ॐ भूर्भुवस्व तत् सवितूर्वरेण्यम् भर्गो देवस्य धीमहि धियो यो न प्रचोदयात्

EnRUPT

Sean O'Neil
VEST Corporation

```
#define w 64 // or 32
#define s H  // or 8
unsigned int##w r, d[2], x[H], f, i;

for (i=0; i<2*s; i++, r++)
  x_{r+2}^=f=rotr(2*x_{r^1}^x_{r+4}^d_r^r,w/4)*9, d_r^=f^x_r;

d_1^=input_word;
```

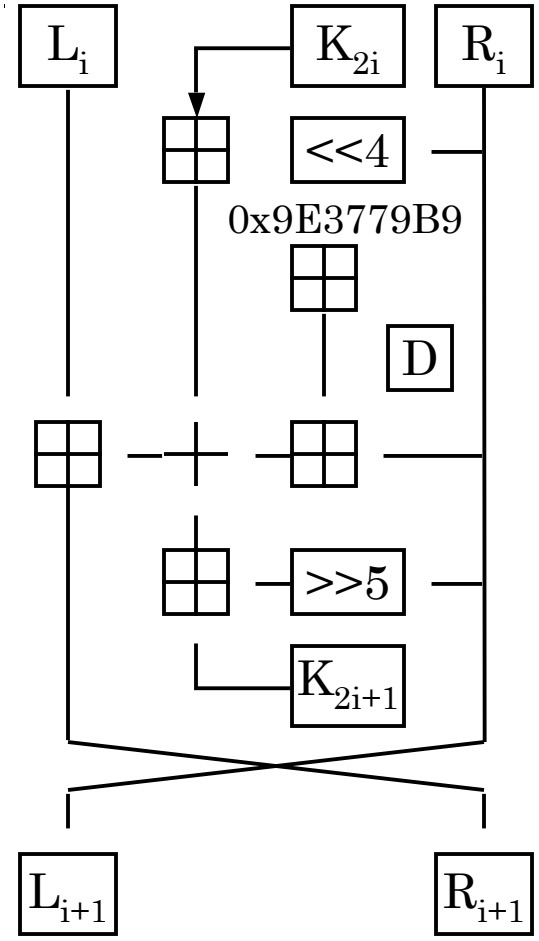
Key Points

1. EnRUPT is the simplest of the SHA-3 submissions.
2. EnRUPT/H is possibly the most area efficient submission.
3. EnRUPT/H is one of the fastest submissions at 10-20 CPB.
4. Stream hashing offers variety. No block chaining required.
5. EnRUPT was submitted with a tunable security parameter.
6. The published preimage attack with 2^{960} time*memory complexity does not invalidate EnRUPT security claims.
7. Collisions were found for ïrRUPT/4 (EnRUPT with s=4).
8. The same attack does not apply to ïrRUPT/5.
9. If allowed, we recommend tuning the security parameter up to s=8 or up to s=H for higher ("provable") security.

