)           1.636     2.544    clk_IBUF
    BUFGCTRL_X0Y0        BUFG (Prop_bufg_I_O)         0.079     2.623 r  clk_IBUF_BUFG_inst
    ↪ /O
                         net (fo=707, routed)         1.331     3.954    u_impl/
    ↪ clk_IBUF_BUFG
    SLICE_X8Y18          FDRE                                           r  u_impl/cnt_reg[2]/
    ↪ C
  -------------------------------------------------------------------
    ↪ -------------------
    SLICE_X8Y18          FDRE (Prop_fdre_C_Q)         0.393     4.347 r  u_impl/cnt_reg[2]/
    ↪ Q
                         net (fo=460, routed)         2.246     6.592    u_impl/cnt_reg_n_0
    ↪ _[2]
    SLICE_X3Y17          LUT6 (Prop_lut6_I2_O)        0.097     6.689 r  u_impl/c[191]_i_3/
    ↪ O
                         net (fo=1, routed)           0.302     6.991    u_impl/c[191]_i_3
    ↪ _n_0
    SLICE_X3Y17          LUT5 (Prop_lut5_I4_O)        0.097     7.088 r  u_impl/c[191]_i_1/
    ↪ O
                         net (fo=32, routed)          2.998    10.086    u_impl/c[191]_i_1
    ↪ _n_0
    SLICE_X11Y35         FDRE                                         r  u_impl/c_reg[168]/
     ↪ CE
    -------------------------------------------------------------------
     ↪ -------------------

                         (clock clk rise edge)        6.500     6.500 r
    U7                                                0.000     6.500 r  clk (IN)
                         net (fo=0)                   0.000     6.500    clk
    U7                   IBUF (Prop_ibuf_I_O)         0.776     7.276 r  clk_IBUF_inst/O
                         net (fo=1, routed)           1.484     8.759    clk_IBUF
    BUFGCTRL_X0Y0        BUFG (Prop_bufg_I_O)         0.072     8.831 r  clk_IBUF_BUFG_inst
     ↪ /O
                         net (fo=707, routed)         1.194    10.025    u_impl/
     ↪ clk_IBUF_BUFG
    SLICE_X11Y35         FDRE                                         r  u_impl/c_reg[168]/
     ↪ C
                         clock pessimism              0.377    10.403
                         clock uncertainty           -0.035    10.367
    SLICE_X11Y35         FDRE (Setup_fdre_C_CE)      -0.150    10.217    u_impl/c_reg[168]
  -------------------------------------------------------------------
                         required time                         10.217
                         arrival time                         -10.086
  -------------------------------------------------------------------
                         slack                                  0.131





Min Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :              0.186ns  (arrival time - required time)
  Source:                  ios_reg[79]/C
                            (rising edge-triggered cell FDRE clocked by clk  {rise@0.000ns
     ↪ fall@3.750ns period=6.500ns})
  Destination:             do_reg/D
                            (rising edge-triggered cell FDRE clocked by clk  {rise@0.000ns
     ↪ fall@3.750ns period=6.500ns})
  Path Group:              clk
  Path Type:               Hold (Min at Fast Process Corner)
  Requirement:             0.000ns  (clk rise@0.000ns - clk rise@0.000ns)
  Data Path Delay:         0.290ns  (logic 0.209ns (72.067%)  route 0.081ns (27.933%))
  Logic Levels:            1  (LUT4=1)
  Clock Path Skew:         0.013ns (DCD - SCD - CPR)
    Destination Clock Delay (DCD):    2.116ns
    Source Clock Delay      (SCD):    1.597ns
    Clock Pessimism Removal (CPR):    0.506ns

    Location             Delay type                Incr(ns)  Path(ns)    Netlist Resource(s
     ↪ )
  -------------------------------------------------------------------
     ↪ -------------------
                         (clock clk rise edge)        0.000     0.000 r
    U7                                                0.000     0.000 r  clk (IN)
                         net (fo=0)                   0.000     0.000    clk
    U7                   IBUF (Prop_ibuf_I_O)         0.286     0.286 r  clk_IBUF_inst/O
                         net (fo=1, routed)           0.663     0.949    clk_IBUF
    BUFGCTRL_X0Y0        BUFG (Prop_bufg_I_O)         0.026     0.975 r  clk_IBUF_BUFG_inst
     ↪ /O
                         net (fo=707, routed)         0.622     1.597    clk_IBUF_BUFG
    SLICE_X0Y8           FDRE                                         r  ios_reg[79]/C
  -------------------------------------------------------------------
     ↪ -------------------
```

```
    SLICE_X0Y8           FDRE (Prop_fdre_C_Q)         0.164     1.761 r  ios_reg[79]/Q
                         net (fo=1, routed)           0.081     1.842    u_impl/Q[79]
    SLICE_X1Y8           LUT4 (Prop_lut4_I0_O)        0.045     1.887 r  u_impl/do_i_1/O
                         net (fo=1, routed)           0.000     1.887    u_impl_n_0
    SLICE_X1Y8           FDRE                                          r  do_reg/D
  -------------------------------------------------------------------
    ↪ -------------------

                         (clock clk rise edge)        0.000     0.000 r
    U7                                                0.000     0.000 r  clk (IN)
                         net (fo=0)                   0.000     0.000    clk
    U7                   IBUF (Prop_ibuf_I_O)         0.476     0.476 r  clk_IBUF_inst/O
                         net (fo=1, routed)           0.719     1.195    clk_IBUF
    BUFGCTRL_X0Y0        BUFG (Prop_bufg_I_O)         0.029     1.224 r  clk_IBUF_BUFG_inst
    ↪ /O
                         net (fo=707, routed)         0.892     2.116    clk_IBUF_BUFG
    SLICE_X1Y8           FDRE                                          r  do_reg/C
                         clock pessimism             -0.506     1.610
    SLICE_X1Y8           FDRE (Hold_fdre_C_D)         0.091     1.701    do_reg
  -------------------------------------------------------------------
                         required time                         -1.701
                         arrival time                           1.887
  -------------------------------------------------------------------
                         slack                                  0.186





Pulse Width Checks
--------------------------------------------------------------------------------------
Clock Name:         clk
Waveform(ns):       { 0.000 3.750 }
Period(ns):         6.500
Sources:            { clk }

Check Type        Corner  Lib Pin  Reference Pin  Required(ns)  Actual(ns)  Slack(ns)
    ↪ Location     Pin
Min Period        n/a     FDRE/C   n/a            1.000         6.500       5.500
    ↪ SLICE_X4Y29  u_impl/c_reg[0]/C
Low Pulse Width   Slow    FDRE/C   n/a            0.500         2.750       2.250
    ↪ SLICE_X7Y37  u_impl/c_reg[99]/C
High Pulse Width  Fast    FDRE/C   n/a            0.500         3.750       3.250
    ↪ SLICE_X4Y30  u_impl/c_reg[100]/C
```

14.3 ARM Cortex M0 assembly listings

Listing 7: DryGASCON128 G function

```
.cpu cortex-m0
.syntax unified
.code    16
.thumb_func

.align    1
.global    drygascon128_g
.global    drygascon128_f

    .equ C0, 0
    .equ C1, 8
    .equ C2, 16
    .equ C3, 24
    .equ C4, 32
    .equ R0, 40
    .equ R1, 48
    .equ X0, 56
    .equ X1, 64
    .equ X2, 72
    .equ X3, 80

    .equ C0L, C0
    .equ C1L, C1
    .equ C2L, C2
    .equ C3L, C3
    .equ C4L, C4
    .equ R0L, R0
    .equ R1L, R1
    .equ X0L, X0
    .equ X1L, X1
    .equ X2L, X2
    .equ X3L, X3

    .equ C0H, C0+4
    .equ C1H, C1+4
    .equ C2H, C2+4
    .equ C3H, C3+4
    .equ C4H, C4+4
    .equ R0H, R0+4
    .equ R1H, R1+4
    .equ X0H, X0+4
    .equ X1H, X1+4
    .equ X2H, X2+4
    .equ X3H, X3+4

    .equ R32_0,R0L
    .equ R32_1,R0H
    .equ R32_2,R1L
    .equ R32_3,R1H

.type    drygascon128_g, %function
drygascon128_g:
    //r0: state: c,r,x
    //r1: rounds
    push    {r4, r5, r6, r7, lr}
    //stack vars:
    // 8 round
    // 4 rounds
    // 0 state address

    //r=0
    movs    r5,#0
    str     r5,[r0,#R32_0]
    str     r5,[r0,#R32_1]
    str     r5,[r0,#R32_2]
    str     r5,[r0,#R32_3]

    //round=r5=rounds-1;
    subs    r6,r1,#1
    //base = round_cst+12-rounds
    adr     r5, round_cst
    adds    r5,r5,#12
    subs    r5,r5,r1

    push    {r0,r5,r6}
    ldr     r4,[r0,#C4L]
    ldr     r3,[r0,#C3L]
    ldr     r2,[r0,#C2L]
    ldr     r1,[r0,#C1L]
    ldr     r0,[r0,#C0L]

    //loop entry
    //assume r1>0 at entry
drygascon128_g_main_loop:
    //r0~r4: lower half of each words of the state
    //r5: base for round constants
    //r6: round, counting from rounds-1 to 0

    //r6 = ((0xf - r6) << 4) | r6;
    ldrb    r6,[r5,r6]
    // addition of round constant
    eors    r2,r2,r6

    // substitution layer, lower half
    eors    r0,r0,r4
    eors    r4,r4,r3
    eors    r2,r2,r1

    mvns    r5,r0
    mvns    r6,r3
    mvns    r7,r4
    ands    r5,r5,r1
    ands    r6,r6,r4
    eors    r4,r4,r5

    ands    r7,r7,r0
    mvns    r5,r2
    ands    r5,r5,r3
    eors    r3,r3,r7

    mvns    r7,r1
    ands    r7,r7,r2
    eors    r2,r2,r6

    eors    r3,r3,r2
    mvns    r2,r2

    eors    r0,r0,r7
    eors    r1,r1,r5
    eors    r1,r1,r0
    eors    r0,r0,r4

    ldr     r7,[sp,#0]
    str     r4,[r7,#C4L]
    str     r3,[r7,#C3L]
    str     r2,[r7,#C2L]
    str     r1,[r7,#C1L]
    str     r0,[r7,#C0L]

    ldr     r4,[r7,#C4H]
    ldr     r3,[r7,#C3H]
    ldr     r2,[r7,#C2H]
    ldr     r1,[r7,#C1H]
    ldr     r0,[r7,#C0H]

    // substitution layer, upper half
    eors    r0,r0,r4
    eors    r4,r4,r3
    eors    r2,r2,r1

    mvns    r5,r0
    mvns    r6,r3
    mvns    r7,r4
    ands    r5,r5,r1
    ands    r6,r6,r4
    eors    r4,r4,r5

    ands    r7,r7,r0
    mvns    r5,r2
    ands    r5,r5,r3
    eors    r3,r3,r7

    mvns    r7,r1
    ands    r7,r7,r2
    eors    r2,r2,r6

    eors    r3,r3,r2
    mvns    r2,r2

    eors    r0,r0,r7
    eors    r1,r1,r5
    eors    r1,r1,r0
    eors    r0,r0,r4

    // linear diffusion layer
    ldr     r7,[sp,#0]

    //c4 ^= gascon_rotr64_interleaved(c4, 40) ^ gascon_rotr64_interleaved(c4, 7);
    //c4 high part
    movs    r6,r4
    movs    r5,#(20)
    rors    r4,r4,r5
    eors    r6,r6,r4
    ldr     r5,[r7,#C4L]
    movs    r7,#(4)
    rors    r5,r5,r7
    eors    r6,r6,r5
    ldr     r7,[sp,#0]
    str     r6,[r7,#C4H]
    //c4 low part
    movs    r7,#(32-4)
    rors    r5,r5,r7
    movs    r6,r5
    movs    r7,#((32-20+3)%32)
    rors    r4,r4,r7
    eors    r4,r4,r6
    movs    r7,#(20)
    rors    r5,r5,r7
    eors    r4,r4,r5
    ldr     r7,[sp,#0]
    str     r4,[r7,#C4L]

    //c0 ^= gascon_rotr64_interleaved(c0, 28) ^ gascon_rotr64_interleaved(c0, 19);
    //c0 high part
    movs    r6,r0
    movs    r5,#(14)
    rors    r0,r0,r5
    eors    r6,r6,r0
    ldr     r5,[r7,#C0L]
    movs    r4,#(10)
```

```
rors    r5,r5,r4
eors    r6,r6,r5
str     r6,[r7,#C0H]
ldr     r4,[r7,#R32_1]
eors    r4,r4,r6
str     r4,[r7,#R32_1]
//c0 low part
movs    r4,#(32-10)
rors    r5,r5,r4
movs    r6,r5
movs    r4,#((32-14+9)%32)
rors    r0,r0,r4
eors    r0,r0,r6
movs    r4,#(14)
rors    r5,r5,r4
eors    r0,r0,r5
ldr     r4,[r7,#R32_0]
eors    r4,r4,r0
str     r4,[r7,#R32_0]

//c1 ^= gascon_rotr64_interleaved(c1, 38) ^ gascon_rotr64_interleaved(c1, 61);
//c1 high part
movs    r6,r1
movs    r5,#(19)
rors    r1,r1,r5
eors    r6,r6,r1
ldr     r5,[r7,#C1L]
movs    r4,#(31)
rors    r5,r5,r4
eors    r6,r6,r5
str     r6,[r7,#C1H]
ldr    r4,[r7,#R32_3]
eors    r4,r4,r6
str     r4,[r7,#R32_3]
//c1 low part
movs    r4,#(32-31)
rors    r5,r5,r4
movs    r6,r5
movs    r4,#((32-19+30)%32)
rors    r1,r1,r4
eors    r1,r1,r6
movs    r4,#(19)
rors    r5,r5,r4
eors    r1,r1,r5
ldr     r4,[r7,#R32_2]
eors    r4,r4,r1
str     r4,[r7,#R32_2]

//c2 ^= gascon_rotr64_interleaved(c2, 6) ^ gascon_rotr64_interleaved(c2, 1);
//c2 high part
movs    r6,r2
movs    r5,#(3)
rors    r2,r2,r5
eors    r6,r6,r2
ldr     r5,[r7,#C2L]
movs    r4,#(1)
rors    r5,r5,r4
eors    r6,r6,r5
str     r6,[r7,#C2H]
ldr     r4,[r7,#R32_0]
eors    r4,r4,r6
str     r4,[r7,#R32_0]
//c2 low part
```

```
    movs    r4,#(32-1)
    rors    r5,r5,r4
    movs    r6,r5
    movs    r4,#((32-3+0)%32)
    rors    r2,r2,r4
    eors    r2,r2,r6
    movs    r4,#(3)
    rors    r5,r5,r4
    eors    r2,r2,r5
    ldr     r4,[r7,#R32_3]
    eors    r4,r4,r2
    str     r4,[r7,#R32_3]

    //c3 ^= gascon_rotr64_interleaved(c3, 10) ^ gascon_rotr64_interleaved(c3, 17);
    //c3 high part
    movs    r6,r3
    movs    r5,#(5)
    rors    r3,r3,r5
    eors    r6,r6,r3
    ldr     r5,[r7,#C3L]
    movs    r4,#(9)
    rors    r5,r5,r4
    eors    r6,r6,r5
    str     r6,[r7,#C3H]
    ldr     r4,[r7,#R32_2]
    eors    r4,r4,r6
    str     r4,[r7,#R32_2]
    //c3 low part
    movs    r4,#(32-9)
    rors    r5,r5,r4
    movs    r6,r5
    movs    r4,#((32-5+8)%32)
    rors    r3,r3,r4
    eors    r3,r3,r6
    movs    r4,#(5)
    rors    r5,r5,r4
    eors    r3,r3,r5
    ldr     r4,[r7,#R32_1]
    eors    r4,r4,r3
    str     r4,[r7,#R32_1]

    ldr     r4,[r7,#C4L]
    ldr     r5,[sp,#4]

    ldr     r6,[sp,#8]
    subs    r6,#1
    bmi     drygascon128_g_exit

    str     r6,[sp,#8]
    b       drygascon128_g_main_loop
drygascon128_g_exit:

    str     r3,[r7,#C3L]
    str     r2,[r7,#C2L]
    str     r1,[r7,#C1L]
    str     r0,[r7,#C0L]

    add     sp,sp,#12
    pop     {r4, r5, r6, r7, pc}
.size   drygascon128_g, .-drygascon128_g
```

Listing 8: DryGASCON128 *G* function

```
.align 2
.type   drygascon128_f, %function
drygascon128_f:
    //r0:state c r x
    //r1:input -> shall be 32 bit aligned
    //r2:ds
    //r3:rounds
    push    {r4, r5, r6, r7, lr}

    //stack frame:
    //0 ~ 28-1: buf
    //28 :pointer on c
    //32 : rounds for g
    //36 :mix round / g round

    movs    r4,#26
    push    {r0,r3,r4}
    sub     sp,sp,#28

    //load 10 bit mask in r4 = 0x3FF
    movs    r4,#0xFF
    lsls    r4,r4,#2
    adds    r4,r4,#3

    movs    r7,#0
    //r=0
    str     r7,[r0,#R32_0]
    str     r7,[r0,#R32_1]
    str     r7,[r0,#R32_2]
    str     r7,[r0,#R32_3]

    //r7 = sp
    add     r7,r7,sp

    ldr     r3,[r1]
    movs    r5,r4
    ands    r5,r5,r3
    strh    r5,[r7,#0+26]

    lsrs    r3,r3,#10
    movs    r5,r4
    ands    r5,r5,r3
    strh    r5,[r7,#0+24]

    lsrs    r3,r3,#10
    movs    r5,r4
    ands    r5,r5,r3
    strh    r5,[r7,#0+22]

    lsrs    r5,r3,#10
    ldr     r3,[r1,#4]
    lsls    r6,r3,#2
    lsrs    r3,r3,#8
    orrs    r6,r6,r5
    movs    r5,r4
    ands    r5,r5,r6
    strh    r5,[r7,#0+20]

    movs    r5,r4
    ands    r5,r5,r3
    strh    r5,[r7,#0+18]
    lsrs    r3,r3,#10
    movs    r5,r4
    ands    r5,r5,r3
    strh    r5,[r7,#0+16]

    lsrs    r5,r3,#10
    ldr     r3,[r1,#8]
    lsls    r6,r3,#4
    lsrs    r3,r3,#6
    orrs    r6,r6,r5
    movs    r5,r4
    ands    r5,r5,r6
    strh    r5,[r7,#0+14]

    movs    r5,r4
    ands    r5,r5,r3
    strh    r5,[r7,#0+12]

    lsrs    r3,r3,#10
    movs    r5,r4
    ands    r5,r5,r3
    strh    r5,[r7,#0+10]

    lsrs    r5,r3,#10
    ldr     r3,[r1,#12]
    lsls    r6,r3,#6
    lsrs    r3,r3,#4
    orrs    r6,r6,r5
    movs    r5,r4
    ands    r5,r5,r6
    strh    r5,[r7,#0+8]

    movs    r5,r4
    ands    r5,r5,r3
    strh    r5,[r7,#0+6]

    lsrs    r3,r3,#10
    movs    r5,r4
    ands    r5,r5,r3
    strh    r5,[r7,#0+4]

    lsrs    r5,r3,#10
    lsls    r6,r2,#8
    lsrs    r3,r2,#2
    orrs    r6,r6,r5
    movs    r5,r4
    ands    r5,r5,r6
    strh    r5,[r7,#0+2]

    movs    r5,r4
    ands    r5,r5,r3
    strh    r5,[r7,#0+0]

    movs    r7,#26
drygascon128_f_mix128_main_loop:
    movs    r6,#0
    add     r6,r6,sp
    ldrh    r6,[r6,r7]

    ldr     r5,[sp,#28]
    movs    r7,r5
    adds    r5,r5,#40+16
    movs    r4,#0xc

    lsls    r0,r6,#2
    ands    r0,r0,r4
    ldr     r1,[r5,r0]
    ldr     r0,[r7,#0*8]
    eors    r0,r0,r1

    lsrs    r1,r6,#0
    ands    r1,r1,r4
    ldr     r2,[r5,r1]
    ldr     r1,[r7,#1*8]
    eors    r1,r1,r2

    lsrs    r2,r6,#2
    ands    r2,r2,r4
    ldr     r3,[r5,r2]
    ldr     r2,[r7,#2*8]
    eors    r2,r2,r3

    lsrs    r3,r6,#4
    ands    r3,r3,r4
    ldr     r4,[r5,r3]
    ldr     r3,[r7,#3*8]
    eors    r3,r3,r4

    lsrs    r4,r6,#6+2
    lsls    r4,r4,#2
    ldr     r6,[r5,r4]
    ldr     r4,[r7,#4*8]
    eors    r4,r4,r6

    ldr     r6,[sp,#36]
    subs    r6,#2
    bpl     drygascon128_f_mix128_coreround
    b       drygascon128_f_mix128_exit
drygascon128_f_mix128_coreround:
    str     r6,[sp,#36]

    movs    r6,#0xf0
    // addition of round constant
    eors    r2,r2,r6

    // substitution layer, lower half
    eors    r0,r0,r4
    eors    r4,r4,r3
    eors    r2,r2,r1

    mvns    r5,r0
    mvns    r6,r3
    mvns    r7,r4
    ands    r5,r5,r1
    ands    r6,r6,r4
    eors    r4,r4,r5

    ands    r7,r7,r0
    mvns    r5,r2
    ands    r5,r5,r3
    eors    r3,r3,r7

    mvns    r7,r1
    ands    r7,r7,r2
    eors    r2,r2,r6
    eors    r3,r3,r2
    mvns    r2,r2

    eors    r0,r0,r7
    eors    r1,r1,r5
    eors    r1,r1,r0
    eors    r0,r0,r4

    ldr     r7,[sp,#28]
    str     r4,[r7,#C4L]
    str     r3,[r7,#C3L]
    str     r2,[r7,#C2L]
    str     r1,[r7,#C1L]
    str     r0,[r7,#C0L]

    ldr     r4,[r7,#C4H]
    ldr     r3,[r7,#C3H]
    ldr     r2,[r7,#C2H]
    ldr     r1,[r7,#C1H]
    ldr     r0,[r7,#C0H]