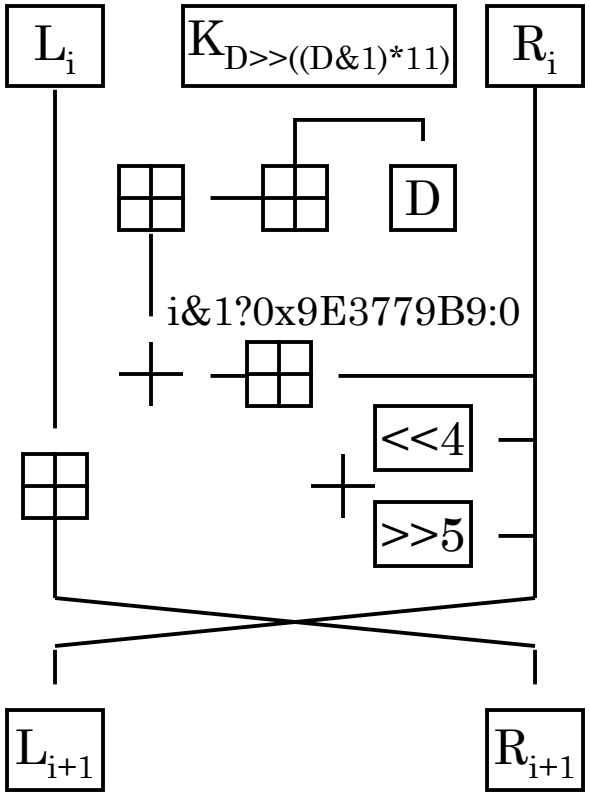
Primary Design Goals

1. Simplicity of every aspect = Kerkhoff #6
2. Scalability = Variable state and word size
3. Flexibility = Stream cipher / hash
4. Error-proof = Easy to implement & debug