    // substitution layer, upper half
    eors    r0,r0,r4
    eors    r4,r4,r3
    eors    r2,r2,r1

    mvns    r5,r0
    mvns    r6,r3
    mvns    r7,r4
    ands    r5,r5,r1
    ands    r6,r6,r4
    eors    r4,r4,r5

    ands    r7,r7,r0
    mvns    r5,r2
    ands    r5,r5,r3
    eors    r3,r3,r7

    mvns    r7,r1
    ands    r7,r7,r2
    eors    r2,r2,r6

    eors    r3,r3,r2
    mvns    r2,r2

    eors    r0,r0,r7
    eors    r1,r1,r5
    eors    r1,r1,r0
    eors    r0,r0,r4

    // linear diffusion layer
    ldr     r7,[sp,#28]

    //c4 ^= gascon_rotr64_interleaved(c4, 40) ^ gascon_rotr64_interleaved(c4, 7);
    //c4 high part
    movs    r6,r4
    movs    r5,#(20)
    rors    r4,r4,r5
    eors    r6,r6,r4
    ldr     r5,[r7,#C4L]
    movs    r7,#(4)
    rors    r5,r5,r7
```

```
eors    r6,r6,r5
ldr     r7,[sp,#28]
str     r6,[r7,#C4H]
//c4 low part
movs    r7,#(32-4)
rors    r5,r5,r7
movs    r6,r5
movs    r7,#((32-20+3)%32)
rors    r4,r4,r7
eors    r4,r4,r6
movs    r7,#(20)
rors    r5,r5,r7
eors    r4,r4,r5
ldr     r7,[sp,#28]
str     r4,[r7,#C4L]

//c0 ^= gascon_rotr64_interleaved(c0, 28) ^ gascon_rotr64_interleaved(c0, 19);
//c0 high part
movs    r6,r0
movs    r5,#(14)
rors    r0,r0,r5
eors    r6,r6,r0
ldr     r5,[r7,#C0L]
movs    r4,#(10)
rors    r5,r5,r4
eors    r6,r6,r5
str     r6,[r7,#C0H]
//c0 low part
movs    r4,#(32-10)
rors    r5,r5,r4
movs    r6,r5
movs    r4,#((32-14+9)%32)
rors    r0,r0,r4
eors    r0,r0,r6
movs    r4,#(14)
rors    r5,r5,r4
eors    r0,r0,r5

//c1 ^= gascon_rotr64_interleaved(c1, 38) ^ gascon_rotr64_interleaved(c1, 61);
//c1 high part
movs    r6,r1
movs    r5,#(19)
rors    r1,r1,r5
eors    r6,r6,r1
ldr     r5,[r7,#C1L]
movs    r4,#(31)
rors    r5,r5,r4
eors    r6,r6,r5
str     r6,[r7,#C1H]
//c1 low part
movs    r4,#(32-31)
rors    r5,r5,r4
movs    r6,r5
movs    r4,#((32-19+30)%32)
rors    r1,r1,r4
eors    r1,r1,r6
movs    r4,#(19)
rors    r5,r5,r4
eors    r1,r1,r5

//c2 ^= gascon_rotr64_interleaved(c2, 6) ^ gascon_rotr64_interleaved(c2, 1);
//c2 high part
movs    r6,r2
```

```
    movs    r5,#(3)
    rors    r2,r2,r5
    eors    r6,r6,r2
    ldr     r5,[r7,#C2L]
    movs    r4,#(1)
    rors    r5,r5,r4
    eors    r6,r6,r5
    str     r6,[r7,#C2H]
    //c2 low part
    movs    r4,#(32-1)
    rors    r5,r5,r4
    movs    r6,r5
    movs    r4,#((32-3+0)%32)
    rors    r2,r2,r4
    eors    r2,r2,r6
    movs    r4,#(3)
    rors    r5,r5,r4
    eors    r2,r2,r5

    //c3 ^= gascon_rotr64_interleaved(c3, 10) ^ gascon_rotr64_interleaved(c3, 17);
    //c3 high part
    movs    r6,r3
    movs    r5,#(5)
    rors    r3,r3,r5
    eors    r6,r6,r3
    ldr     r5,[r7,#C3L]
    movs    r4,#(9)
    rors    r5,r5,r4
    eors    r6,r6,r5
    str     r6,[r7,#C3H]
    //c3 low part
    movs    r4,#(32-9)
    rors    r5,r5,r4
    movs    r6,r5
    movs    r4,#((32-5+8)%32)
    rors    r3,r3,r4
    eors    r3,r3,r6
    movs    r4,#(5)
    rors    r5,r5,r4
    eors    r3,r3,r5

    str     r3,[r7,#C3L]
    str     r2,[r7,#C2L]
    str     r1,[r7,#C1L]
    str     r0,[r7,#C0L]

    ldr     r7,[sp,#36]
    b       drygascon128_f_mix128_main_loop
drygascon128_f_mix128_exit:
    ldr     r7,[sp,#32]
    //round=r5=rounds-1;
    subs    r6,r7,#1
    //base = round_cst+12-rounds
    adr     r5, round_cst
    adds    r5,r5,#12
    subs    r5,r5,r7

    add     sp,sp,#28
    str     r5,[sp,#4]
    str     r6,[sp,#8]
    b       drygascon128_g_main_loop

.align 2
```

```
round_cst:
.byte 0x4b
.byte 0x5a
.byte 0x69
.byte 0x78
.byte 0x87
.byte 0x96
.byte 0xa5
.byte 0xb4
.byte 0xc3
.byte 0xd2
.byte 0xe1
.byte 0xf0
.align 2
.size    drygascon128_f, .-drygascon128_f
```

14.4 DryGascon128 AEAD test vectors

Listing 9: Detailed test vector: m=0, a=0, s=0, full key profile

```
Encrypting 0 bytes message with 0 bytes of associated data and 0 bytes of static data
Key size: 56 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F2021222
    ↪ 32425262728292A2B2C2D2E2F3031323334353637
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   F/G entry 0 (F with DS): padded=0, domain=1, finalize=1
   C[ 0] = 000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   C[ 1] = 2021222324252627
   X[ 0] = 28292A2B2C2D2E2F3031323334353637
       R = 00000000000000000000000000000000
       I = F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Final state:
   C[ 0] = 39CFA356FB50450916D5CC1ACE846209FC712BA8AE0779B6034DBA7779272D9D
   C[ 1] = CD63122514B04744
   X[ 0] = 28292A2B2C2D2E2F3031323334353637
       R = F1FBA3D719B00A49BF170F832EB7649F
   CipherText:
   Tag:             F1FBA3D719B00A49BF170F832EB7649F
```

Listing 10: Detailed test vector: m=0, a=0, s=0, fast key profile

```
Encrypting 0 bytes message with 0 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   F/G entry 0 (F with DS): padded=0, domain=1, finalize=1
   C[ 0] = 000102030405060708090A0B0C0D0E0F000102030405060708090A0B0C0D0E0F
   C[ 1] = 0001020304050607
   X[ 0] = 101112131415161718191A1B1C1D1E1F
       R = 00000000000000000000000000000000
       I = F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Final state:
   C[ 0] = 364FAC5A4D6914E83D8612A7BB7BD7F850238EE2BE8730DCB5C2EF70C0A4E76D
   C[ 1] = CBC44F9B2E3C420B
   X[ 0] = 101112131415161718191A1B1C1D1E1F
       R = 363001DCCA220E28470EA42E63C9A2AF
   CipherText:
   Tag:             363001DCCA220E28470EA42E63C9A2AF
```

Listing 11: Detailed test vector: m=0, a=0, s=0

```
Encrypting 0 bytes message with 0 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   F/G entry 0 (F with DS): padded=0, domain=1, finalize=1
   C[ 0] = 000102030405060708090A0B0C0D0E0F10FEFEFE8AF2F2F2FA171D872CB43FDF
   C[ 1] = 90C1F2A364656667
   X[ 0] = FA8EC1869CD9166FD9293729F9181919
       R = 00000000000000000000000000000000
       I = F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Final state:
   C[ 0] = 8232FF43CE5594AC4733AB6F2A6FE8AB3080443B1EF9FC1FB2D692ADAA181190
   C[ 1] = 4B34249457CC4F76
   X[ 0] = FA8EC1869CD9166FD9293729F9181919
       R = 98AF3B3FB2000908C698C3C595FAB6E8
   CipherText:
   Tag:             98AF3B3FB2000908C698C3C595FAB6E8
```

Listing 12: Detailed test vector: m=0, a=0, s=1

```
Encrypting 0 bytes message with 0 bytes of associated data and 1 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Static data:     BB
   Padded S. data:  BB010000000000000000000000000000
   F/G entry 0 (F with DS): padded=1, domain=2, finalize=1
   C[ 0] = 000102030405060708090A0B0C0D0E0F10FEFEFE8AF2F2F2FA171D872CB43FDF
   C[ 1] = 90C1F2A364656667
   X[ 0] = FA8EC1869CD9166FD9293729F9181919
       R = 00000000000000000000000000000000
       I = BB010000000000000000000000000000
   F/G entry 1 (G):
   C[ 0] = B1319FF260A01D9F6F26FF701056867F6242B53ED4E12A4EC40EE2373E2CE2BD
   C[ 1] = 0A2D8D7190EC8AEF
   X[ 0] = FA8EC1869CD9166FD9293729F9181919
       R = 67778E18D165B2CF5F43A90E03B1D1EB
   F/G entry 2 (G):
   C[ 0] = 5A7A82A62BBE002B60B97B56902CCE84630B3726B86EAA004FFE3390EC88900A
   C[ 1] = 335673C555A85657
   X[ 0] = FA8EC1869CD9166FD9293729F9181919
       R = DEEEE5D8E9E3DC32D2275E46A9435A1E
   F/G entry 3 (F with DS): padded=0, domain=1, finalize=0
   C[ 0] = 67778E18D165B2CF5F43A90E03B1D1EBDEEEE5D8E9E3DC32D2275E46A9435A1E
   C[ 1] = 9F955DCEDAE67D33
   X[ 0] = FA8EC1869CD9166FD9293729F9181919
       R = 9F955DCEDAE67D330C8F81FA903B27F8
       I = F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Final state:
   C[ 0] = DB9F4B85E34D92170AA44EC1CC9330F8CD3158F47D7083D15F45510C172C06B9
   C[ 1] = F5689A70D147702D
   X[ 0] = FA8EC1869CD9166FD9293729F9181919
       R = A84886CDAE5BDC293461C42664BF0320
   CipherText:
   Tag:             A84886CDAE5BDC293461C42664BF0320
```

Listing 13: Detailed test vector: m=0, a=1, s=0

```
Encrypting 0 bytes message with 1 bytes of associated data and 0 bytes of static data
```

```
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: AA
   F/G entry 0 (F with DS): padded=0, domain=1, finalize=0
   C[ 0] = 000102030405060708090A0B0C0D0E0F10FEFEFE8AF2F2F2FA171D872CB43FDF
   C[ 1] = 90C1F2A364656667
   X[ 0] = FA8EC1869CD9166FD9293729F9181919
       R = 00000000000000000000000000000000
       I = F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Padded A. data:  AA010000000000000000000000000000
   F/G entry 1 (F with DS): padded=1, domain=2, finalize=1
   C[ 0] = 5773FFA1ADDAD8BA6F841F9AA462BE9892CAFD3592A1955746C86F43D84F4DB4
   C[ 1] = C166D666D8E5D34B
   X[ 0] = FA8EC1869CD9166FD9293729F9181919
       R = F8CFEAFDB6712D548C7059F16D20CFC7
       I = AA010000000000000000000000000000
   Final state:
   C[ 0] = 13F24D5E35F1ED5071101BF2EDAC7C1D456064549711E9DB74B15747E84B8674
   C[ 1] = D176521C572967AD
   X[ 0] = FA8EC1869CD9166FD9293729F9181919
       R = BACC65F77A62EF1F22A0AC0B67B31E58
   CipherText:
   Tag:             BACC65F77A62EF1F22A0AC0B67B31E58
```

Listing 14: Detailed test vector: m=1, a=0, s=0

```
Encrypting 1 bytes message with 0 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Message:         DD
   F/G entry 0 (F with DS): padded=0, domain=1, finalize=0
   C[ 0] = 000102030405060708090A0B0C0D0E0F10FEFEFE8AF2F2F2FA171D872CB43FDF
   C[ 1] = 90C1F2A364656667
   X[ 0] = FA8EC1869CD9166FD9293729F9181919
       R = 00000000000000000000000000000000
       I = F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Padded Message:  DD010000000000000000000000000000
   F/G entry 1 (F with DS): padded=1, domain=3, finalize=1
   C[ 0] = 5773FFA1ADDAD8BA6F841F9AA462BE9892CAFD3592A1955746C86F43D84F4DB4
   C[ 1] = C166D666D8E5D34B
   X[ 0] = FA8EC1869CD9166FD9293729F9181919
       R = F8CFEAFDB6712D548C7059F16D20CFC7
       I = DD010000000000000000000000000000
   Final state:
   C[ 0] = 03AFECC889C6E881BB49FCE0FDACFE22B8ABC759EECA72377CC472CF06C5F78A
   C[ 1] = 4C98F5E2C612BE1F
   X[ 0] = FA8EC1869CD9166FD9293729F9181919
       R = 8A2DCEBC101D4FDDF788D17B73583B9F
   CipherText:      25
   Tag:             8A2DCEBC101D4FDDF788D17B73583B9F
```

Listing 15: Detailed test vector: m=1, a=1, s=0

```
Encrypting 1 bytes message with 1 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: AA
   Message:         DD
   F/G entry 0 (F with DS): padded=0, domain=1, finalize=0
   C[ 0] = 000102030405060708090A0B0C0D0E0F10FEFEFE8AF2F2F2FA171D872CB43FDF
   C[ 1] = 90C1F2A364656667
   X[ 0] = FA8EC1869CD9166FD9293729F9181919
       R = 00000000000000000000000000000000
       I = F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Padded A. data:  AA010000000000000000000000000000
   F/G entry 1 (F with DS): padded=1, domain=2, finalize=0
   C[ 0] = 5773FFA1ADDAD8BA6F841F9AA462BE9892CAFD3592A1955746C86F43D84F4DB4
   C[ 1] = C166D666D8E5D34B
   X[ 0] = FA8EC1869CD9166FD9293729F9181919
       R = F8CFEAFDB6712D548C7059F16D20CFC7
       I = AA010000000000000000000000000000
   Padded Message:  DD010000000000000000000000000000
   F/G entry 2 (F with DS): padded=1, domain=3, finalize=1
   C[ 0] = A300E3B16DBDAEC751F46DF6A5EDFE2D2FEC66E8C0B97BA14A84FDAED38B4493
   C[ 1] = 8C62730E8742911B
   X[ 0] = FA8EC1869CD9166FD9293729F9181919
       R = B04FEC7B440242E0AB89E60B290A217A
       I = DD010000000000000000000000000000
   Final state:
   C[ 0] = 759E7E7F38CBBA101FB2DD2B763300588C2846D1EFAB003D7BE82E3AA8819135
   C[ 1] = FD09291937AAEDF5
   X[ 0] = FA8EC1869CD9166FD9293729F9181919
       R = 08652BB8D084B1052790340A93E842C6
   CipherText:      6D
   Tag:             08652BB8D084B1052790340A93E842C6
```

Listing 16: Test vectors

```
Encrypting 0 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   CipherText:
   Tag:             FDA739118F1450B4B210AC07662A5394
Encrypting 1 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         00
   Padded Message:  00010000000000000000000000000000
   CipherText:      4F
   Tag:             74216DD16930300E660DDC70EF16434F
Encrypting 2 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         0001
   Padded Message:  00010100000000000000000000000000
   CipherText:      4F3E
   Tag:             BA14E80D0CE3C203F0D9268280B7843A
Encrypting 3 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102
   Padded Message:  00010201000000000000000000000000
   CipherText:      4F3E61
   Tag:             FFD51488453AEE3C5ED5294676E92620
Encrypting 4 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         00010203
   Padded Message:  00010203010000000000000000000000
   CipherText:      4F3E61DC
   Tag:             35A74C18080DA8B5C57946F1D6631E83
Encrypting 5 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         0001020304
   Padded Message:  00010203040100000000000000000000
   CipherText:      4F3E61DC0E
   Tag:             5CCDCE16A89C85505953EB6041284633
Encrypting 6 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405
   Padded Message:  00010203040501000000000000000000
   CipherText:      4F3E61DC0E33
   Tag:             85175E1FC21E5391B93D1FE6249BAF2A
Encrypting 7 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         00010203040506
   Padded Message:  00010203040506010000000000000000
   CipherText:      4F3E61DC0E3383
   Tag:             1F3F7EA55CD22A42D3C2B0CB6E61B228
Encrypting 8 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         0001020304050607
   Padded Message:  00010203040506070100000000000000
   CipherText:      4F3E61DC0E3383A6
   Tag:             4261A131EF45C68DFACA86C4F6895EF9
Encrypting 9 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708
   Padded Message:  00010203040506070801000000000000
   CipherText:      4F3E61DC0E3383A69D
   Tag:             CC124B5CF775070DD5925C2185A95BF3
Encrypting 10 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         00010203040506070809
   Padded Message:  00010203040506070809010000000000
   CipherText:      4F3E61DC0E3383A69D7B
   Tag:             9B6389AA9886129C303B5535CFEFFF42
Encrypting 11 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A
   Padded Message:  000102030405060708090A0100000000
   CipherText:      4F3E61DC0E3383A69D7BE9
   Tag:             8925A500CF57F0473BFA447850A7B47F
Encrypting 12 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B
   Padded Message:  000102030405060708090A0B01000000
   CipherText:      4F3E61DC0E3383A69D7BE92A
   Tag:             C93A6F3DBEF254B38ABBFF60DC3C6555
Encrypting 13 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C
   Padded Message:  000102030405060708090A0B0C010000
   CipherText:      4F3E61DC0E3383A69D7BE92AF8
   Tag:             3623AE02E8621B03738AFB3E7FD15571
Encrypting 14 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D
   Padded Message:  000102030405060708090A0B0C0D0100
   CipherText:      4F3E61DC0E3383A69D7BE92AF809
   Tag:             B9A20C19CBCB2B47B2495E93BB9475E4
Encrypting 15 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E
   Padded Message:  000102030405060708090A0B0C0D0E01
   CipherText:      4F3E61DC0E3383A69D7BE92AF80995
   Tag:             065E5BC539CFD35A9E7C24B8C39365FE
Encrypting 16 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F
   CipherText:      4F3E61DC0E3383A69D7BE92AF8099593
   Tag:             A659B375A17A71394D7AAC067DF0FBDA
Encrypting 17 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F10
   Padded Message:  000102030405060708090A0B0C0D0E0F1001000000000000000000000000000000
   CipherText:      4F3E61DC0E3383A69D7BE92AF80995932D
   Tag:             1E5D065A1C9E87A56341C035A4259266
Encrypting 18 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F1011
   Padded Message:  000102030405060708090A0B0C0D0E0F10110100000000000000000000000000
   CipherText:      4F3E61DC0E3383A69D7BE92AF80995932DA0
   Tag:             6768027504767A178C31E17F233AF1B7
Encrypting 19 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F101112
   Padded Message:  000102030405060708090A0B0C0D0E0F10111201000000000000000000000000
   CipherText:      4F3E61DC0E3383A69D7BE92AF80995932DA0CD
   Tag:             E41118900222C061161E4B9593575E17
Encrypting 20 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F10111213
   Padded Message:  000102030405060708090A0B0C0D0E0F10111213010000000000000000000000
   CipherText:      4F3E61DC0E3383A69D7BE92AF80995932DA0CD01
   Tag:             7196C8354C8CC071E3BC6BD6DA73A20F
Encrypting 21 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F1011121314
   Padded Message:  000102030405060708090A0B0C0D0E0F10111213140100000000000000000000
   CipherText:      4F3E61DC0E3383A69D7BE92AF80995932DA0CD0110
   Tag:             7F410BD854140B74BD8C12E0A7250A3F
Encrypting 22 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F101112131415
   Padded Message:  000102030405060708090A0B0C0D0E0F10111213141501000000000000000000
   CipherText:      4F3E61DC0E3383A69D7BE92AF80995932DA0CD0110B3
   Tag:             8B04A04AF742E50A25BBEA5C72CA51D4
Encrypting 23 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F10111213141516
   Padded Message:  000102030405060708090A0B0C0D0E0F10111213141516010000000000000000
   CipherText:      4F3E61DC0E3383A69D7BE92AF80995932DA0CD0110B323
   Tag:             0E5861F508E87264C11B2E2E129E04F4
Encrypting 24 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:             000102030405060708090A0B0C0D0E0F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F1011121314151617
   Padded Message:  000102030405060708090A0B0C0D0E0F10111213141516170100000000000000
   CipherText:      4F3E61DC0E3383A69D7BE92AF80995932DA0CD0110B323CA
   Tag:              19D18B2548C2C9436179606DE64C499E
Encrypting 25 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:              000102030405060708090A0B0C0D0E0F
   Nonce:            F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data:  00000000000000000000000000000000
   Message:          000102030405060708090A0B0C0D0E0F101112131415161718
   Padded Message:   000102030405060708090A0B0C0D0E0F1011121314151617180100000000000000
   CipherText:       4F3E61DC0E3383A69D7BE92AF80995932DA0CD0110B323CA80
   Tag:              119BDB05DD455F29BCE4DAA51EAA5B22
Encrypting 26 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:              000102030405060708090A0B0C0D0E0F
   Nonce:            F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data:  00000000000000000000000000000000
   Message:          000102030405060708090A0B0C0D0E0F10111213141516171819
   Padded Message:   000102030405060708090A0B0C0D0E0F1011121314151617181901000000000000
   CipherText:       4F3E61DC0E3383A69D7BE92AF80995932DA0CD0110B323CA80EA
   Tag:              39DED9E28619FC5B577E87CF4C645C61
Encrypting 27 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:              000102030405060708090A0B0C0D0E0F
   Nonce:            F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data:  00000000000000000000000000000000
   Message:          000102030405060708090A0B0C0D0E0F101112131415161718191A
   Padded Message:   000102030405060708090A0B0C0D0E0F101112131415161718191A0100000000
   CipherText:       4F3E61DC0E3383A69D7BE92AF80995932DA0CD0110B323CA80EA30
   Tag:              35A7DB76A1C1160C40012585E120EA12
Encrypting 28 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:              000102030405060708090A0B0C0D0E0F
   Nonce:            F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data:  00000000000000000000000000000000
   Message:          000102030405060708090A0B0C0D0E0F101112131415161718191A1B
   Padded Message:   000102030405060708090A0B0C0D0E0F101112131415161718191A1B01000000
   CipherText:       4F3E61DC0E3383A69D7BE92AF80995932DA0CD0110B323CA80EA3021
   Tag:              B52166C607AC92BADB259A6745957D4F
Encrypting 29 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:              000102030405060708090A0B0C0D0E0F
   Nonce:            F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data:  00000000000000000000000000000000
   Message:          000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C
   Padded Message:   000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C010000
   CipherText:       4F3E61DC0E3383A69D7BE92AF80995932DA0CD0110B323CA80EA302160
   Tag:              3EC3CD61B1685883F30FFC6FFB2C3CCB
Encrypting 30 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:              000102030405060708090A0B0C0D0E0F
   Nonce:            F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data:  00000000000000000000000000000000
   Message:          000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D
   Padded Message:   000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D0100
   CipherText:       4F3E61DC0E3383A69D7BE92AF80995932DA0CD0110B323CA80EA3021603D
   Tag:              A020F44CFFB344C5E8EB0381B7A3A592
Encrypting 31 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:              000102030405060708090A0B0C0D0E0F
   Nonce:            F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data:  00000000000000000000000000000000
   Message:          000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E
   Padded Message:   000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E01
   CipherText:       4F3E61DC0E3383A69D7BE92AF80995932DA0CD0110B323CA80EA3021603D00
   Tag:              8DA9500BA2EE4E809C1DE1977D3CCC77
Encrypting 32 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 16 bytes
   Key:              000102030405060708090A0B0C0D0E0F
   Nonce:            F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data:  00000000000000000000000000000000
   Message:          000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   CipherText:       4F3E61DC0E3383A69D7BE92AF80995932DA0CD0110B323CA80EA3021603D00CD
   Tag:              BCDBD3BF6977DC808F47F0E40B5D9DED
```

Listing 17: Iterative test vector

```
Encrypting null message 100 times with tag feedback as associated data
   CipherText: A9D9DF805B3E65452220A4B72B732F55
```

14.5 DryGascon128 hash test vectors

Listing 18: Detailed test vector

```
Hashing 8 bytes message: 0001020304050607
   Padded Message:  00010203040506070100000000000000
   F/G entry 0 (F with DS): padded=1, domain=2, finalize=1
   C[ 0] = 243F6A8885A308D313198A2E03707344243F6A8885A308D313198A2E03707344
   C[ 1] = 243F6A8885A308D3
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = 00000000000000000000000000000000
       I = 00010203040506070100000000000000
ds = b
MixPhaseRound entry  0 243F6A8885A308D313198A2E03707344243F6A8885A308D313198A2E03707344243F
     ↪ 6A8885A308D3
CoreRound entry      0 803652AA85A308D3B710B20C03707344803652AA85A308D3B710B20C037073440DA0
     ↪ 5B5885A308D3
MixPhaseRound entry  1 1D20A609812447D1CDB40F3FFD61D844EB19F0F39A065640DE4C9BB3C92CAC747D21
     ↪ 3259BD45B5DC
CoreRound entry      1 B9299E2B812447D169BD371DFD61D8444F10C8D19A065640D662612BC92CAC74D928
     ↪ 0A7BBD45B5DC
MixPhaseRound entry  2 2431255722DCF9237E328835559412E4C8C80B1C89EF89CC50114568A502E45EE4BE
     ↪ 2A6F54B85E9B
CoreRound entry      2 80381D7522DCF923DA3BB017559412E42486679589EF89CCF4187D4AA502E45E40B7
     ↪ 124D54B85E9B
MixPhaseRound entry  3 26B4256EFBC612BF3C82AFE81CBA89DE3C8551010589A4802A22CA0D075561ECFD3A
     ↪ 10E910DB08C3
CoreRound entry      3 82BD1D4CFBC612BF988B97CA1CBA89DE151A60D10589A4808E2BF22F075561EC5933
     ↪ 28CB10DB08C3
MixPhaseRound entry  4 47844153D0F43FC87C3BE934E69D2D0FFF375AE2413181E4323B70461FDE27839266
     ↪ 78584097ED55
CoreRound entry      4 6E1B7083D0F43FC855A4D8E4E69D2D0F5B3E62C0413181E4963248641FDE27839A48
     ↪ 82C04097ED55
MixPhaseRound entry  5 D2410B4ADBF4C7275F801B3737F8602DB09A2430DFAE6BE5CDB56F7F836BB83CA6F4
     ↪ BE179765B879
CoreRound entry      5 FBDE3A9ADBF4C727FB89231537F8602D14931C12DFAE6BE521FB03F6836BB83C8F6B
     ↪ 8FC79765B879
MixPhaseRound entry  6 B1BD8F42BF91A0E938A2EEFD13C9677AC0D49150144A8C646F06CA8E948DEC62F9C7
     ↪ 8C34E5F97A31
CoreRound entry      6 15B4B760BF91A0E99CABD6DF13C9677AE94BA080144A8C64CB0FF2AC948DEC625
     ↪ DCEB416E5F97A31
MixPhaseRound entry  7 419CCA3BF6448A43C6C060A2252A21E11A68DA94BD224097CC4048C8C55D25AB951
     ↪ DA4196BD6DD6D
CoreRound entry      7 E595F219F6448A4362C95880252A21E1BE61E2B6BD224097684970EAC55D25AB3114
     ↪ 9C3B6BD6DD6D
MixPhaseRound entry  8 F94722529A8232D1F5D72099971436B4F71E964E051A8A18B8CD8F0667BBE06C15B4
     ↪ 5050A482FF38
CoreRound entry      8 5D4E1A709A8232D151DE18BB971436B45317AE6C051A8A181CC4B72467BBE06CB1BD
     ↪ 6872A482FF38
MixPhaseRound entry  9 EF9799083DE230C09AEC012C4D7D50DBB4DF1DB606E837D60A781B95381C9F35E91E
     ↪ 89D01B7E045C
CoreRound entry      9 4B9EA12A3DE230C03EE5390E4D7D50DB10D6259406E837D6AE7123B7381C9F354D17
     ↪ B1F21B7E045C
MixPhaseRound entry 10 ACD382C8883FB088DC97B86B923AECD89BBB459B91EC725F2B6FE62E294300D98DD6
     ↪ CDB6EA11D62E
CoreRound entry     10 08DABAEA883FB088789E8049923AECD83FB27DB991EC725F8F66DE0C294300D929
     ↪ DFF594EA11D62E
MixPhaseRound entry 11 684DFF56DB76FF38C46FE9124706C22C1B1A70D22D7BF0DA08896C0AEDCF2EB72E02
     ↪ 7F5E99CAE1A2
CoreRound entry     11 CC44C774DB76FF386066D1304706C22CBF1348F02D7BF0DAAC805428EDCF2EB78A0B
     ↪ 477C99CAE1A2
MixPhaseRound entry 12 85D648742114A7220B8489F0473CEECD6DB3052F3CD29F9D1D02272C9C454905AC2
     ↪ DD368D13E3CA7
```

```
CoreRound entry     12 21DF70562114A722AF8DB1D2473CEECDC9BA3D0D3CD29F9DB90B1F0E9C4549054063
    ↪ BFE1D13E3CA7
MixPhaseRound entry 13 8A7F7448152D17A0D6B2AC3C48A90F17E9F653F4A9BAE44CBDA88D4FFA9B8272673E
    ↪ 45F47CC71C86
   After mix phase:
   C[ 0] = 82518ED0152D17A072BB941E48A90F174DFF6BD6A9BAE44C19A1B56DFA9B8272
   C[ 1] = C3377DD67CC71C86
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = 00000000000000000000000000000000
   CoreRound entry 0:
   C[ 0] = 82518ED0152D17A072BB941E48A90F174DFF6BD6A9BAE44C19A1B56DFA9B8272
   C[ 1] = C3377DD67CC71C86
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = 00000000000000000000000000000000
   Accumulate entry 0:
   C[ 0] = A600807683DD525766818A1DFBE3CF385610E9D1E0EBAD36E36B07ECCC559690
   C[ 1] = 64CE81BF21F37ECA
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = 00000000000000000000000000000000
accumulate in[0] = A600807683DD525766818A1DFBE3CF38
accumulate in[1] = E0EBAD36E36B07ECCC5596905610E9D1
   CoreRound entry 1:
   C[ 0] = A600807683DD525766818A1DFBE3CF385610E9D1E0EBAD36E36B07ECCC559690
   C[ 1] = 64CE81BF21F37ECA
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = 46EB2D4060B655BBAAD41C8DADF326E9
   Accumulate entry 1:
   C[ 0] = E1D0DD2E955E6F59FE91123FD95C98A8AA22EADF688A851C22856BFAD1C249C8
   C[ 1] = 25FC03025CEA20F8
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = 46EB2D4060B655BBAAD41C8DADF326E9
accumulate in[0] = E1D0DD2E955E6F59FE91123FD95C98A8
accumulate in[1] = 688A851C22856BFAD1C249C8AA22EADF
   CoreRound entry 2:
   C[ 0] = E1D0DD2E955E6F59FE91123FD95C98A8AA22EADF688A851C22856BFAD1C249C8
   C[ 1] = 25FC03025CEA20F8
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = CFB17572D76D51188587477ADE8D549E
   Accumulate entry 2:
   C[ 0] = 2922FCA600B7206AFF6B0D6CF756938AB1D325E754DB393B9A12383AC83F969B
   C[ 1] = 981AF4C2913D4670
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = CFB17572D76D51188587477ADE8D549E
accumulate in[0] = 2922FCA600B7206AFF6B0D6CF756938A
accumulate in[1] = 54DB393B9A12383AC83F969BB1D325E7
   CoreRound entry 3:
   C[ 0] = 2922FCA600B7206AFF6B0D6CF756938AB1D325E754DB393B9A12383AC83F969B
   C[ 1] = 981AF4C2913D4670
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = B248B0EF4DC84948B2D3DC8D9808E2F3
   Accumulate entry 3:
   C[ 0] = D0BB354FEBADF3E980CA3BACEE3047F1D15484CCBD7BDED1096C20BC6401B2DB
   C[ 1] = 87AE15893F342EB1
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = B248B0EF4DC84948B2D3DC8D9808E2F3
accumulate in[0] = D0BB354FEBADF3E980CA3BACEE3047F1
accumulate in[1] = BD7BDED1096C20BC6401B2DBD15484CC
   CoreRound entry 4:
   C[ 0] = D0BB354FEBADF3E980CA3BACEE3047F1D15484CCBD7BDED1096C20BC6401B2DB
   C[ 1] = 87AE15893F342EB1
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = DF885B71AF099A1D561855FAA76C21CE
   Accumulate entry 4:
```

```
   C[ 0] = F78A3165596A2FC254756B8B3E5B5E0358DFF5F26F21AD4FE450BD6F2F04B210
   C[ 1] = 722BDDC448E91BCB
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = DF885B71AF099A1D561855FAA76C21CE
accumulate in[0] = F78A3165596A2FC254756B8B3E5B5E03
accumulate in[1] = 6F21AD4FE450BD6F2F04B21058DFF5F2
   CoreRound entry 5:
   C[ 0] = F78A3165596A2FC254756B8B3E5B5E0358DFF5F26F21AD4FE450BD6F2F04B210
   C[ 1] = 722BDDC448E91BCB
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = 4723C75B123308B02D698C61C1E88A3F
   Accumulate entry 5:
   C[ 0] = 8DFF7C6F00618CFEBB74DA46E1EA2791623DE0FE517E15B44177CF4C2F06AD4C
   C[ 1] = 7933A3A82B1AA188
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = 4723C75B123308B02D698C61C1E88A3F
accumulate in[0] = 8DFF7C6F00618CFEBB74DA46E1EA2791
accumulate in[1] = 517E15B44177CF4C2F06AD4C623DE0FE
   CoreRound entry 6:
   C[ 0] = 8DFF7C6F00618CFEBB74DA46E1EA2791623DE0FE517E15B44177CF4C2F06AD4C
   C[ 1] = 7933A3A82B1AA188
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = 9BA2AE8053254B02B91BFB6B423F4D50
   Accumulate entry 6:
   C[ 0] = 344616A2DC4B29B8E1F75F43B138AC801673D4A1620666C2AC3D2771A2BD4804
   C[ 1] = ED998EABD71F8492
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = 9BA2AE8053254B02B91BFB6B423F4D50
accumulate in[0] = 344616A2DC4B29B8E1F75F43B138AC80
accumulate in[1] = 620666C2AC3D2771A2BD48041673D4A1
   F/G entry 1 (G):
   C[ 0] = 344616A2DC4B29B8E1F75F43B138AC801673D4A1620666C2AC3D2771A2BD4804
   C[ 1] = ED998EABD71F8492
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = CDE2DEE0235345CBFA51EC2CE5743571
   CoreRound entry 0:
   C[ 0] = 344616A2DC4B29B8E1F75F43B138AC801673D4A1620666C2AC3D2771A2BD4804
   C[ 1] = ED998EABD71F8492
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = 00000000000000000000000000000000
   Accumulate entry 0:
   C[ 0] = 8B8D3119955D9EF399615E287DD5FE67B7A7B28ACA86B9C1B4E609A6B6F1CAB2
   C[ 1] = B70C71D2EF33EEC7
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = 00000000000000000000000000000000
accumulate in[0] = 8B8D3119955D9EF399615E287DD5FE67
accumulate in[1] = CA86B9C1B4E609A6B6F1CAB2B7A7B28A
   CoreRound entry 1:
   C[ 0] = 8B8D3119955D9EF399615E287DD5FE67B7A7B28ACA86B9C1B4E609A6B6F1CAB2
   C[ 1] = B70C71D2EF33EEC7
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = 410B88D821BB97552F90949ACA724CED
   Accumulate entry 1:
   C[ 0] = 7D9627F86B554E02734C7981432C49EB14F9537058A3BEB9827938939D8A19A1
   C[ 1] = AFE51BBA9EF8D9B6
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = 410B88D821BB97552F90949ACA724CED
accumulate in[0] = 7D9627F86B554E02734C7981432C49EB
accumulate in[1] = 58A3BEB9827938939D8A19A114F95370
   CoreRound entry 2:
   C[ 0] = 7D9627F86B554E02734C7981432C49EB14F9537058A3BEB9827938939D8A19A1
   C[ 1] = AFE51BBA9EF8D9B6
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
```