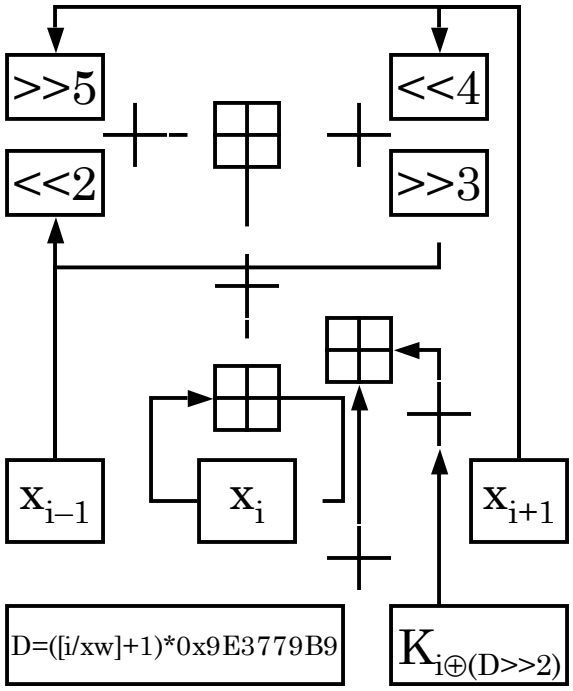
ADD-XOR-ROL Family



TEA: 10/∞



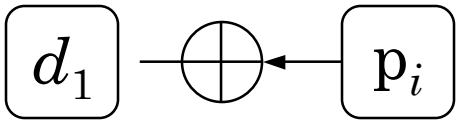
XTEA: 9/19

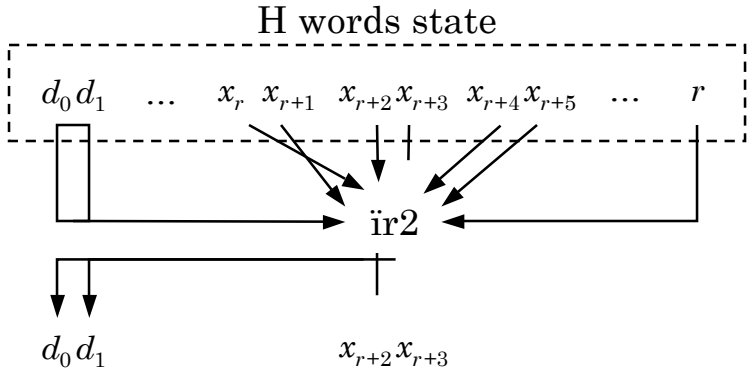


XXTEA: 10/21

Encrypts blocks of any size
No serious attacks
=> A good starting point…

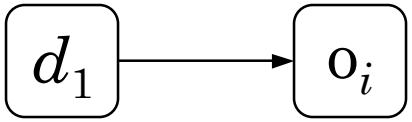
EnRUPTx2 in stream modes