```
       R = 643E1199C897E1C4C156F4BA9DA75676
   Accumulate entry 2:
   C[ 0] = 856F4AA55D29BC1DA63813F0A9D38A8C4D17C479C967DAFF6E7B083826311423
   C[ 1] = CA54F3AAC94E2E11
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = 643E1199C897E1C4C156F4BA9DA75676
accumulate in[0] = 856F4AA55D29BC1DA63813F0A9D38A8C
accumulate in[1] = C967DAFF6E7B0838263114234D17C479
   CoreRound entry 3:
   C[ 0] = 856F4AA55D29BC1DA63813F0A9D38A8C4D17C479C967DAFF6E7B083826311423
   C[ 1] = CA54F3AAC94E2E11
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = 283681C3FBC555E1415FF36979631883
   Accumulate entry 3:
   C[ 0] = 0579EF20D4BA62AD84E9C0C15D46AFED8BABC596D74F78B542CC91C8E3AB16F0
   C[ 1] = D3344F261516E1DA
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = 283681C3FBC555E1415FF36979631883
accumulate in[0] = 0579EF20D4BA62AD84E9C0C15D46AFED
accumulate in[1] = D74F78B542CC91C8E3AB16F08BABC596
   CoreRound entry 4:
   C[ 0] = 0579EF20D4BA62AD84E9C0C15D46AFED8BABC596D74F78B542CC91C8E3AB16F0
   C[ 1] = D3344F261516E1DA
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = FA0016566DB3A684261D2558AF8E72F8
   Accumulate entry 4:
   C[ 0] = BD6F29364D80A070EB04415562F0B9110EFEAB92C7982A5F414EC2B0924EC587
   C[ 1] = 95354C9E4EB0DC0F
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = FA0016566DB3A684261D2558AF8E72F8
accumulate in[0] = BD6F29364D80A070EB04415562F0B911
accumulate in[1] = C7982A5F414EC2B0924EC5870EFEAB92
   CoreRound entry 5:
   C[ 0] = BD6F29364D80A070EB04415562F0B9110EFEAB92C7982A5F414EC2B0924EC587
   C[ 1] = 95354C9E4EB0DC0F
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = 80F7153F617DC4445F57A18AC380607B
   Accumulate entry 5:
   C[ 0] = 3B69E030D5B1D86EE9D2991D425B4047B835AEE9E13956F50683396E096067AD
   C[ 1] = 780F6C0DC411CB7C
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = 80F7153F617DC4445F57A18AC380607B
accumulate in[0] = 3B69E030D5B1D86EE9D2991D425B4047
accumulate in[1] = E13956F50683396E096067ADB835AEE9
   CoreRound entry 6:
   C[ 0] = 3B69E030D5B1D86EE9D2991D425B4047B835AEE9E13956F50683396E096067AD
   C[ 1] = 780F6C0DC411CB7C
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = 5AA7A3FAB24F2544BFE55F3A39EE8ED5
   Accumulate entry 6:
   C[ 0] = FFD03BDF0ECAFA7A00679D18A7A52CE7501ABC162BB78B1B7EF0B13DE026C6E3
   C[ 1] = B3FF14D00C927030
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = 5AA7A3FAB24F2544BFE55F3A39EE8ED5
accumulate in[0] = FFD03BDF0ECAFA7A00679D18A7A52CE7
accumulate in[1] = 2BB78B1B7EF0B13DE026C6E3501ABC16
   Final state:
   C[ 0] = FFD03BDF0ECAFA7A00679D18A7A52CE7501ABC162BB78B1B7EF0B13DE026C6E3
   C[ 1] = B3FF14D00C927030
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = 8EC0133EC2756E035FA404C1CE511E24
   Digest: CDE2DEE0235345CBFA51EC2CE57435718EC0133EC2756E035FA404C1CE511E24
```

Listing 19: Less detailed test vector

```
Hashing 8 bytes message: 0001020304050607
   Padded Message:  00010203040506070100000000000000
   F/G entry 0 (F with DS): padded=1, domain=2, finalize=1
   C[ 0] = 243F6A8885A308D313198A2E03707344243F6A8885A308D313198A2E03707344
   C[ 1] = 243F6A8885A308D3
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = 00000000000000000000000000000000
       I = 00010203040506070100000000000000
   F/G entry 1 (G):
   C[ 0] = 344616A2DC4B29B8E1F75F43B138AC801673D4A1620666C2AC3D2771A2BD4804
   C[ 1] = ED998EABD71F8492
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = CDE2DEE0235345CBFA51EC2CE5743571
   Final state:
   C[ 0] = FFD03BDF0ECAFA7A00679D18A7A52CE7501ABC162BB78B1B7EF0B13DE026C6E3
   C[ 1] = B3FF14D00C927030
   X[ 0] = A4093822299F31D0082EFA98EC4E6C89
       R = 8EC0133EC2756E035FA404C1CE511E24
   Digest: CDE2DEE0235345CBFA51EC2CE57435718EC0133EC2756E035FA404C1CE511E24
```

Listing 20: Test vectors

```
Hashing 0 bytes message:
   Padded Message:  01000000000000000000000000000000
   Digest: 1EDC77386E20A37C721D6E77ADABB9C4830F199F5ED25284A13C1D84B9FC257A
Hashing 1 bytes message: 00
   Padded Message:  00010000000000000000000000000000
   Digest: 1BEC89506E75D725BF93BCCFDD6EC81DF05CA281CF5201E3EE0865A7063763EE
Hashing 2 bytes message: 0001
   Padded Message:  00010100000000000000000000000000
   Digest: 0FE4ED67EA1FF705E94E6D8AF07197728C1FC2D7D5ACCECB8D08CF39AE4D208D
Hashing 3 bytes message: 000102
   Padded Message:  00010201000000000000000000000000
   Digest: 8E3E664C48CC6972CB7DEE3B6769F1AE86D0E248473E11C1EE2C12C13CDECB0D
Hashing 4 bytes message: 00010203
   Padded Message:  00010203010000000000000000000000
   Digest: 591A2858B4E3B0B99BC116E18B44B55D711F2A8E83FAE677CED46DB03E031B73
Hashing 5 bytes message: 0001020304
   Padded Message:  00010203040100000000000000000000
   Digest: 8C16038F28FEDF8B547B481F259851C2E72AB9D6679B8E1EF9756FF3172F43B9
Hashing 6 bytes message: 000102030405
   Padded Message:  00010203040501000000000000000000
   Digest: 4246D676478A3986BCAA03576F97C2906B2A230F6F12BC3025C868B5F43F78C3
Hashing 7 bytes message: 00010203040506
   Padded Message:  00010203040506010000000000000000
   Digest: 12BCF926FCC2283DB1CD5788AF34AD34A0CF562DCA24D977AC602D05F6FEF015
Hashing 8 bytes message: 0001020304050607
   Padded Message:  00010203040506070100000000000000
   Digest: CDE2DEE0235345CBFA51EC2CE57435718EC0133EC2756E035FA404C1CE511E24
Hashing 9 bytes message: 000102030405060708
   Padded Message:  00010203040506070801000000000000
   Digest: 265B5B605B22F3369E71685813B842F830BAB72D61AADE051E3B8901D3BBA7DF
Hashing 10 bytes message: 00010203040506070809
   Padded Message:  00010203040506070809010000000000
   Digest: B41A3F96C2D965E5D0C617300305F36ADFC02C267DFF679238956684CDD292FF
Hashing 11 bytes message: 000102030405060708090A
   Padded Message:  000102030405060708090A0100000000
   Digest: BDE27ECFF70883573139D7A231436DE245CA2BB1258CA20314A22351C06A375E
```

```
Hashing 12 bytes message: 000102030405060708090A0B
   Padded Message:  000102030405060708090A0B01000000
   Digest: 8E5CD148040C457627F0018AE33A3FEBF75A2BB54C1565983E62195454662710
Hashing 13 bytes message: 000102030405060708090A0B0C
   Padded Message:  000102030405060708090A0B0C010000
   Digest: 2B9943184961C1DE2A54449E8D6C98076F4C1FAF99547A53D8E1809EDD1912D5
Hashing 14 bytes message: 000102030405060708090A0B0C0D
   Padded Message:  000102030405060708090A0B0C0D0100
   Digest: 19197D41382063708ED78AAE09517E4295CD855FB1FAB1D3006F5A0600503198
Hashing 15 bytes message: 000102030405060708090A0B0C0D0E
   Padded Message:  000102030405060708090A0B0C0D0E01
   Digest: 96C2D304E40755091BA9DBBCDC8B1F5D7A59D949A4AA94490B57A232B7692ACE
Hashing 16 bytes message: 000102030405060708090A0B0C0D0E0F
   Digest: 572821D80D943E153CBB8C4556C3AD8CF20D77EDAD7998E8CD46F590D8D13EEB
Hashing 17 bytes message: 000102030405060708090A0B0C0D0E0F10
   Padded Message:  000102030405060708090A0B0C0D0E0F1001000000000000000000000000000000
   Digest: 20CDB78974D692100612978096CCFE82E39F15969F493FAD8FA870F93B7252EA
Hashing 18 bytes message: 000102030405060708090A0B0C0D0E0F1011
   Padded Message:  000102030405060708090A0B0C0D0E0F1011010000000000000000000000000000
   Digest: AFCE36F85AF503EFEC0AB342425706351A9794D2FEE7D5D6BEC9926866E7FCE8
Hashing 19 bytes message: 000102030405060708090A0B0C0D0E0F101112
   Padded Message:  000102030405060708090A0B0C0D0E0F1011120100000000000000000000000000
   Digest: 56865892C9136B9904ECA6E472E5F193F1AFF9912CBF3A7FB2721E2D36CAA282
Hashing 20 bytes message: 000102030405060708090A0B0C0D0E0F10111213
   Padded Message:  000102030405060708090A0B0C0D0E0F1011121301000000000000000000000000
   Digest: F874F392AE5A582D979B6C339A28C9876C846207135065CFF5001CB41FA7179B
Hashing 21 bytes message: 000102030405060708090A0B0C0D0E0F1011121314
   Padded Message:  000102030405060708090A0B0C0D0E0F1011121314010000000000000000000000
   Digest: 24651BB0300C1E3893EB80916E21BF9BFB4055F0B52743A3ED9C7569B6DC908B
Hashing 22 bytes message: 000102030405060708090A0B0C0D0E0F101112131415
   Padded Message:  000102030405060708090A0B0C0D0E0F1011121314150100000000000000000000
   Digest: 2CB786F73CA8548C894C5BFFEB1AEE5B600D17DE381C56539353EDF93DC6C39D
Hashing 23 bytes message: 000102030405060708090A0B0C0D0E0F10111213141516
   Padded Message:  000102030405060708090A0B0C0D0E0F1011121314151601000000000000000000
   Digest: F2ECA48277570B0CEAA2466241EDACA900E41B860DDC59FC7C83B709F00F23F6
Hashing 24 bytes message: 000102030405060708090A0B0C0D0E0F1011121314151617
   Padded Message:  000102030405060708090A0B0C0D0E0F1011121314151617010000000000000000
   Digest: E882DC9BF99A4A0639C2DF9CA21A7BCE12B37D1AB0E77515E715BAD5DBC3720F
Hashing 25 bytes message: 000102030405060708090A0B0C0D0E0F101112131415161718
   Padded Message:  000102030405060708090A0B0C0D0E0F1011121314151617180100000000000000
   Digest: 755EB69D89176498EF0C364E2DBE330E15E5E58A728515FC119E15DEB8396F5E
Hashing 26 bytes message: 000102030405060708090A0B0C0D0E0F10111213141516171819
   Padded Message:  000102030405060708090A0B0C0D0E0F1011121314151617181901000000000000
   Digest: 586F946255B3C39B5BA2ECDEBADCC86173B30A348463FC70260332504D40C10A
Hashing 27 bytes message: 000102030405060708090A0B0C0D0E0F101112131415161718191A
   Padded Message:  000102030405060708090A0B0C0D0E0F101112131415161718191A0100000000
   Digest: 413996541B9E7C66696BF55F33E01176D3F5C60EE2037E6DB854B61B5987CDB9
Hashing 28 bytes message: 000102030405060708090A0B0C0D0E0F101112131415161718191A1B
   Padded Message:  000102030405060708090A0B0C0D0E0F101112131415161718191A1B01000000
   Digest: 067610CF9772EFBFC144067F696A37128808C2D31772A27D36E36172A6D2B48F
Hashing 29 bytes message: 000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C
   Padded Message:  000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C010000
   Digest: 8ADD16D3F3BFF80BF923B9730D754DB63D800A1D318A3CC9A545D9D4DFC14A3D
Hashing 30 bytes message: 000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D
   Padded Message:  000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D0100
   Digest: 76ECF7939AC17E42EE5650E617E58842FAC759C70B5A2463ED1B9622F8608A46
Hashing 31 bytes message: 000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E
   Padded Message:  000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E01
   Digest: 7F905617D5E71D6DC4DA89CC5E8AEB3458D7D08A1148FFE56A5347A30BEE4424
Hashing 32 bytes message: 000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Digest: 6625F35944752F63DCB7403F80D9A15A2DB8D517428239EC6A9C4BAC28CA8C44
```

Listing 21: Iterative test vector

```
Hashing null message 100 times
   Digest: DC1217E41E6ACF596FE183979DB5409A6E3367F22358406DE8D2CE3EDFD36C40
```

14.6 DryGascon256 AEAD test vectors

Listing 22: Detailed test vector: m=0, a=0, s=0, full key profile

```
Encrypting 0 bytes message with 0 bytes of associated data and 0 bytes of static data
Key size: 88 bytes
   Key:              000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F2021222
    ↪ 32425262728292A2B2C2D2E2F303132333435363738393A3B3C3D3E3F404142434445464748494A4B4C
    ↪ 4D4E4F5051525354555657
   Nonce:            F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   F/G entry 0 (F with DS): padded=0, domain=1, finalize=1
   C[ 0] = 000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   C[ 1] = 202122232425262728292A2B2C2D2E2F303132333435363738393A3B3C3D3E3F
   C[ 2] = 4041424344454647
   X[ 0] = 48494A4B4C4D4E4F5051525354555657
       R = 00000000000000000000000000000000
       I = F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   F/G entry 1 (G):
   C[ 0] = CF6DC28F622A3773A8CBB69E4F1B53387A22C8CB1A82CD0EE2AA57AA13D2DE03
   C[ 1] = 45BF2A9575369E02DA9FA98FB9C519B9224114F6996F869068A468A96DEF36AB
   C[ 2] = BF6A921462AFC933
   X[ 0] = 48494A4B4C4D4E4F5051525354555657
       R = C94B4B87FFA0E3B1B3F31A6C3B2997B9
   Final state:
   C[ 0] = 9878CA3CE9C6DC6500A9E89D97121074134DB021F3A3C5441D59CC12045FF18E
   C[ 1] = 9E7E4A9AD0474811407BFD26365D35ED59218248E30BF96A3EDDD89894AB7821
   C[ 2] = 1E11C9B821AD549D
   X[ 0] = 48494A4B4C4D4E4F5051525354555657
       R = 3BDA620C3BBFCDF638224DFB1A62A750
   CipherText:
   Tag:              C94B4B87FFA0E3B1B3F31A6C3B2997B93BDA620C3BBFCDF638224DFB1A62A750
```

Listing 23: Detailed test vector: m=0, a=0, s=0, fast key profile

```
Encrypting 0 bytes message with 0 bytes of associated data and 0 bytes of static data
Key size: 48 bytes
   Key:              000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F2021222
    ↪ 32425262728292A2B2C2D2E2F
   Nonce:            F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   F/G entry 0 (F with DS): padded=0, domain=1, finalize=1
   C[ 0] = 000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   C[ 1] = 000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   C[ 2] = 0001020304050607
   X[ 0] = 202122232425262728292A2B2C2D2E2F
       R = 00000000000000000000000000000000
       I = F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   F/G entry 1 (G):
   C[ 0] = 52B58E62EC6C971198169461E4BA769E8FE8BFC2E437F6BA6B4AC7CDF3F54847
   C[ 1] = 994497D0FC817271E973627EF172151DE0585C58CC9FD72F5D1861A73A6AE28E
   C[ 2] = F42FCFD300D4808A
   X[ 0] = 202122232425262728292A2B2C2D2E2F
       R = 8B03010FAC69A86BDF0A9585E2C498DD
   Final state:
   C[ 0] = 33E28BCF3DAE3A654E260FDC902FFD7C5615D2AA33ACF34DF277079DE264F071
   C[ 1] = C12A5D71562F304BD1B28405125967586E4943EB735BAE80F3CA2611E0E3804E
   C[ 2] = 32EB0921C21F6F87
   X[ 0] = 202122232425262728292A2B2C2D2E2F
```

```
      R = 3B43779015622FE43123EA70FA658DEE
   CipherText:
   Tag:            8B03010FAC69A86BDF0A9585E2C498DD3B43779015622FE43123EA70FA658DEE
```

Listing 24: Detailed test vector: m=0, a=0, s=0

```
Encrypting 0 bytes message with 0 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:            000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:          F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   F/G entry 0 (F with DS): padded=0, domain=1, finalize=1
   C[ 0] = 000102030405060708090A0B0C0D0E0FFF03030367171717E700000000000070
   C[ 1] = 86767976F8F7F7F7F00080070000F000DEDEDEDEDBDBDBDB56617047021D140B
   C[ 2] = A139B22A53DF58D4
   X[ 0] = 0A8E01859CD916537B7B7B7B2B2B2B2B
       R = 00000000000000000000000000000000
       I = F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   F/G entry 1 (G):
   C[ 0] = DAEFF115C243B8FCF5C6AB3A9D3C78027413C9D9A414579F41210FB3F93DAB91
   C[ 1] = 00DBB026E4326D7565C75DD4DEBD8C209E992BD116458E0B5CA4350E4A307844
   C[ 2] = CA1343E2FCBFCF24
   X[ 0] = 0A8E01859CD916537B7B7B7B2B2B2B2B
       R = 964D44DD21CC8BE88FBC51B427885529
   Final state:
   C[ 0] = E424DDBD0B9492320155AF5248EA621D46EA400BD5B967C58612A65FCA369AED
   C[ 1] = 1C0D5D7D02B3F118F03B06087E14A6BD099D13EBB5F53CC8857E2AB159A60AE6
   C[ 2] = 49E9760CD04FD648
   X[ 0] = 0A8E01859CD916537B7B7B7B2B2B2B2B
       R = F0A6EF8125F76731FAF005884B0695CE
   CipherText:
   Tag:            964D44DD21CC8BE88FBC51B427885529F0A6EF8125F76731FAF005884B0695CE
```

Listing 25: Detailed test vector: m=0, a=0, s=1

```
Encrypting 0 bytes message with 0 bytes of associated data and 1 bytes of static data
Key size: 32 bytes
   Key:            000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:          F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Static data:    BB
   Padded S. data: BB010000000000000000000000000000
   F/G entry 0 (F with DS): padded=1, domain=2, finalize=1
   C[ 0] = 000102030405060708090A0B0C0D0E0FFF03030367171717E700000000000070
   C[ 1] = 86767976F8F7F7F7F00080070000F000DEDEDEDEDBDBDBDB56617047021D140B
   C[ 2] = A139B22A53DF58D4
   X[ 0] = 0A8E01859CD916537B7B7B7B2B2B2B2B
       R = 00000000000000000000000000000000
       I = BB010000000000000000000000000000
   F/G entry 1 (G):
   C[ 0] = 90A9C61C1C8AE0B5DEC8500DDC8D370499612CE392535FD20A5DB46435997D85
   C[ 1] = 13B605C6B2662E7FCBF7360F2873B74452C8323FF7DF889B4C4F51F04A39DB2C
   C[ 2] = 54BB314742778CB9
   X[ 0] = 0A8E01859CD916537B7B7B7B2B2B2B2B
       R = 3BF4433B3CE9695716BE4FADD308AB2D
   F/G entry 2 (G):
   C[ 0] = 0EB2849C1BCFF09C9A58066C46FAB37C4C2F457E6C885B6079B3E9CACD73E114
   C[ 1] = 5B093703F64BBD4E45F4E8D1BBF7855D097D33D118E3A1EBB63453949AD58268
   C[ 2] = 64F05FB0BA6D15C7
   X[ 0] = 0A8E01859CD916537B7B7B7B2B2B2B2B
       R = E4EAE7450AC019F3DDB2F56D99A7C841
   F/G entry 3 (G):
   C[ 0] = F75E6372A955BC31740C0F7C265A97AD7BD6DFB0852CC42819A13571B79C3B9F
   C[ 1] = 96330367EF765621D9F1CF50C0C29A54BAD4CA6280593D279C18FB0E33C549BF
   C[ 2] = 663A55D262FF6DE3
   X[ 0] = 0A8E01859CD916537B7B7B7B2B2B2B2B
       R = AA16D68ABA9E58B0F55A89C7C138A81A
   F/G entry 4 (G):
   C[ 0] = B6C6FB006E4D906CDCE7AD5A08B2ECE8DEC8CD24940C94806900872EADC50C5F
   C[ 1] = F1C45CF9F96A47BF3918BB22D04A5800F5A98EFCA20074C02C77256D7BA7202E
   C[ 2] = CB86AE3970057DDA
   X[ 0] = 0A8E01859CD916537B7B7B7B2B2B2B2B
       R = 6929E5DDD3BD4CAEDFD5035A49C1B1E9
   F/G entry 5 (F with DS): padded=0, domain=1, finalize=0
   C[ 0] = 3BF4433B3CE9695716BE4FADD308AB2DE4EAE7450AC019F3DDB2F56D99A7C841
   C[ 1] = AA16D68ABA9E58B0F55A89C7C138A81A6929E5DDD3BD4CAEDFD5035A49C1B1E9
   C[ 2] = 57F67F117DAF47CD
   X[ 0] = 0A8E01859CD916537B7B7B7B2B2B2B2B
       R = 57F67F117DAF47CD4621C774785D6874
       I = F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   F/G entry 6 (G):
   C[ 0] = E2B4E8B7111F223F608565858C19EB3D16371AF7888832BFCD7B66C120F2A2F0
   C[ 1] = 83E288E1E0736CE3073AFCAA89B0CC774C80AD181EC4340A727C85819A357F7B
   C[ 2] = 7EBAA95874A35362
   X[ 0] = 0A8E01859CD916537B7B7B7B2B2B2B2B
       R = 9268039ACDA85F60402FDD0A5E8232DB
   Final state:
   C[ 0] = 8E94247EDB66DEF0554ADC3CC237773F3FB132D1C719DE8B36D628F1C0483CA5
   C[ 1] = EE489881D3D5153B3BF8C5906ACCFA5C43134F86D3F7F585E6A1EFDA9D866918
   C[ 2] = 39CB6D54B8378E58
   X[ 0] = 0A8E01859CD916537B7B7B7B2B2B2B2B
       R = 3C7565296C650FA300B1126B78F9A50A
   CipherText:
   Tag:            9268039ACDA85F60402FDD0A5E8232DB3C7565296C650FA300B1126B78F9A50A
```

Listing 26: Detailed test vector: m=0, a=1, s=0

```
Encrypting 0 bytes message with 1 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:            000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:          F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: AA
   F/G entry 0 (F with DS): padded=0, domain=1, finalize=0
   C[ 0] = 000102030405060708090A0B0C0D0E0FFF03030367171717E700000000000070
   C[ 1] = 86767976F8F7F7F7F00080070000F000DEDEDEDEDBDBDBDB56617047021D140B
   C[ 2] = A139B22A53DF58D4
   X[ 0] = 0A8E01859CD916537B7B7B7B2B2B2B2B
       R = 00000000000000000000000000000000
       I = F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Padded A. data:  AA010000000000000000000000000000
   F/G entry 1 (F with DS): padded=1, domain=2, finalize=1
   C[ 0] = BB3981D6525A7FBDFA4CE0449B8F6FBB702755114446EEB6F64FE69510C40698
   C[ 1] = 34C79A74CA5F8415BE8741CBFC0686D72BC0DCCF0346FC4E8C812A7BB5ACC0F5
   C[ 2] = 0F0C3C668C70129E
   X[ 0] = 0A8E01859CD916537B7B7B7B2B2B2B2B
       R = 448864D0BE66F4FE319851F0284C06AD
       I = AA010000000000000000000000000000
   F/G entry 2 (G):
   C[ 0] = 8B29F43BD5C3815B67F3D0B3A97704498A4FAED0826B219C5AC0AC487C2B8F07
   C[ 1] = 5E10B0813530438A54A2EA9DD57F014082D843C3F08A766E1CB252A111C74F88
   C[ 2] = 9652BA284DFDE7A9
   X[ 0] = 0A8E01859CD916537B7B7B7B2B2B2B2B
       R = BCD2933B6992216784AF3F1EF2B28A1C
   Final state:
   C[ 0] = 468190ED58DF6391F98821EBF631072736D5F42F91F6F487D1B9050DD7C706F0
   C[ 1] = 15B62565010D250016E3A8371358F4FC4C5059F549E867D2E45E1790F8BF97CD
   C[ 2] = 4B6DDB21A41C9D28
   X[ 0] = 0A8E01859CD916537B7B7B7B2B2B2B2B
       R = E218EE7A4325BA5B21F8E522A95C68C8
   CipherText:
   Tag:             BCD2933B6992216784AF3F1EF2B28A1CE218EE7A4325BA5B21F8E522A95C68C8
```

Listing 27: Detailed test vector: m=1, a=0, s=0

```
Encrypting 1 bytes message with 0 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Message:         DD
   F/G entry 0 (F with DS): padded=0, domain=1, finalize=0
   C[ 0] = 000102030405060708090A0B0C0D0E0FFF03030367171717E700000000000070
   C[ 1] = 86767976F8F7F7F7F00080070000F000DEDEDEDEDBDBDBDB56617047021D140B
   C[ 2] = A139B22A53DF58D4
   X[ 0] = 0A8E01859CD916537B7B7B7B2B2B2B2B
       R = 00000000000000000000000000000000
       I = F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Padded Message:  DD010000000000000000000000000000
   F/G entry 1 (F with DS): padded=1, domain=3, finalize=1
   C[ 0] = BB3981D6525A7FBDFA4CE0449B8F6FBB702755114446EEB6F64FE69510C40698
   C[ 1] = 34C79A74CA5F8415BE8741CBFC0686D72BC0DCCF0346FC4E8C812A7BB5ACC0F5
   C[ 2] = 0F0C3C668C70129E
   X[ 0] = 0A8E01859CD916537B7B7B7B2B2B2B2B
       R = 448864D0BE66F4FE319851F0284C06AD
       I = DD010000000000000000000000000000
   F/G entry 2 (G):
   C[ 0] = CBFA7FC1F221ADB1C41876673F0ED595432CA77936EF68F265FA7838168BB874
   C[ 1] = 49BB2693BB1AFB2900C0B512F1F2D3E6D8BC0985CCE752CBD26AA16E0297F013
   C[ 2] = C4187A644C610FD1
   X[ 0] = 0A8E01859CD916537B7B7B7B2B2B2B2B
       R = 2237FA30F612334B7EBDFC9AD4979EFB
   Final state:
   C[ 0] = 2F3494E724A5734BC611003C9920CFB5298FB085034D6F8FE3EEF3451CCF65D2
   C[ 1] = 2CA0E9368D25AB916D16846AFD604EC42C9F8AA6827D525B2DCC6CB88F96698A
   C[ 2] = C9A40730FEBADD53
   X[ 0] = 0A8E01859CD916537B7B7B7B2B2B2B2B
       R = E6EE8DF05A5848046B933ECDE8F3C627
   CipherText:      99
   Tag:             2237FA30F612334B7EBDFC9AD4979EFBE6EE8DF05A5848046B933ECDE8F3C627
```

Listing 28: Detailed test vector: m=1, a=1, s=0

```
Encrypting 1 bytes message with 1 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: AA
   Message:         DD
   F/G entry 0 (F with DS): padded=0, domain=1, finalize=0
   C[ 0] = 000102030405060708090A0B0C0D0E0FFF03030367171717E700000000000070
   C[ 1] = 86767976F8F7F7F7F00080070000F000DEDEDEDEDBDBDBDB56617047021D140B
   C[ 2] = A139B22A53DF58D4
   X[ 0] = 0A8E01859CD916537B7B7B7B2B2B2B2B
       R = 00000000000000000000000000000000
       I = F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Padded A. data:  AA010000000000000000000000000000
   F/G entry 1 (F with DS): padded=1, domain=2, finalize=0
   C[ 0] = BB3981D6525A7FBDFA4CE0449B8F6FBB702755114446EEB6F64FE69510C40698
   C[ 1] = 34C79A74CA5F8415BE8741CBFC0686D72BC0DCCF0346FC4E8C812A7BB5ACC0F5
   C[ 2] = 0F0C3C668C70129E
   X[ 0] = 0A8E01859CD916537B7B7B7B2B2B2B2B
       R = 448864D0BE66F4FE319851F0284C06AD
       I = AA010000000000000000000000000000
   Padded Message:  DD010000000000000000000000000000
   F/G entry 2 (F with DS): padded=1, domain=3, finalize=1
   C[ 0] = F54DBE30E911D447732FC2CEE024CA04B38E17997A276CEC3EC9E68F6BD2F366
   C[ 1] = 96A27CAD8A5C0282498A4FA084288A0D2C5BE414B72BDF545B15B9D7A62209AD
   C[ 2] = 92E35F216F29BF80
   X[ 0] = 0A8E01859CD916537B7B7B7B2B2B2B2B
       R = BFBE727844CCF50E4362DEF1ADF93A15
       I = DD010000000000000000000000000000
   F/G entry 3 (G):
   C[ 0] = E1B4E22EC8C6DD6010B62361CCB71965C1166C7FB356BC009E8FEBA69F88A2EF
   C[ 1] = 6778BC3C4AA34C804C5A76F782E8BB8FE9D1CAD038F5D16933F83B9E3DA7F4A7
   C[ 2] = ADD5C32FFF4E84D0
   X[ 0] = 0A8E01859CD916537B7B7B7B2B2B2B2B
       R = 9FDF52C2D33FC048752FBEA5CD4D8137
   Final state:
   C[ 0] = 2702D6339E46BCB19D7812B84D8CF93DADF18E4CA35252A9872426A3986C9879
   C[ 1] = A5278252369FBD98EBA223BFA6EA4E65D6E6E5B0CAA3C8D6A2A82ED82C5A0307
   C[ 2] = 702EFF73CABC7E84
   X[ 0] = 0A8E01859CD916537B7B7B7B2B2B2B2B
       R = CB224CD0BD276895B3DC7956CF9A2D54
   CipherText:      62
   Tag:             9FDF52C2D33FC048752FBEA5CD4D8137CB224CD0BD276895B3DC7956CF9A2D54
```

Listing 29: Test vectors

```
Encrypting 0 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   CipherText:
   Tag:             8E3007C23CAB803D4BE7D74DDD8D6B6DD32044D4E0F0BF1E74E6B50E25580436
Encrypting 1 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         00
   Padded Message:  00010000000000000000000000000000
   CipherText:      5D
   Tag:             9B7CC3704F3742960B1A5162BC1C6052516241DD90BCBF2EC1D1311EDE9ECA9C
Encrypting 2 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         0001
   Padded Message:  00010100000000000000000000000000
   CipherText:      5D38
   Tag:             139FB3A44F1E19975B7B6F0E31C6DDDB36619CB15E016973AAC8F27F74800518
Encrypting 3 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102
   Padded Message:  00010201000000000000000000000000
   CipherText:      5D38C0
   Tag:             ACB261A128390165E477810494F081D4233B8762C9603700A61BF240E3F4A923
Encrypting 4 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         00010203
   Padded Message:  00010203010000000000000000000000
   CipherText:      5D38C033
   Tag:             9269364CBC59E1170231D7C70E61DE189A9EFAE6BD04BC376B5B3E91C1FCD3CC
Encrypting 5 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         0001020304
   Padded Message:  00010203040100000000000000000000
   CipherText:      5D38C033BF
   Tag:             F38D130AE94B178D3E0D4D1389DFF106E3DC7446838D6200B2D2D81CACDE345B
Encrypting 6 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405
   Padded Message:  00010203040501000000000000000000
   CipherText:      5D38C033BF85
   Tag:             E0AB83AFDC77AFDAA52913F4D1250B3C83FF5AF21A949217B0ABA77630ABD7C0
Encrypting 7 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         00010203040506
   Padded Message:  00010203040506010000000000000000
   CipherText:      5D38C033BF85D5
   Tag:             738A0B0DE65528ECCEC8E47EBBE9AAC2C882245829CB0E5F2D4838F0DBBCACA2
Encrypting 8 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         0001020304050607
   Padded Message:  00010203040506070100000000000000
   CipherText:      5D38C033BF85D53E
   Tag:             53DE459ECEFCA512A279126272C8C906252D57CC39D2563FAD9CC706E3E56AD0
Encrypting 9 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708
   Padded Message:  00010203040506070801000000000000
   CipherText:      5D38C033BF85D53E14
   Tag:             3A03371522043008EC5231F35323C4A84DABF01399CCA8A0798C2B6D41D118BD
Encrypting 10 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         00010203040506070809
   Padded Message:  000102030405060708090100000000000
   CipherText:      5D38C033BF85D53E14A9
   Tag:             BD7CBE6EF165C25773C4528C5A7C133B61BC21C5F0EBA1059826497E24D161E8
Encrypting 11 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A
   Padded Message:  000102030405060708090A0100000000
   CipherText:      5D38C033BF85D53E14A93E
   Tag:             921C0EF7C44A342238C7B4D45248245FEDAB60293EC467F669E6517DD059C1C6
Encrypting 12 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B
   Padded Message:  000102030405060708090A0B01000000
   CipherText:      5D38C033BF85D53E14A93E04
   Tag:             AF7F1CE177CBEEDA83A10021BBCDDFCD0B8EA9BD92BC373CBEF667F212A4B362
Encrypting 13 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C
   Padded Message:  000102030405060708090A0B0C010000
   CipherText:      5D38C033BF85D53E14A93E04A9
   Tag:             3C293F6C46E31A7363008701637B36E6C1D14C2CC62B5F7DF915A4AC41BDBD71
Encrypting 14 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D
   Padded Message:  000102030405060708090A0B0C0D0100
   CipherText:      5D38C033BF85D53E14A93E04A987
   Tag:             C261DD08494AD763EDBE069518AF54DA1FE8E00677ED0B3867F2521E372FD106
Encrypting 15 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E
   Padded Message:  000102030405060708090A0B0C0D0E01
   CipherText:      5D38C033BF85D53E14A93E04A98711
   Tag:             E43EFBF7736661D6F84231C7E043394A255B76BCB40E14371C1C330EACA103D4
Encrypting 16 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F
   CipherText:      5D38C033BF85D53E14A93E04A987116A
   Tag:             788C658074093E3F86284D48FEB6D63FA59CEC7AD6E3F73DC884D84743915BBA
Encrypting 17 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F10
   Padded Message:  000102030405060708090A0B0C0D0E0F1001000000000000000000000000000000
   CipherText:      5D38C033BF85D53E14A93E04A987116A20
   Tag:             8441C5B02BF34773D5478CE530D8BF540B4E1D71AB998F5D1C2A1E809AF5ACDD
Encrypting 18 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F1011
   Padded Message:  000102030405060708090A0B0C0D0E0F101101000000000000000000000000000
   CipherText:      5D38C033BF85D53E14A93E04A987116A209A
   Tag:             383E310745605981DA9DFC8CC788242783B421BC6F63627AF3ECF57ABB20FE61
Encrypting 19 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F101112
   Padded Message:  000102030405060708090A0B0C0D0E0F1011120100000000000000000000000000
   CipherText:      5D38C033BF85D53E14A93E04A987116A209A65
   Tag:             A16944D07E1CDD0EBDAFC3F4C4ECBA32F14E0C0CCFD9A68BADEF14DD0B41AB7A
Encrypting 20 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F10111213
   Padded Message:  000102030405060708090A0B0C0D0E0F10111213010000000000000000000000
   CipherText:      5D38C033BF85D53E14A93E04A987116A209A65D4
   Tag:             EF19F8427FDC6D2352C3CE315E13AFC433D2482812A89A6D7CD0F9E87CFD8734
Encrypting 21 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F1011121314
   Padded Message:  000102030405060708090A0B0C0D0E0F10111213140100000000000000000000
   CipherText:      5D38C033BF85D53E14A93E04A987116A209A65D44D
   Tag:             2E9AE19FD8BBF2DCCDBC833848C5251DBBFB492B013198DE9517FC5BF15F9785
Encrypting 22 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F101112131415
   Padded Message:  000102030405060708090A0B0C0D0E0F10111213141501000000000000000000
   CipherText:      5D38C033BF85D53E14A93E04A987116A209A65D44D87
   Tag:             48356204BA2BEA2CD1EEC53453CF4B9875AD404E9AC6EA1A06A8CAA91BE78742
Encrypting 23 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F10111213141516
   Padded Message:  000102030405060708090A0B0C0D0E0F10111213141516010000000000000000
   CipherText:      5D38C033BF85D53E14A93E04A987116A209A65D44D8707
   Tag:             CC9D90263115E2FCA842BBEBDE116ABA4996D5F30F88514A7CB44566EE2AF90C
Encrypting 24 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F1011121314151617
   Padded Message:  000102030405060708090A0B0C0D0E0F10111213141516170100000000000000
   CipherText:      5D38C033BF85D53E14A93E04A987116A209A65D44D8707CD
   Tag:             E894E86E618412DA6BB2B74A051242B76EC7172B710968B8656D708CA610AD48
Encrypting 25 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F101112131415161718
   Padded Message:  000102030405060708090A0B0C0D0E0F101112131415161718010000000000000
   CipherText:      5D38C033BF85D53E14A93E04A987116A209A65D44D8707CD16
   Tag:             9B686273D4891CE11B96216D468E1A12FD6A866A94A6600B9DCACD5902548613
Encrypting 26 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F10111213141516171819
   Padded Message:  000102030405060708090A0B0C0D0E0F10111213141516171819010000000000
   CipherText:      5D38C033BF85D53E14A93E04A987116A209A65D44D8707CD16D0
   Tag:             C1AAD977AEA76615E40F12F95D851A07D24B7902B78EB6829E4593B242558DAB
Encrypting 27 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F101112131415161718191A
   Padded Message:  000102030405060708090A0B0C0D0E0F101112131415161718191A0100000000
   CipherText:      5D38C033BF85D53E14A93E04A987116A209A65D44D8707CD16D045
   Tag:             619F023D3947BA3E5712CD89AC8AD213CB1BD5282D84E0BB4C5DB7499BEDA9FD
Encrypting 28 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F101112131415161718191A1B
   Padded Message:  000102030405060708090A0B0C0D0E0F101112131415161718191A1B01000000
   CipherText:      5D38C033BF85D53E14A93E04A987116A209A65D44D8707CD16D045D6
   Tag:             DFD56A67241C8BC6956459EF55B099AFB54DDD764591EB074DDF54F4D8DC81AC
Encrypting 29 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C
   Padded Message:  000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C010000
   CipherText:      5D38C033BF85D53E14A93E04A987116A209A65D44D8707CD16D045D68D
   Tag:             719CE05A0E0133BEA4981959C55171C33EDD705198D8B383D3FA5DB859AB3197
Encrypting 30 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D
   Padded Message:  000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D0100
   CipherText:      5D38C033BF85D53E14A93E04A987116A209A65D44D8707CD16D045D68D77
   Tag:             6F554907022442F77A6F73102E850FDA05F75FC702D772874E021F5136307ACD
Encrypting 31 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E
   Padded Message:  000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E01
   CipherText:      5D38C033BF85D53E14A93E04A987116A209A65D44D8707CD16D045D68D776B
   Tag:             5D0F3409FE5A644CB1FB0B59B489CE53DC813953DD027681E7CC4C2A29341C05
Encrypting 32 bytes message with 16 bytes of associated data and 0 bytes of static data
Key size: 32 bytes
   Key:             000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Nonce:           F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
   Associated data: 00000000000000000000000000000000
   Message:         000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   CipherText:      5D38C033BF85D53E14A93E04A987116A209A65D44D8707CD16D045D68D776B16
   Tag:             2EA33FD52360CB7A2636C09430C0E6721942DCB423487247CFE50304E044C6C6
```

Listing 30: Iterative test vector