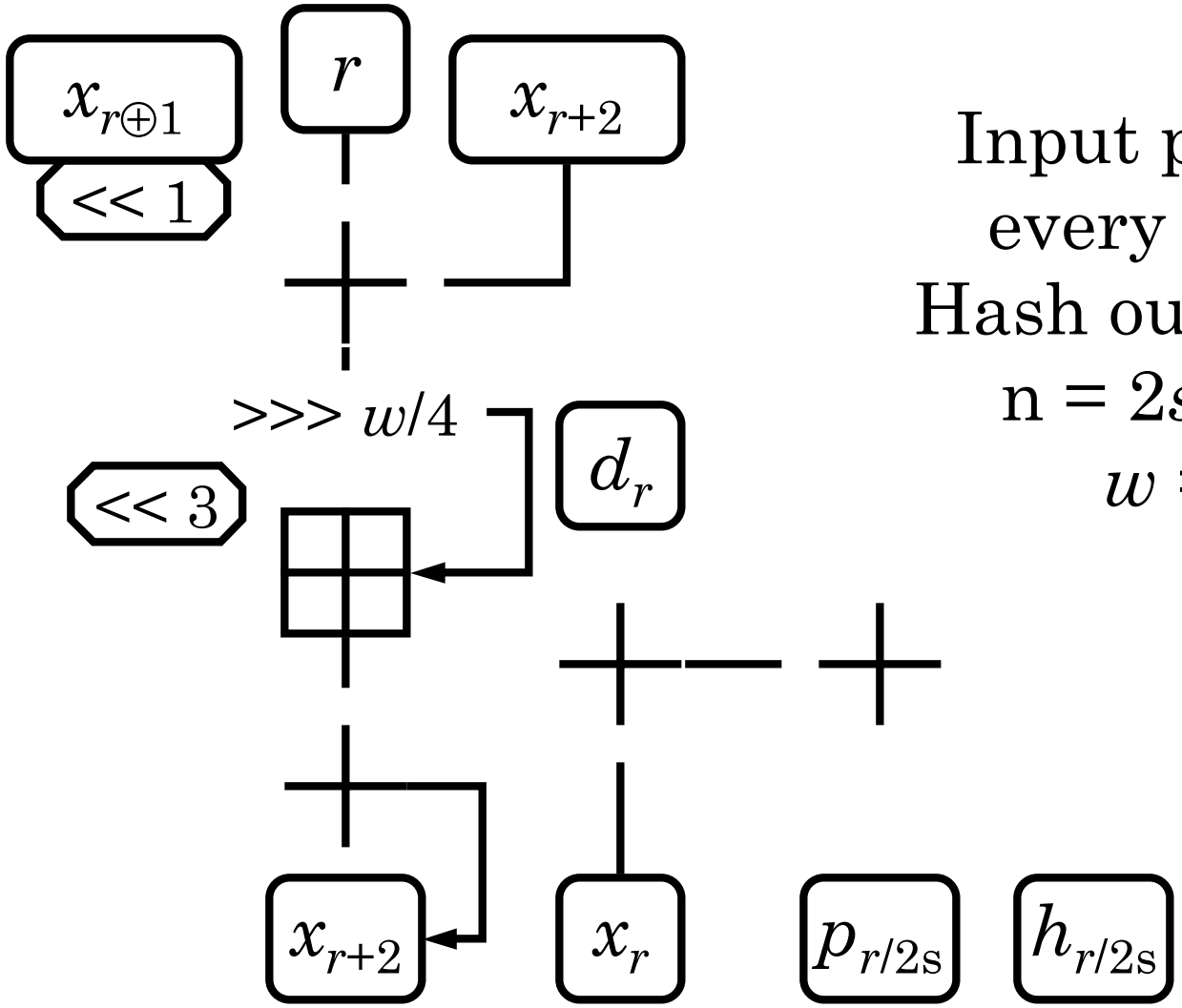
Loading

Mixing



Output

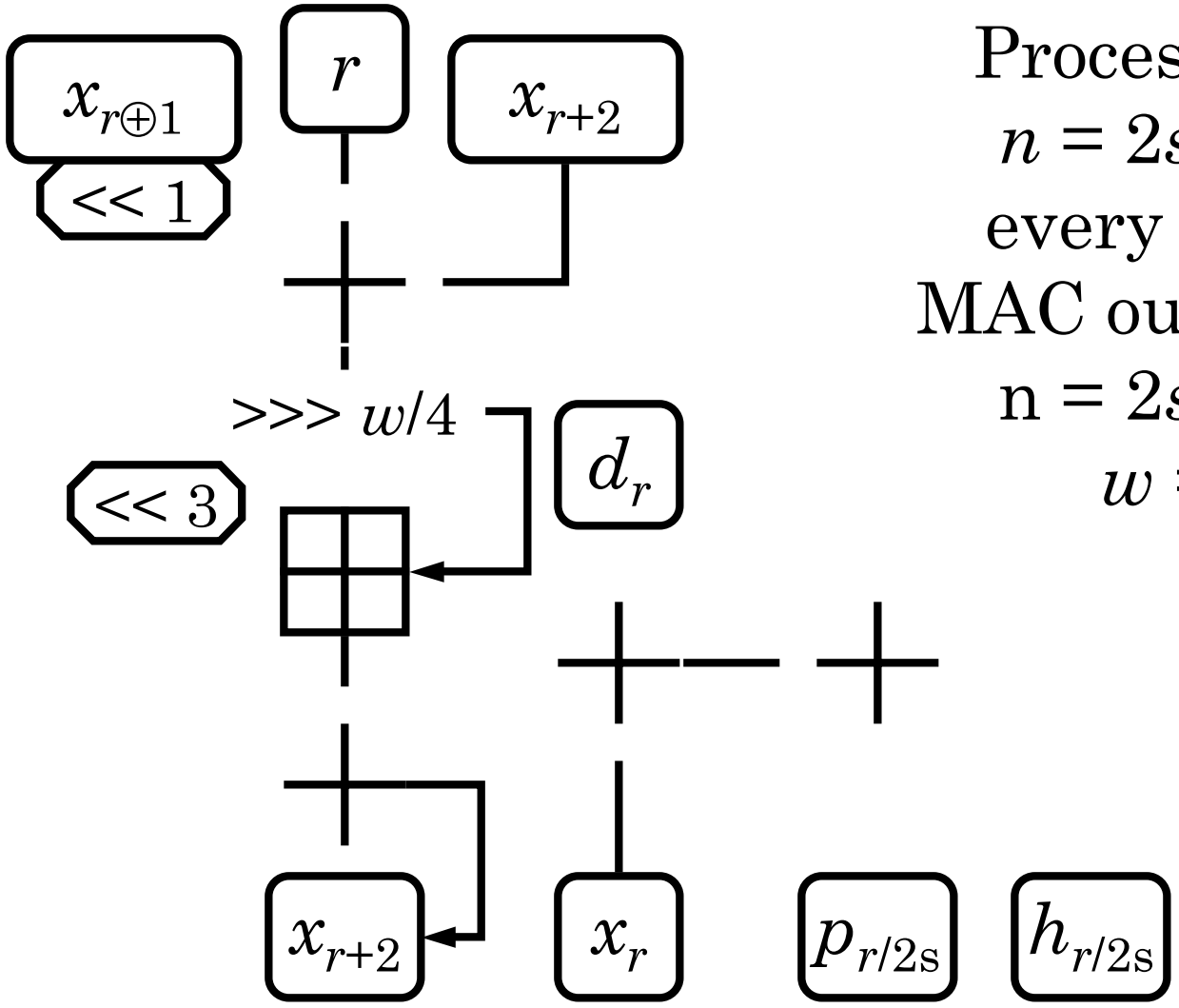
irRUPTx2: Hash



Input processing
every $2s$ rounds
Hash output after
$n = 2sH$ rounds
$w = 32$ or 64
$s = 8$ or H

```
x[r+2]^=f=rotr(2*x[r^1]^x[r+4]^d[r&1]^r,8)*9;     (d[1]^=p^f^x[r]);
```

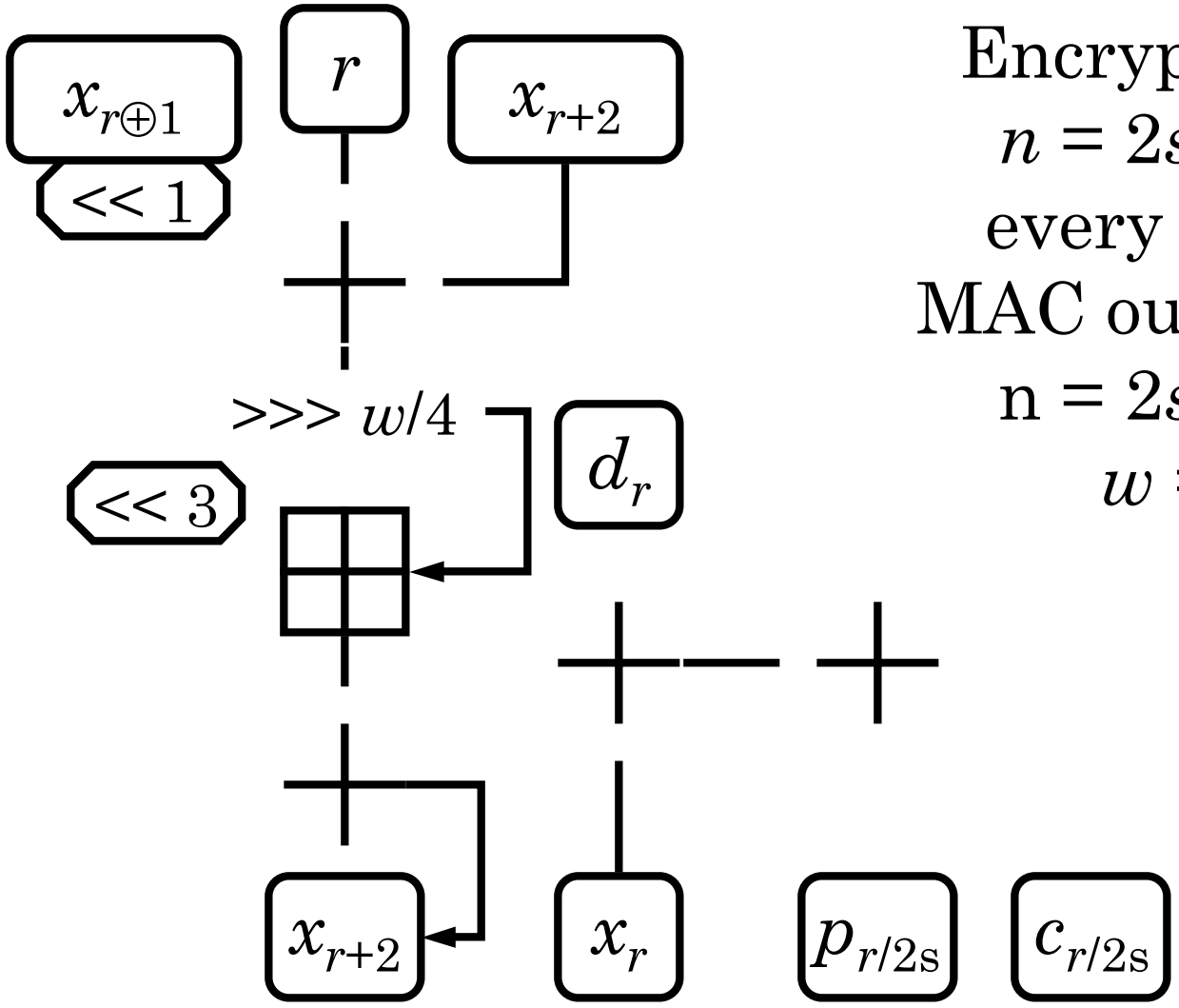
mcRUPTx2: MAC, HMAC



Processing after
$n = 2s\text{H}$ rounds
every $2s$ rounds
MAC output after
$\text{n} = 2s\text{H}$ rounds
$w = 32$ or 64
$s = 8$ or H

```
x_{r+2}^=f=rotr(2*x_{r^1}^x_{r+4}^d_{r&1}^r,8)*9;     (d_1^=p^f^x_r);
```