```
Encrypting null message 100 times with tag feedback as associated data
   CipherText: E05AD218BF732DD42164F6631033C5FE7088881C0E6FB785072B73B0E14DE3C6
```

14.7 DryGascon256 hash test vectors

Listing 31: Detailed test vector

```
Hashing 8 bytes message: 0001020304050607
   Padded Message:  00010203040506070100000000000000
   F/G entry 0 (F with DS): padded=1, domain=2, finalize=1
   C[ 0] = 243F6A8885A308D313198A2E03707344A4093822299F31D0082EFA98EC4E6C89
   C[ 1] = 243F6A8885A308D313198A2E03707344A4093822299F31D0082EFA98EC4E6C89
   C[ 2] = 243F6A8885A308D3
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 00000000000000000000000000000000
       I = 00010203040506070100000000000000
ds = b
MixPhaseRound entry  0 243F6A8885A308D313198A2E03707344A4093822299F31D0082EFA98EC4E6C89243F
      ↪ 6A8885A308D313198A2E03707344A4093822299F31D0082EFA98EC4E6C89243F6A8885A308D3
CoreRound entry      0 61174B6E85A308D35631ABC803707344E12119C4299F31D04D06DB7EEC4E6C891CEF
      ↪ 79FF85A308D35631ABC803707344E12119C4299F31D04D06DB7EEC4E6C899A6B0C4785A308D3
MixPhaseRound entry  1 A04B48B71A2068E50D26A24BCD92C80D1EE862BA657B437864D5D11B345839E7EC92
      ↪ 916D88AFE47BEE2AACEA0E1411CDFC3A96F0D41B9F42AA0F41DD23E5B410DF5522FA80676AE8
CoreRound entry      1 E56369511A2068E5480E83ADCD92C80D5BC0435C657B4378503CDD77345839E7A9
      ↪ BAB08B88AFE47BAB028D0C0E1411CDB912B716D41B9F42EF27603B23E5B410E785318D80676AE8
MixPhaseRound entry  2 E568CC04A3DA2360668D632F0EE27988B90B935BE981165C79ED03581EF82A098E88
      ↪ 0B7B17338FC02E29EA4941D037B21F46749EDA65969CF306837D1A70406CCBBD10CEC563B6B9
CoreRound entry      2 A040EDE2A3DA236023A542C90EE2798881DB802CE981165C413D102F1EF82A09CBA0
      ↪ 2A9D17338FC06B01CBAF41D037B2A1121251DA65969CCBD6900A1A70406C8E953128C563B6B9
MixPhaseRound entry  3 E1DD78D69CF6FFC729A4DCDB94B4A40EAD81BF6FB65383EC4EDFAF2D1C808702F563
      ↪ 8ECDC18511CA3E4BC7E4A089538B1713D5EC705DA898F168213953056B0FF97F9C00B3AA3B97
CoreRound entry      3 A4F559309CF6FFC71D4DD0B794B4A40E9551AC18B65383EC0BF78ECB1C808702B04
      ↪ BAF2BC18511CA069BD493A089538B523BF40A705DA898B44000DF53056B0FBC57BDE6B3AA3B97
MixPhaseRound entry  4 8AAAA15ADB133242A37784915525A798B25481B4DB94DF082B2E94B279E605B15517
      ↪ 38B5306A597CCC1760B2951FA2D19991AF8F636606AF0BE4B2D60B98540132D87BCE59831F1B
CoreRound entry      4 CF8280BCDB133242E65FA5775525A798F77CA052DB94DF086E06B55479E605B1103F
      ↪ 1953306A597C893F4154951FA2D1DCB98E69636606AF4ECC93300B98540177F05A2859831F1B
MixPhaseRound entry  5 10018C6767FD38CEEBB3B6D175ED2C6B4D00ECDEF723A294655593B08D0A7A1942F9
      ↪ 2B867C868730937AD6EF97944A60DCAC64B2136322F18A9B3A5FA10913CE14B204C425C0934E
CoreRound entry      5 5529AD8167FD38CEAE9B973775ED2C6B0828CD38F723A294207DB2568D0A7A1907D1
      ↪ 0A607C868730D652F70997944A6099844554136322F1CFB31BB9A10913CE519A252225C0934E
MixPhaseRound entry  6 0DF7D6D4A9F263AAD557B5D90354CC5CED3FB3D5143A4313E357A4827499F862F48
      ↪ CD1280E89F67FB22C8FF1220C305781CC2DD96F4072ECBADE0FE820A3F4792F34F43B7803048D
CoreRound entry      6 48DFF732A9F263AA907F943F0354CC5CA8179233143A4313A67F85647499F862B1A4
      ↪ F0CE0E89F67FF704AE17220C3057C4E40C3F6F4072ECFFF62E0E20A3F4796A1CD5DD7803048D
MixPhaseRound entry  7 0CAAB7B72B4CA935834736668483673B6B5B31E4E237E33BAA1EB0E17B4828DE3936
      ↪ 9DD8228797F7F4F25823A57AB972E3AC7FE2C32A2DEBB3D93925C03E16508701A6B1019E722F
   After mix phase:
   C[ 0] = 498296512B4CA935B7AE3A0A8483673BD50F572BE237E33BEF3691077B4828DE
   C[ 1] = 7C1EBC3E228797F7B1DA79C5A57AB972A6845E04C32A2DEBF6F118C3C03E1650
   C[ 2] = C2298757019E722F
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 00000000000000000000000000000000
   CoreRound entry 0:
   C[ 0] = 498296512B4CA935B7AE3A0A8483673BD50F572BE237E33BEF3691077B4828DE
   C[ 1] = 7C1EBC3E228797F7B1DA79C5A57AB972A6845E04C32A2DEBF6F118C3C03E1650
   C[ 2] = C2298757019E722F
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 00000000000000000000000000000000
   Accumulate entry 0:
   C[ 0] = ED946DE63DAFC0EAD980F015D5B7522D28AF7A7BBB8191E93E74F98DC2D1BFEE
   C[ 1] = EBF90500293827B06781907170F5C01C44FF96C5A09158B58BCBD7BB6263A545
   C[ 2] = BCF651C3980160A9
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 00000000000000000000000000000000
accumulate in[0] = ED946DE63DAFC0EAD980F015D5B7522D
accumulate in[1] = BB8191E93E74F98DC2D1BFEE28AF7A7B
accumulate in[2] = 6781907170F5C01CEBF90500293827B0
accumulate in[3] = 6263A54544FF96C5A09158B58BCBD7BB
   CoreRound entry 1:
   C[ 0] = ED946DE63DAFC0EAD980F015D5B7522D28AF7A7BBB8191E93E74F98DC2D1BFEE
   C[ 1] = EBF90500293827B06781907170F5C01C44FF96C5A09158B58BCBD7BB6263A545
   C[ 2] = BCF651C3980160A9
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 53F7C93B37D16FBE5039124E5FEBD85D
   Accumulate entry 1:
   C[ 0] = 63B6E976C20FAA04BA253C391EB33BC8686C9C97F72E1C99C350D662A086A4D5
   C[ 1] = 7951C6265F06764ECE2013D3AA5773ADA8082D22F604D28FFBA31B6DBF9B1C1E
   C[ 2] = 0F3485F06B5BD891
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 53F7C93B37D16FBE5039124E5FEBD85D
accumulate in[0] = 63B6E976C20FAA04BA253C391EB33BC8
accumulate in[1] = F72E1C99C350D662A086A4D5686C9C97
accumulate in[2] = CE2013D3AA5773AD7951C6265F06764E
accumulate in[3] = BF9B1C1EA8082D22F604D28FFBA31B6D
   CoreRound entry 2:
   C[ 0] = 63B6E976C20FAA04BA253C391EB33BC8686C9C97F72E1C99C350D662A086A4D5
   C[ 1] = 7951C6265F06764ECE2013D3AA5773ADA8082D22F604D28FFBA31B6DBF9B1C1E
   C[ 2] = 0F3485F06B5BD891
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = B6D4331934D14D57C5CF9E0B8D911221
   Accumulate entry 2:
   C[ 0] = 5E346980F82D3BBFFCB5438561CFEA8653DC81A47D0A4BC5366104B7EA740A40
   C[ 1] = E1207D285506E4C81023975466A5337F59182FB4B5072E8F44442A272C8500BA
   C[ 2] = 0A97C9883318B51B
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = B6D4331934D14D57C5CF9E0B8D911221
accumulate in[0] = 5E346980F82D3BBFFCB5438561CFEA86
accumulate in[1] = 7D0A4BC5366104B7EA740A4053DC81A4
accumulate in[2] = 1023975466A5337FE1207D285506E4C8
accumulate in[3] = 2C8500BA59182FB4B5072E8F44442A27
   CoreRound entry 3:
   C[ 0] = 5E346980F82D3BBFFCB5438561CFEA8653DC81A47D0A4BC5366104B7EA740A40
   C[ 1] = E1207D285506E4C81023975466A5337F59182FB4B5072E8F44442A272C8500BA
   C[ 2] = 0A97C9883318B51B
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = A94C86B2C5206E9487298469AEC0B7EC
   Accumulate entry 3:
   C[ 0] = 800B67C48ADBBBE186100F5D4FFBEBEC93C3C9073FCBF066FBBD34B383DB53AF
   C[ 1] = 523B2714CA7AEA03CBA3473E2E52BBC0645A9A41AA64D8737BC3312CD3D9E6B1
   C[ 2] = EDC97AD0445061CA
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = A94C86B2C5206E9487298469AEC0B7EC
accumulate in[0] = 800B67C48ADBBBE186100F5D4FFBEBEC
accumulate in[1] = 3FCBF066FBBD34B383DB53AF93C3C907
accumulate in[2] = CBA3473E2E52BBC0523B2714CA7AEA03
accumulate in[3] = D3D9E6B1645A9A41AA64D8737BC3312C
   CoreRound entry 4:
   C[ 0] = 800B67C48ADBBBE186100F5D4FFBEBEC93C3C9073FCBF066FBBD34B383DB53AF
   C[ 1] = 523B2714CA7AEA03CBA3473E2E52BBC0645A9A41AA64D8737BC3312CD3D9E6B1
   C[ 2] = EDC97AD0445061CA
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 0EF6B09FFE4EC0477ABD27FCC3414E28
   Accumulate entry 4:
   C[ 0] = 01A37DFADFCEE68587CCC62C1F9DC3020B7BAEFFB79A8734D8057D2FF757B71A
   C[ 1] = 178F68972A5E32664D0792186838C6D548D8A58BF2A78DC0EF4FDB521B4C1B3D
   C[ 2] = B3A508261C60B248
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 0EF6B09FFE4EC0477ABD27FCC3414E28
```

```
accumulate in[0] = 01A37DFADFCEE68587CCC62C1F9DC302
accumulate in[1] = B79A8734D8057D2FF757B71A0B7BAEFF
accumulate in[2] = 4D0792186838C6D5178F68972A5E3266
accumulate in[3] = 1B4C1B3D48D8A58BF2A78DC0EF4FDB52
   CoreRound entry 5:
   C[ 0] = 01A37DFADFCEE68587CCC62C1F9DC3020B7BAEFFB79A8734D8057D2FF757B71A
   C[ 1] = 178F68972A5E32664D0792186838C6D548D8A58BF2A78DC0EF4FDB521B4C1B3D
   C[ 2] = B3A508261C60B248
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = EE84C374D96538B3EF0EB39D12B6CAE1
   Accumulate entry 5:
   C[ 0] = 9B46C6B8976B4E3333DC221C95ED18E0BB3541D98390B879F0353A80AAA84ACE
   C[ 1] = EF9F782E5E8324E82A4C4C219B9FA3652D606374DAA1591C97C3AAA326BB286B
   C[ 2] = 4AD5282DF1D5BDA6
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = EE84C374D96538B3EF0EB39D12B6CAE1
accumulate in[0] = 9B46C6B8976B4E3333DC221C95ED18E0
accumulate in[1] = 8390B879F0353A80AAA84ACEBB3541D9
accumulate in[2] = 2A4C4C219B9FA365EF9F782E5E8324E8
accumulate in[3] = 26BB286B2D606374DAA1591C97C3AAA3
   CoreRound entry 6:
   C[ 0] = 9B46C6B8976B4E3333DC221C95ED18E0BB3541D98390B879F0353A80AAA84ACE
   C[ 1] = EF9F782E5E8324E82A4C4C219B9FA3652D606374DAA1591C97C3AAA326BB286B
   C[ 2] = 4AD5282DF1D5BDA6
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = FAA5D9FF08C48C114344FA7DF52E1D93
   Accumulate entry 6:
   C[ 0] = 5147A6E1628DF20C6C3ED14F2D124083B371C36F4A40251B83F573127D324806
   C[ 1] = A37BF09A00AFD46E747B0C7A0CA0E9F40A5E88CBFCA7F333169E7E514C37D90B
   C[ 2] = 5010D52E0A72616A
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = FAA5D9FF08C48C114344FA7DF52E1D93
accumulate in[0] = 5147A6E1628DF20C6C3ED14F2D124083
accumulate in[1] = 4A40251B83F573127D324806B371C36F
accumulate in[2] = 747B0C7A0CA0E9F4A37BF09A00AFD46E
accumulate in[3] = 4C37D90B0A5E88CBFCA7F333169E7E51
   CoreRound entry 7:
   C[ 0] = 5147A6E1628DF20C6C3ED14F2D124083B371C36F4A40251B83F573127D324806
   C[ 1] = A37BF09A00AFD46E747B0C7A0CA0E9F40A5E88CBFCA7F333169E7E514C37D90B
   C[ 2] = 5010D52E0A72616A
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = D9EE8F74EF426C300D94609D7D7C3440
   Accumulate entry 7:
   C[ 0] = AFD67766FE95CC59D97A246E9D43DE4F23D2712D76C6A6DAEC77C894927DAA25
   C[ 1] = 93762763608B49AD3EDFA778C88413A863F0BDA9B4DF4C7E1CB4517392B80421
   C[ 2] = 12ACD2284F4D4232
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = D9EE8F74EF426C300D94609D7D7C3440
accumulate in[0] = AFD67766FE95CC59D97A246E9D43DE4F
accumulate in[1] = 76C6A6DAEC77C894927DAA2523D2712D
accumulate in[2] = 3EDFA778C88413A893762763608B49AD
accumulate in[3] = 92B8042163F0BDA9B4DF4C7E1CB45173
   F/G entry 1 (G):
   C[ 0] = AFD67766FE95CC59D97A246E9D43DE4F23D2712D76C6A6DAEC77C894927DAA25
   C[ 1] = 93762763608B49AD3EDFA778C88413A863F0BDA9B4DF4C7E1CB4517392B80421
   C[ 2] = 12ACD2284F4D4232
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = AC99FD9156D4C6FC613A85CBBFD283FC
   CoreRound entry 0:
   C[ 0] = AFD67766FE95CC59D97A246E9D43DE4F23D2712D76C6A6DAEC77C894927DAA25
   C[ 1] = 93762763608B49AD3EDFA778C88413A863F0BDA9B4DF4C7E1CB4517392B80421
   C[ 2] = 12ACD2284F4D4232
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
```

```
       R = 00000000000000000000000000000000
   Accumulate entry 0:
   C[ 0] = 5B6F45BE1805FEF0931C79FC451BA8B26DFA45190673A7EE56524780C13C9902
   C[ 1] = E6CCC120CF4DE33C02A1884879182E75D0911E124404F1D7470ED74D17C6BB3B
   C[ 2] = CBB7B90DB4A70C90
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 00000000000000000000000000000000
accumulate in[0] = 5B6F45BE1805FEF0931C79FC451BA8B2
accumulate in[1] = 0673A7EE56524780C13C99026DFA4519
accumulate in[2] = 02A1884879182E75E6CCC120CF4DE33C
accumulate in[3] = 17C6BB3BD0911E124404F1D7470ED74D
   CoreRound entry 1:
   C[ 0] = 5B6F45BE1805FEF0931C79FC451BA8B26DFA45190673A7EE56524780C13C9902
   C[ 1] = E6CCC120CF4DE33C02A1884879182E75D0911E124404F1D7470ED74D17C6BB3B
   C[ 2] = CBB7B90DB4A70C90
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 487BD123E7DE8917F0E8D009A0A2D9DA
   Accumulate entry 1:
   C[ 0] = 70D85385874A952019EA387B2CD2E4B8EDCE665D1BB9BE2E7B71BFA89E64691F
   C[ 1] = F6041212D7292DEF047A8DAD05D5AD8AC0EC85393FFAD263DCD3F5324CBBF474
   C[ 2] = C46B76C0CF35F558
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 487BD123E7DE8917F0E8D009A0A2D9DA
accumulate in[0] = 70D85385874A952019EA387B2CD2E4B8
accumulate in[1] = 1BB9BE2E7B71BFA89E64691FEDCE665D
accumulate in[2] = 047A8DAD05D5AD8AF6041212D7292DEF
accumulate in[3] = 4CBBF474C0EC85393FFAD263DCD3F532
   CoreRound entry 2:
   C[ 0] = 70D85385874A952019EA387B2CD2E4B8EDCE665D1BB9BE2E7B71BFA89E64691F
   C[ 1] = F6041212D7292DEF047A8DAD05D5AD8AC0EC85393FFAD263DCD3F5324CBBF474
   C[ 2] = C46B76C0CF35F558
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 6BDB4551DEDC8B2CBE98411C6A4483E2
   Accumulate entry 2:
   C[ 0] = 0C1F97EA106D6E61E2598CCD8D61545A98868B4440BEBAA2C1CF4D29F6924EC5
   C[ 1] = 243869F8847E34F25BFAA8353E7E28D49EA6D5ABFCAED86B60B85809434137F3
   C[ 2] = 91C516AD531AC281
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 6BDB4551DEDC8B2CBE98411C6A4483E2
accumulate in[0] = 0C1F97EA106D6E61E2598CCD8D61545A
accumulate in[1] = 40BEBAA2C1CF4D29F6924EC598868B44
accumulate in[2] = 5BFAA8353E7E28D4243869F8847E34F2
accumulate in[3] = 434137F39EA6D5ABFCAED86B60B85809
   CoreRound entry 3:
   C[ 0] = 0C1F97EA106D6E61E2598CCD8D61545A98868B4440BEBAA2C1CF4D29F6924EC5
   C[ 1] = 243869F8847E34F25BFAA8353E7E28D49EA6D5ABFCAED86B60B85809434137F3
   C[ 2] = 91C516AD531AC281
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 3FC1F7DFAFA6551B72C532879B653007
   Accumulate entry 3:
   C[ 0] = CFDE0DA419E045E75EF2B0179B00C91A7AA1DD181A92165FB1A98A41BAC3C613
   C[ 1] = 7C6421F8BDAD29601EC609067C3830BD5A679125C0B6D50924A92418531BD510
   C[ 2] = 40D6A02719D9FA33
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 3FC1F7DFAFA6551B72C532879B653007
accumulate in[0] = CFDE0DA419E045E75EF2B0179B00C91A
accumulate in[1] = 1A92165FB1A98A41BAC3C6137AA1DD18
accumulate in[2] = 1EC609067C3830BD7C6421F8BDAD2960
accumulate in[3] = 531BD5105A679125C0B6D50924A92418
   CoreRound entry 4:
   C[ 0] = CFDE0DA419E045E75EF2B0179B00C91A7AA1DD181A92165FB1A98A41BAC3C613
   C[ 1] = 7C6421F8BDAD29601EC609067C3830BD5A679125C0B6D50924A92418531BD510
   C[ 2] = 40D6A02719D9FA33
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = A750303221B03B252A26B072E3C0297D
   Accumulate entry 4:
   C[ 0] = 7D66C65AE99768397CE23DF34C7F903D052B5304E262EEFA50A9FA69720D09E2
   C[ 1] = 55D7623789DEDEBBA38E5B15F837CBE101E20CD790DBABAA5DAF90C3B508B31B
   C[ 2] = 6A9EC49FEB7ABACC
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = A750303221B03B252A26B072E3C0297D
accumulate in[0] = 7D66C65AE99768397CE23DF34C7F903D
accumulate in[1] = E262EEFA50A9FA69720D09E2052B5304
accumulate in[2] = A38E5B15F837CBE155D7623789DEDEBB
accumulate in[3] = B508B31B01E20CD790DBABAA5DAF90C3
   CoreRound entry 5:
   C[ 0] = 7D66C65AE99768397CE23DF34C7F903D052B5304E262EEFA50A9FA69720D09E2
   C[ 1] = 55D7623789DEDEBBA38E5B15F837CBE101E20CD790DBABAA5DAF90C3B508B31B
   C[ 2] = 6A9EC49FEB7ABACC
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 2ED2F09C615B6E43E1C54DFE7EE5A43C
   Accumulate entry 5:
   C[ 0] = 28247B7F14E5154CC73E36342D71A0ED0583EB032F99CAD5D8A20588BC420875
   C[ 1] = E90BF7C8D1907ED438149E4BA8B144D52510AFF42DFF43365D31F6E5213CDA4A
   C[ 2] = 100AF011C41474FF
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 2ED2F09C615B6E43E1C54DFE7EE5A43C
accumulate in[0] = 28247B7F14E5154CC73E36342D71A0ED
accumulate in[1] = 2F99CAD5D8A20588BC4208750583EB03
accumulate in[2] = 38149E4BA8B144D5E90BF7C8D1907ED4
accumulate in[3] = 213CDA4A2510AFF42DFF43365D31F6E5
   CoreRound entry 6:
   C[ 0] = 28247B7F14E5154CC73E36342D71A0ED0583EB032F99CAD5D8A20588BC420875
   C[ 1] = E90BF7C8D1907ED438149E4BA8B144D52510AFF42DFF43365D31F6E5213CDA4A
   C[ 2] = 100AF011C41474FF
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 3047053720BD95A65E4DC741DAB667E3
   Accumulate entry 6:
   C[ 0] = D8A397162E65CE32C540246DFD4C2AC83A3AD661392D88742F261EA4A5A82677
   C[ 1] = 9DCAD8213AD1010A54CD9CA22315313DA5EE6A5EFEFF1801C751D3DD9632FCED
   C[ 2] = 31A9618C246794A6
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 3047053720BD95A65E4DC741DAB667E3
accumulate in[0] = D8A397162E65CE32C540246DFD4C2AC8
accumulate in[1] = 392D88742F261EA4A5A826773A3AD661
accumulate in[2] = 54CD9CA22315313D9DCAD8213AD1010A
accumulate in[3] = 9632FCEDA5EE6A5EFEFF1801C751D3DD
   CoreRound entry 7:
   C[ 0] = D8A397162E65CE32C540246DFD4C2AC83A3AD661392D88742F261EA4A5A82677
   C[ 1] = 9DCAD8213AD1010A54CD9CA22315313DA5EE6A5EFEFF1801C751D3DD9632FCED
   C[ 2] = 31A9618C246794A6
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 13367A1AA7051E535D90057BE040499D
   Accumulate entry 7:
   C[ 0] = 66FBC1A8A711CF42AF215D7684CA16196CD43758B20FAEE4B0A4AED4F3AACCDD
   C[ 1] = 998E170587177E369981CFC80FE21861812A09902BA8B555AFBE8D1C2C57A3B5
   C[ 2] = F2F9242D1472A76D
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 13367A1AA7051E535D90057BE040499D
accumulate in[0] = 66FBC1A8A711CF42AF215D7684CA1619
accumulate in[1] = B20FAEE4B0A4AED4F3AACCDD6CD43758
accumulate in[2] = 9981CFC80FE21861998E170587177E36
accumulate in[3] = 2C57A3B5812A09902BA8B555AFBE8D1C
   F/G entry 2 (G):
   C[ 0] = 66FBC1A8A711CF42AF215D7684CA16196CD43758B20FAEE4B0A4AED4F3AACCDD
   C[ 1] = 998E170587177E369981CFC80FE21861812A09902BA8B555AFBE8D1C2C57A3B5
```

```
    C[ 2] = F2F9242D1472A76D
    X[ 0] = 452821E638D01377BE5466CF34E90C6C
         R = 7214792B3E786E34B33D368020F79BF6
    CoreRound entry 0:
    C[ 0] = 66FBC1A8A711CF42AF215D7684CA16196CD43758B20FAEE4B0A4AED4F3AACCDD
    C[ 1] = 998E170587177E369981CFC80FE21861812A09902BA8B555AFBE8D1C2C57A3B5
    C[ 2] = F2F9242D1472A76D
    X[ 0] = 452821E638D01377BE5466CF34E90C6C
         R = 00000000000000000000000000000000
    Accumulate entry 0:
    C[ 0] = 1D607CD25A5995053B89BDBF00F277D449501EB5F175C693D30A0DEEE180254F
    C[ 1] = 9795A9315795A6E701C0E12646B7729814358B9C98723FA55CBC6101E4329415
    C[ 2] = 17BB6B0E56EADD04
    X[ 0] = 452821E638D01377BE5466CF34E90C6C
         R = 00000000000000000000000000000000
accumulate in[0] = 1D607CD25A5995053B89BDBF00F277D4
accumulate in[1] = F175C693D30A0DEEE180254F49501EB5
accumulate in[2] = 01C0E12646B772989795A9315795A6E7
accumulate in[3] = E432941514358B9C98723FA55CBC6101
    CoreRound entry 1:
    C[ 0] = 1D607CD25A5995053B89BDBF00F277D449501EB5F175C693D30A0DEEE180254F
    C[ 1] = 9795A9315795A6E701C0E12646B7729814358B9C98723FA55CBC6101E4329415
    C[ 2] = 17BB6B0E56EADD04
    X[ 0] = 452821E638D01377BE5466CF34E90C6C
         R = 09E7CF72DBD161EFD5EE0E64428BAE87
    Accumulate entry 1:
    C[ 0] = F99597E7741DB572F303577DCBFE6CB93BB7503F9046541601185DBD47224FCF
    C[ 1] = 918B4B83A2A85983524A1DD7601BD4F5AC3564567BA13A594207B90EE7F0805B
    C[ 2] = 472C648962292FF2
    X[ 0] = 452821E638D01377BE5466CF34E90C6C
         R = 09E7CF72DBD161EFD5EE0E64428BAE87
accumulate in[0] = F99597E7741DB572F303577DCBFE6CB9
accumulate in[1] = 9046541601185DBD47224FCF3BB7503F
accumulate in[2] = 524A1DD7601BD4F5918B4B83A2A85983
accumulate in[3] = E7F0805BAC3564567BA13A594207B90E
    CoreRound entry 2:
    C[ 0] = F99597E7741DB572F303577DCBFE6CB93BB7503F9046541601185DBD47224FCF
    C[ 1] = 918B4B83A2A85983524A1DD7601BD4F5AC3564567BA13A594207B90EE7F0805B
    C[ 2] = 472C648962292FF2
    X[ 0] = 452821E638D01377BE5466CF34E90C6C
         R = D58E910F62FA39838BE5670C526D728C
    Accumulate entry 2:
    C[ 0] = C9ECEACCC7F4D7B7FEB0B3D2B4EB401F80A12DA721ADCFFAA44B62591213B4FA
    C[ 1] = A93F60632C5B8B9A3316F960D2818385E02EBB31A3D4666DB0D053CE408DABEE
    C[ 2] = 89EFD7A25FA7E791
    X[ 0] = 452821E638D01377BE5466CF34E90C6C
         R = D58E910F62FA39838BE5670C526D728C
accumulate in[0] = C9ECEACCC7F4D7B7FEB0B3D2B4EB401F
accumulate in[1] = 21ADCFFAA44B62591213B4FA80A12DA7
accumulate in[2] = 3316F960D2818385A93F60632C5B8B9A
accumulate in[3] = 408DABEEE02EBB31A3D4666DB0D053CE
    CoreRound entry 3:
    C[ 0] = C9ECEACCC7F4D7B7FEB0B3D2B4EB401F80A12DA721ADCFFAA44B62591213B4FA
    C[ 1] = A93F60632C5B8B9A3316F960D2818385E02EBB31A3D4666DB0D053CE408DABEE
    C[ 2] = 89EFD7A25FA7E791
    X[ 0] = 452821E638D01377BE5466CF34E90C6C
         R = 4E54E6B733EAB4D96DAD662AFAACC760
    Accumulate entry 3:
    C[ 0] = F30879518E9AD487BD9EDBC12BA5AF152BEFE938F49DFBF258F7EF2F3E1AC296
    C[ 1] = C2223430A516D5A4FDB29DBEB58EA3AFB772AB975737680CEB385105B9918037
    C[ 2] = 4997509B7880737A
    X[ 0] = 452821E638D01377BE5466CF34E90C6C
         R = 4E54E6B733EAB4D96DAD662AFAACC760
```

```
accumulate in[0] = F30879518E9AD487BD9EDBC12BA5AF15
accumulate in[1] = F49DFBF258F7EF2F3E1AC2962BEFE938
accumulate in[2] = FDB29DBEB58EA3AFC2223430A516D5A4
accumulate in[3] = B9918037B772AB975737680CEB385105
   CoreRound entry 4:
   C[ 0] = F30879518E9AD487BD9EDBC12BA5AF152BEFE938F49DFBF258F7EF2F3E1AC296
   C[ 1] = C2223430A516D5A4FDB29DBEB58EA3AFB772AB975737680CEB385105B9918037
   C[ 2] = 4997509B7880737A
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 0DE2799DE77B87497B3C2341B4C805EC
   Accumulate entry 4:
   C[ 0] = 2DE08E5C4C887ECE4AD4BCA2E84A4024D80347330D23386BBCCA23E9B4DD71E6
   C[ 1] = 874C1F5E51C892FFA12D4B39B558200DA515AAC4E39DB5D077DE53BE8803CADB
   C[ 2] = 073EA2AB914E9FFE
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 0DE2799DE77B87497B3C2341B4C805EC
accumulate in[0] = 2DE08E5C4C887ECE4AD4BCA2E84A4024
accumulate in[1] = 0D23386BBCCA23E9B4DD71E6D8034733
accumulate in[2] = A12D4B39B558200D874C1F5E51C892FF
accumulate in[3] = 8803CADBA515AAC4E39DB5D077DE53BE
   CoreRound entry 5:
   C[ 0] = 2DE08E5C4C887ECE4AD4BCA2E84A4024D80347330D23386BBCCA23E9B4DD71E6
   C[ 1] = 874C1F5E51C892FFA12D4B39B558200DA515AAC4E39DB5D077DE53BE8803CADB
   C[ 2] = 073EA2AB914E9FFE
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 040F4E48077450A7E1E4448BA297C3BA
   Accumulate entry 5:
   C[ 0] = E3B788755C0BB5B3C56088B26B251AA14640E1C158CDD69F7182070D78E09CB7
   C[ 1] = 481C24BCA6DABAF85AB7B237B9B282EFDD896B7EE7C3A7BDB98B802EADE8FA2B
   C[ 2] = 3031B80FFC19770C
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 040F4E48077450A7E1E4448BA297C3BA
accumulate in[0] = E3B788755C0BB5B3C56088B26B251AA1
accumulate in[1] = 58CDD69F7182070D78E09CB74640E1C1
accumulate in[2] = 5AB7B237B9B282EF481C24BCA6DABAF8
accumulate in[3] = ADE8FA2BDD896B7EE7C3A7BDB98B802E
   CoreRound entry 6:
   C[ 0] = E3B788755C0BB5B3C56088B26B251AA14640E1C158CDD69F7182070D78E09CB7
   C[ 1] = 481C24BCA6DABAF85AB7B237B9B282EFDD896B7EE7C3A7BDB98B802EADE8FA2B
   C[ 2] = 3031B80FFC19770C
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 482A58BE4EC60B88F3BBD38F90A3020C
   Accumulate entry 6:
   C[ 0] = 5E7ED0BEB1AC1FAB910A606D7B47401539E9CE0DE5B297394799F94234699994
   C[ 1] = 30A917A4F5A8D3383BD49E69826317352879F18795B002FBB4D13AC1C3816FB5
   C[ 2] = 614D7DD78FEF152E
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 482A58BE4EC60B88F3BBD38F90A3020C
accumulate in[0] = 5E7ED0BEB1AC1FAB910A606D7B474015
accumulate in[1] = E5B297394799F9423469999439E9CE0D
accumulate in[2] = 3BD49E698263173530A917A4F5A8D338
accumulate in[3] = C3816FB52879F18795B002FBB4D13AC1
   CoreRound entry 7:
   C[ 0] = 5E7ED0BEB1AC1FAB910A606D7B47401539E9CE0DE5B297394799F94234699994
   C[ 1] = 30A917A4F5A8D3383BD49E69826317352879F18795B002FBB4D13AC1C3816FB5
   C[ 2] = 614D7DD78FEF152E
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 0BB3EEE512E90BD3F3C13F29937465ED
   Accumulate entry 7:
   C[ 0] = 1FF9BEB938CAE5D087135CFD79186C9324AEA8D878BA5386988BBA42555756E7
   C[ 1] = 3DAA26E2D019A03A0D1955862E2BEB6B34ED10389A7F7CE1B36849879E2B94C3
   C[ 2] = 85D29DC280385E08
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
```

```
      R = 0BB3EEE512E90BD3F3C13F29937465ED
accumulate in[0] = 1FF9BEB938CAE5D087135CFD79186C93
accumulate in[1] = 78BA5386988BBA42555756E724AEA8D8
accumulate in[2] = 0D1955862E2BEB6B3DAA26E2D019A03A
accumulate in[3] = 9E2B94C334ED10389A7F7CE1B3684987
   F/G entry 3 (G):
   C[ 0] = 1FF9BEB938CAE5D087135CFD79186C9324AEA8D878BA5386988BBA42555756E7
   C[ 1] = 3DAA26E2D019A03A0D1955862E2BEB6B34ED10389A7F7CE1B36849879E2B94C3
   C[ 2] = 85D29DC280385E08
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
      R = FFC2C29FA86EAF1286506F30ADB3481B
   CoreRound entry 0:
   C[ 0] = 1FF9BEB938CAE5D087135CFD79186C9324AEA8D878BA5386988BBA42555756E7
   C[ 1] = 3DAA26E2D019A03A0D1955862E2BEB6B34ED10389A7F7CE1B36849879E2B94C3
   C[ 2] = 85D29DC280385E08
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
      R = 00000000000000000000000000000000
   Accumulate entry 0:
   C[ 0] = 80D6A7FFB599D459E95801C06E849DA7DB24D1582211D46ED996CB899E5FE605
   C[ 1] = 51208900627C35DAF6C4F38CB8BD6872F27A0BBFAFA61B4E798BB29050437A92
   C[ 2] = 974D0775BD7B2BAD
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
      R = 00000000000000000000000000000000
accumulate in[0] = 80D6A7FFB599D459E95801C06E849DA7
accumulate in[1] = 2211D46ED996CB899E5FE605DB24D158
accumulate in[2] = F6C4F38CB8BD687251208900627C35DA
accumulate in[3] = 50437A92F27A0BBFAFA61B4E798BB290
   CoreRound entry 1:
   C[ 0] = 80D6A7FFB599D459E95801C06E849DA7DB24D1582211D46ED996CB899E5FE605
   C[ 1] = 51208900627C35DAF6C4F38CB8BD6872F27A0BBFAFA61B4E798BB29050437A92
   C[ 2] = 974D0775BD7B2BAD
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
      R = 0440FA8F26C87C1D8981758BAE57CBB5
   Accumulate entry 1:
   C[ 0] = 74910812363D2DFE9A38CE614E74CE3EBCE292D8808D5BDD7948314F4210A889
   C[ 1] = 0DEC0B97FECB84B57DBB8841BE775F8A14C09ABD206E5C455CD7A4F62A0D8158
   C[ 2] = B6DC13F68363D7C6
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
      R = 0440FA8F26C87C1D8981758BAE57CBB5
accumulate in[0] = 74910812363D2DFE9A38CE614E74CE3E
accumulate in[1] = 808D5BDD7948314F4210A889BCE292D8
accumulate in[2] = 7DBB8841BE775F8A0DEC0B97FECB84B5
accumulate in[3] = 2A0D815814C09ABD206E5C455CD7A4F6
   CoreRound entry 2:
   C[ 0] = 74910812363D2DFE9A38CE614E74CE3EBCE292D8808D5BDD7948314F4210A889
   C[ 1] = 0DEC0B97FECB84B57DBB8841BE775F8A14C09ABD206E5C455CD7A4F62A0D8158
   C[ 2] = B6DC13F68363D7C6
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
      R = A7EAA059C30AA59B7C2B44B1FEDDB710
   Accumulate entry 2:
   C[ 0] = 3A9D19461D41FF03B69136F1D0B4ABCC0683EDFB6C2BC899EA7E045695A5BA8A
   C[ 1] = 6CE7B89A9B65AA2D0E2438B9BB982B39462A54499ADAD08328097BA6474BC576
   C[ 2] = 30DE992A2AF79A7D
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
      R = A7EAA059C30AA59B7C2B44B1FEDDB710
accumulate in[0] = 3A9D19461D41FF03B69136F1D0B4ABCC
accumulate in[1] = 6C2BC899EA7E045695A5BA8A0683EDFB
accumulate in[2] = 0E2438B9BB982B396CE7B89A9B65AA2D
accumulate in[3] = 474BC576462A54499ADAD08328097BA6
   CoreRound entry 3:
   C[ 0] = 3A9D19461D41FF03B69136F1D0B4ABCC0683EDFB6C2BC899EA7E045695A5BA8A
   C[ 1] = 6CE7B89A9B65AA2D0E2438B9BB982B39462A54499ADAD08328097BA6474BC576
   C[ 2] = 30DE992A2AF79A7D
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = B8338C49C98721BEA922A0D39B8620AC
   Accumulate entry 3:
   C[ 0] = D29076BA3775B54BB1C3BD100515D86A753E4C429A6D5E7F7D05AED8624217DF
   C[ 1] = 74248F3E9D16D08A68996CA558E75E71F398F7C75B4D7531FA7E25DAB5B88E29
   C[ 2] = FCFD125A1F5A1530
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = B8338C49C98721BEA922A0D39B8620AC
accumulate in[0] = D29076BA3775B54BB1C3BD100515D86A
accumulate in[1] = 9A6D5E7F7D05AED8624217DF753E4C42
accumulate in[2] = 68996CA558E75E7174248F3E9D16D08A
accumulate in[3] = B5B88E29F398F7C75B4D7531FA7E25DA
   CoreRound entry 4:
   C[ 0] = D29076BA3775B54BB1C3BD100515D86A753E4C429A6D5E7F7D05AED8624217DF
   C[ 1] = 74248F3E9D16D08A68996CA558E75E71F398F7C75B4D7531FA7E25DAB5B88E29
   C[ 2] = FCFD125A1F5A1530
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 2DEF46002888939B55CAF0138CC541D4
   Accumulate entry 4:
   C[ 0] = 0E28C5FF1FCA5C9CEF9676F1DE8C5A51EEAA28EF20CF966DF79B90002B48CDD1
   C[ 1] = 05FFE3F34F449D258913EEFE0F5B9A9330867F78AFBBDD80274C71C01941E85D
   C[ 2] = 91B23BD861FB3C40
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 2DEF46002888939B55CAF0138CC541D4
accumulate in[0] = 0E28C5FF1FCA5C9CEF9676F1DE8C5A51
accumulate in[1] = 20CF966DF79B90002B48CDD1EEAA28EF
accumulate in[2] = 8913EEFE0F5B9A9305FFE3F34F449D25
accumulate in[3] = 1941E85D30867F78AFBBDD80274C71C0
   CoreRound entry 5:
   C[ 0] = 0E28C5FF1FCA5C9CEF9676F1DE8C5A51EEAA28EF20CF966DF79B90002B48CDD1
   C[ 1] = 05FFE3F34F449D258913EEFE0F5B9A9330867F78AFBBDD80274C71C01941E85D
   C[ 2] = 91B23BD861FB3C40
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 935A1331FF04BAEC3B507540D4EBDF8F
   Accumulate entry 5:
   C[ 0] = 9ADBED931D9BC9EC83A60F4E9B956E2EBBB446C84D6359ADA145E28C9616BF23
   C[ 1] = 36D519BC7C32102EC2820A7F01E25E2661AE94BEF369AD5A58D4613D9A566483
   C[ 2] = 5451809CD7113E50
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 935A1331FF04BAEC3B507540D4EBDF8F
accumulate in[0] = 9ADBED931D9BC9EC83A60F4E9B956E2E
accumulate in[1] = 4D6359ADA145E28C9616BF23BBB446C8
accumulate in[2] = C2820A7F01E25E2636D519BC7C32102E
accumulate in[3] = 9A56648361AE94BEF369AD5A58D4613D
   CoreRound entry 6:
   C[ 0] = 9ADBED931D9BC9EC83A60F4E9B956E2EBBB446C84D6359ADA145E28C9616BF23
   C[ 1] = 36D519BC7C32102EC2820A7F01E25E2661AE94BEF369AD5A58D4613D9A566483
   C[ 2] = 5451809CD7113E50
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 1C36C9F323965B14EB5C71CBD02C867A
   Accumulate entry 6:
   C[ 0] = 9EDA55E88EBB24F99014BF0A9C28A2AFC144AD2F756DF905738B91875A1B146F
   C[ 1] = 7A8DF3BE75AC26433496F0B0B1EA61C656ED8D73E66B6068930CA76D30941238
   C[ 2] = 77C1A182B34BFFF6
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 1C36C9F323965B14EB5C71CBD02C867A
accumulate in[0] = 9EDA55E88EBB24F99014BF0A9C28A2AF
accumulate in[1] = 756DF905738B91875A1B146FC144AD2F
accumulate in[2] = 3496F0B0B1EA61C67A8DF3BE75AC2643
accumulate in[3] = 3094123856ED8D73E66B6068930CA76D
   CoreRound entry 7:
   C[ 0] = 9EDA55E88EBB24F99014BF0A9C28A2AFC144AD2F756DF905738B91875A1B146F
   C[ 1] = 7A8DF3BE75AC26433496F0B0B1EA61C656ED8D73E66B6068930CA76D30941238
   C[ 2] = 77C1A182B34BFFF6
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = F383879639A102DFBDB549786BE008D4
   Accumulate entry 7:
   C[ 0] = 0BB63BC606D91A7FA883DB0B43534BFA8ABFA7149BF78EB150E9A5EC31EF18F8
   C[ 1] = 5801EA53292F0D170469A39CEF3CEE682EDFA271BFDC3D00295B7DFC5F9B7068
   C[ 2] = F52851070885013F
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = F383879639A102DFBDB549786BE008D4
accumulate in[0] = 0BB63BC606D91A7FA883DB0B43534BFA
accumulate in[1] = 9BF78EB150E9A5EC31EF18F88ABFA714
accumulate in[2] = 0469A39CEF3CEE685801EA53292F0D17
accumulate in[3] = 5F9B70682EDFA271BFDC3D00295B7DFC
   Final state:
   C[ 0] = 0BB63BC606D91A7FA883DB0B43534BFA8ABFA7149BF78EB150E9A5EC31EF18F8
   C[ 1] = 5801EA53292F0D170469A39CEF3CEE682EDFA271BFDC3D00295B7DFC5F9B7068
   C[ 2] = F52851070885013F
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 3830E115AE72F155C3045DD8A27894D1
   Digest: AC99FD9156D4C6FC613A85CBBFD283FC7214792B3E786E34B33D368020F79BF6FFC2C29FA86EAF12
     ↪ 86506F30ADB3481B3830E115AE72F155C3045DD8A27894D1
```

Listing 32: Less detailed test vector

```
Hashing 8 bytes message: 0001020304050607
   Padded Message:  00010203040506070100000000000000
   F/G entry 0 (F with DS): padded=1, domain=2, finalize=1
   C[ 0] = 243F6A8885A308D313198A2E03707344A4093822299F31D0082EFA98EC4E6C89
   C[ 1] = 243F6A8885A308D313198A2E03707344A4093822299F31D0082EFA98EC4E6C89
   C[ 2] = 243F6A8885A308D3
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 00000000000000000000000000000000
       I = 00010203040506070100000000000000
   F/G entry 1 (G):
   C[ 0] = AFD67766FE95CC59D97A246E9D43DE4F23D2712D76C6A6DAEC77C894927DAA25
   C[ 1] = 93762763608B49AD3EDFA778C88413A863F0BDA9B4DF4C7E1CB4517392B80421
   C[ 2] = 12ACD2284F4D4232
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = AC99FD9156D4C6FC613A85CBBFD283FC
   F/G entry 2 (G):
   C[ 0] = 66FBC1A8A711CF42AF215D7684CA16196CD43758B20FAEE4B0A4AED4F3AACCDD
   C[ 1] = 998E170587177E369981CFC80FE21861812A09902BA8B555AFBE8D1C2C57A3B5
   C[ 2] = F2F9242D1472A76D
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 7214792B3E786E34B33D368020F79BF6
   F/G entry 3 (G):
   C[ 0] = 1FF9BEB938CAE5D087135CFD79186C9324AEA8D878BA5386988BBA42555756E7
   C[ 1] = 3DAA26E2D019A03A0D1955862E2BEB6B34ED10389A7F7CE1B36849879E2B94C3
   C[ 2] = 85D29DC280385E08
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = FFC2C29FA86EAF1286506F30ADB3481B
   Final state:
   C[ 0] = 0BB63BC606D91A7FA883DB0B43534BFA8ABFA7149BF78EB150E9A5EC31EF18F8
   C[ 1] = 5801EA53292F0D170469A39CEF3CEE682EDFA271BFDC3D00295B7DFC5F9B7068
   C[ 2] = F52851070885013F
   X[ 0] = 452821E638D01377BE5466CF34E90C6C
       R = 3830E115AE72F155C3045DD8A27894D1
   Digest: AC99FD9156D4C6FC613A85CBBFD283FC7214792B3E786E34B33D368020F79BF6FFC2C29FA86EAF12
     ↪ 86506F30ADB3481B3830E115AE72F155C3045DD8A27894D1
```

Listing 33: Test vectors

```
Hashing 0 bytes message:
   Padded Message:  01000000000000000000000000000000
   Digest: 6896590A319FDE1F3B18EBAE1DF1E5E8FB0756A878EE9E2165B085FF3AED6805F8F73D5714C75960
     ↪ A6A8095DAE5EF9C00D3F055490D4CF45D4A26B37FD7B5441
Hashing 1 bytes message: 00
   Padded Message:  00010000000000000000000000000000
   Digest: DAB16B97C37160586B647B0DCA6897943654803 24E539CD63F87B119B0C46668DCDE5163A170E06
     ↪ DA9361B05F7CE7645EF68BDC99B3B813B8B1583C5C62D4E4A
Hashing 2 bytes message: 0001
   Padded Message:  00010100000000000000000000000000
   Digest: D1982BC43D8C42DCD94C1C7E9611951374DC8BF5E6FC407E8A8DC423F4F0F45909A4AEAA1000B35A
     ↪ 8081862E797508807E8763F611AEF1D3C06ECAEDB5229980
Hashing 3 bytes message: 000102
   Padded Message:  00010201000000000000000000000000
   Digest: 79E24057775CB0DC401185CC7BE13EC58CE52748DA5627A0C1E1CE16A84BF7CEF149C6CE09671772
     ↪ CA2927EF956A63C3C28EF1A3CDFCDC3449BA92F870AA1C47
Hashing 4 bytes message: 00010203
   Padded Message:  00010203010000000000000000000000
   Digest: 3C4C288299FD3986FB213B945ADDB70F26EDEA09FA4291CF42467355DA09FDD7E69A7B306636DAC0
     ↪ 78EE81643A19F5126EA71DBC4032BE2320B8382119238D5B
Hashing 5 bytes message: 0001020304
   Padded Message:  00010203040100000000000000000000
   Digest: 0E03800EBCD2F207C1D662425E116450F214DD3EE3F3718C8BB3A3F498A77FD2985271E57D6A9432
     ↪ 0141255DFD64DBB1386033EF4E312F6349F7C04A67D58E32
Hashing 6 bytes message: 000102030405
   Padded Message:  00010203040501000000000000000000
   Digest: 5273531B1C705DA74D0D6ABD098CDFDBDA7B501CE2951B5B1C40BCC607D434F4AF35EB9E8821C9B5
     ↪ 27C0132D54831928BE55C1592C1B454033646B238BE54A89
Hashing 7 bytes message: 00010203040506
   Padded Message:  00010203040506010000000000000000
   Digest: E989422BEAFB61CEBE8905D1B23A24227AABFEA39466DD12A9D29F88511B35E896075D1307995EF8
     ↪ 07FBF84BB24318F000911AC01DCEA33349A0A1B215D26618
Hashing 8 bytes message: 0001020304050607
   Padded Message:  00010203040506070100000000000000
   Digest: AC99FD9156D4C6FC613A85CBBFD283FC7214792B3E786E34B33D368020F79BF6FFC2C29FA86EAF12
     ↪ 86506F30ADB3481B3830E115AE72F155C3045DD8A27894D1
Hashing 9 bytes message: 000102030405060708
   Padded Message:  00010203040506070801000000000000
   Digest: 256B688F8EA525B1F04933E42C2044123BBE9B006AA0A716E4CD804359C61F686162DAFC0CE50CD1
     ↪ 105E84ECE9A2066583A16954432296D22F98A894C180758F
Hashing 10 bytes message: 00010203040506070809
   Padded Message:  00010203040506070809010000000000
   Digest: FD4C5AF23E347BF0A113E4B3F7D2C61968567EDD0958880F86F79C4E443E6A7FEB79166E8D7ADF16
     ↪ 67ADB8E1387F18C0CFEF0EBA7D292F03D5902939C73EDB85
Hashing 11 bytes message: 000102030405060708090A
   Padded Message:  000102030405060708090A0100000000
   Digest: 9777E2A2FFEDEDD2CA5F673F1BA2F6739FCEDC157F76B00305FE642E926640CCD4C04C02BA9F3C26
     ↪ BEC45001A89631F6394F8AF17E7CD1B1DE6E47C8D3DF2B20
Hashing 12 bytes message: 000102030405060708090A0B
   Padded Message:  000102030405060708090A0B01000000
   Digest: 09941CBA60C7DB6FDC165D2BEAFE6140ED5BE657B7B506F4A09D2D571122C64A7F9D96B7C89AF522
     ↪ 73B7F7A0C15D542E2188EAAA72A53214DCAB8AE9DF48DC76
Hashing 13 bytes message: 000102030405060708090A0B0C
   Padded Message:  000102030405060708090A0B0C010000
   Digest: 337E62424FE04893D2AF9C2B741F7A9665F14FEAB1E901BDBD5178C37A1D80F4459827F63C34B59E
     ↪ 22CE95A73548F911B78D4535EF69B46FCCC1F9E865DF99AA
Hashing 14 bytes message: 000102030405060708090A0B0C0D
   Padded Message:  000102030405060708090A0B0C0D0100
   Digest: E12B53E381B1BB5014409CFFB107ACA484234B090E51381415922CA9B7782C93F23E2AFCF0CE81
     ↪ DEFC23D33904125EE801E81D5649B675DA986411BD5B3112AA
Hashing 15 bytes message: 000102030405060708090A0B0C0D0E
```

```
   Padded Message:  000102030405060708090A0B0C0D0E01
   Digest: 3E40E4E0617FA663A73AC47E084C22982BC2CCC890B1C8AAF4BBF524DA3AAD18310553D731B4B7BE
     ↪ 24981689F1C91FC4E28C7A36186A32C5273BF213E99A3F75
Hashing 16 bytes message: 000102030405060708090A0B0C0D0E0F
   Digest: E743DE651072AE1A078D201373BC383FFAE607545308D268AC663B0B680FEE8BD0D053EA40A55C5
     ↪ DD2AEE281C1CBFFA79152ACC9BD5705F3FB4DAF415458CA12
Hashing 17 bytes message: 000102030405060708090A0B0C0D0E0F10
   Padded Message:  000102030405060708090A0B0C0D0E0F1001000000000000000000000000000
   Digest: 9E8CF4C968DB79AAFEC63811AFEE47B26F534C29EE2EAE75F0DA67840D616EC99831224A4A775F20
     ↪ CBF6EBB6DD58646A321FB588ADD0DA1294A4F360DDE539ED
Hashing 18 bytes message: 000102030405060708090A0B0C0D0E0F1011
   Padded Message:  000102030405060708090A0B0C0D0E0F1011010000000000000000000000000
   Digest: 629706AD6F840D1D6AFBE54E9982124CE4EFAE0C05031CD70B16912A6B814A4AE718FB061057E418
     ↪ 229B01F43F77757E883917606735CC78613255CA4AEBAE78
Hashing 19 bytes message: 000102030405060708090A0B0C0D0E0F101112
   Padded Message:  000102030405060708090A0B0C0D0E0F1011120100000000000000000000000
   Digest: 2DAD2C34486A69BE5E13F1CD896921D45D130D457B79D14B558021558AFDBB1744D0382442B77C16
     ↪ D0E9F7C3CFA113AAE185EC62815D7FA998ADAB4E4DE06463
Hashing 20 bytes message: 000102030405060708090A0B0C0D0E0F10111213
   Padded Message:  000102030405060708090A0B0C0D0E0F1011121301000000000000000000000
   Digest: E04222D814CFAED61001FEA219A886E6715362C401ED2766FC78899E5C91F7FB61DBCBA92937EFE2
     ↪ C21C7BEB713DD53C10C5396F373F8C59F4169C67106A7542
Hashing 21 bytes message: 000102030405060708090A0B0C0D0E0F1011121314
   Padded Message:  000102030405060708090A0B0C0D0E0F1011121314010000000000000000000
   Digest: D35BD6A740909294FD0E0C529B5767A2899C3089662807664CF76E3EA0B2FECEBC2756334150DD39
     ↪ 869378F48DB14F730A821EDE079C3CE6689EB7AF555BAB9F
Hashing 22 bytes message: 000102030405060708090A0B0C0D0E0F101112131415
   Padded Message:  000102030405060708090A0B0C0D0E0F1011121314150100000000000000000
   Digest: AB2997609E2BDE9AC8E5C2F71577E8D408D102C79229F8C4AF726FABCF4729FBF0007530D2EF07D3
     ↪ 3496A69EBE5AD64DB8B45B9A90CD98773139CC3EAED90195
Hashing 23 bytes message: 000102030405060708090A0B0C0D0E0F10111213141516
   Padded Message:  000102030405060708090A0B0C0D0E0F1011121314151601000000000000000
   Digest: 526A7ED6CBF7168A1FFCA2EE05B3FDCB6DBA95732231438621C748DFA35814C182298A988CF04CE5
     ↪ 4C00C77DE01362497942005486 9ADB12283B9FED13177797
Hashing 24 bytes message: 000102030405060708090A0B0C0D0E0F1011121314151617
   Padded Message:  000102030405060708090A0B0C0D0E0F1011121314151617010000000000000
   Digest: CC7B7615BA0F2348F40417BE5AAAEA59104746BF0E97775F72A47A904684401BB6C4A37E01165CCB
     ↪ 1270FB8E2BFE835D351AE9CCFDB911C3BF377C9A8310D502
Hashing 25 bytes message: 000102030405060708090A0B0C0D0E0F101112131415161718
   Padded Message:  000102030405060708090A0B0C0D0E0F1011121314151617180100000000000
   Digest: 4F67784DDC3588395304D377CA4EBAB7720F9A1523EDA512AA331234B899E1A601527C2799B164A5
     ↪ 6771C3EBD0C8D2E2EEBDD988B3BAA8A86CEE260AC54F0A42
Hashing 26 bytes message: 000102030405060708090A0B0C0D0E0F10111213141516171819
   Padded Message:  000102030405060708090A0B0C0D0E0F1011121314151617181901000000000
   Digest: 8F322FFFE6449EFB29FEDB30FB6CC517BF9CA2FA97BC10C198F5D27B0320CA08B0915597C33BB77D
     ↪ 33F5EA8671A6ECF3FCA97DC2C0FC2C8163087F8CC1287F58
Hashing 27 bytes message: 000102030405060708090A0B0C0D0E0F101112131415161718191A
   Padded Message:  000102030405060708090A0B0C0D0E0F101112131415161718191A010000000
   Digest: ADB4B9BD0E8E517FCB064E04E4098ABFBC4D018C3E33DE7A15306937E644519AE02686C00C2E919C
     ↪ 0D3CB05048F0B6796A9E72379D15245FD3BB706C2FAC92C3
Hashing 28 bytes message: 000102030405060708090A0B0C0D0E0F101112131415161718191A1B
   Padded Message:  000102030405060708090A0B0C0D0E0F101112131415161718191A1B01000000
   Digest: 16CE6D71B868A814616FCA42D66F8B1B1103574E4D68CB10EDF5BFDB2C9F5FDC23B878F4376DDF60
     ↪ 776E107DE4D72EE7077A7E698D3AC80D5F8DE219ABB6B8B3
Hashing 29 bytes message: 000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C
   Padded Message:  000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C010000
   Digest: 34CEE3BDEA0C2B2D8498B5A3B9A8BF2EDE84ABD01B12B7669846B03875AB6CC2DD71D42468A8EBCE
     ↪ 9BF715788F982548B97026AE331C919C56CCA3F73DF65A31
Hashing 30 bytes message: 000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D
   Padded Message:  000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D0100
   Digest: 53F880F1B4531DC3F886C6B6BD5D8044D56CC7ADAFFEAAC41A6B27533DEB9D6B56D6D5F80DE30E4C
     ↪ 594537CDBFA5403F82F784A634B9D265EE665646D069DD6E
Hashing 31 bytes message: 000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E
```

```
   Padded Message:  000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E01
   Digest: 92CB95C38582CE60FEECA5CA41A92A179EF70622EC03DAD7A5ED56DD48D24BC76CE9EABD898B9F67
     ↪ F6E8B5B57FBD82C1946DB2206AB47730CBA0B9C7E0C09784
Hashing 32 bytes message: 000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
   Digest: 2B6750959062AAB4545949BA67D64A921014A75E86FD413FB715817DA54363710083A71D1E7C63
     ↪ ADB50101A7D418D77AAF3AC16BCC4FED75B2D5DDA570A3A814
```

Listing 34: Iterative test vector

```
Hashing null message 100 times
   Digest: A17875C70ED5189609380EF437576D5D646708CE7ADD5C5F8A5469F10C8BC05916524B333FA95ADC
     ↪ 687F6951FAE5143179B8F408C3C6D42E223F2D480E026859
```