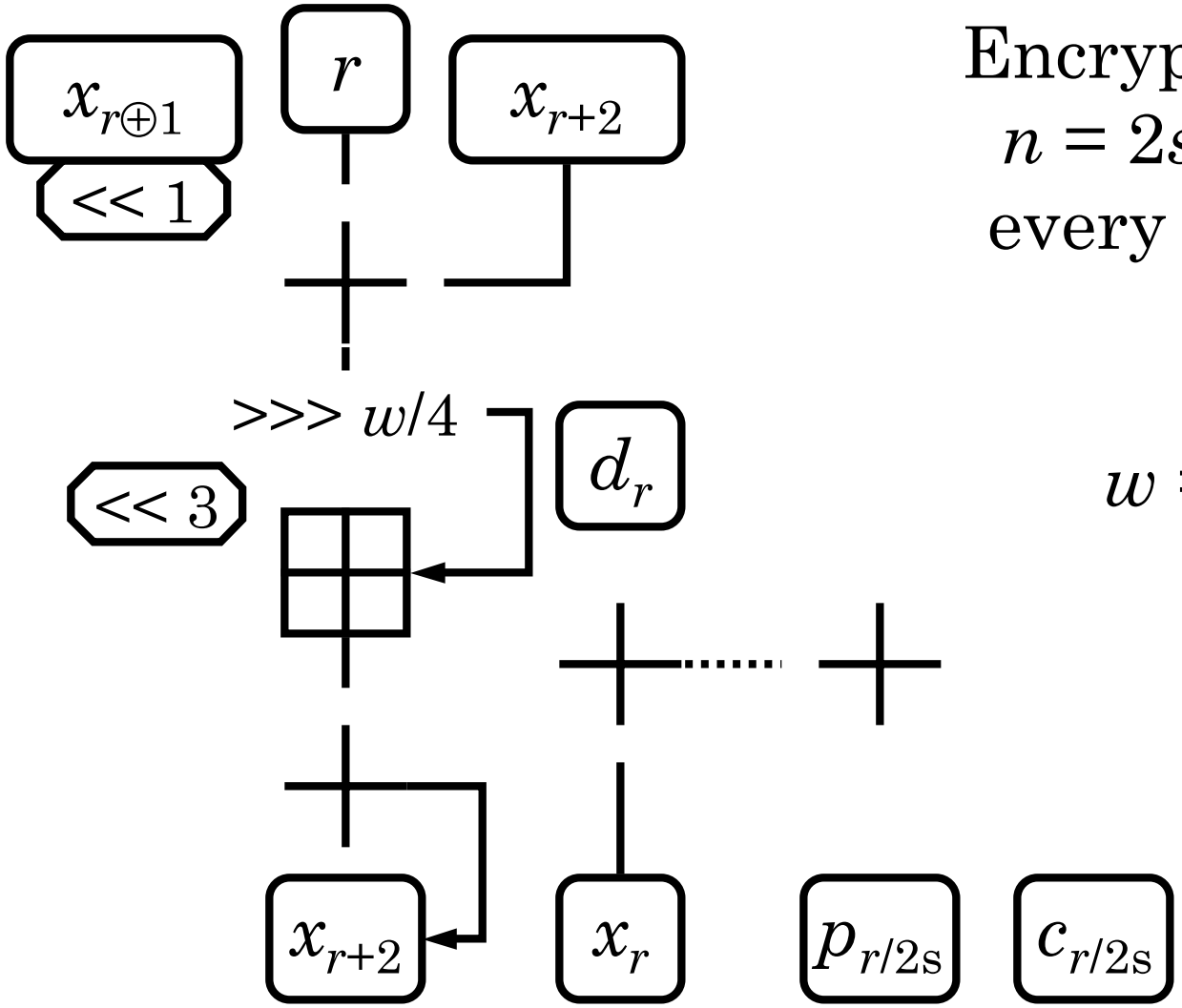
aeRUPTx2: AE stream cipher



Encryption after
$n = 2sH$ rounds
every $2s$ rounds
MAC output after
n = $2sH$ rounds
$w = 32$ or 64
$s = 8$ or H

```
x_{r+2}^=f=rotr(2*x_{r^1}^x_{r+4}^d_{r&1}^r,8)*9;   c=(d_1^=p^f^x_r);
```

RUPTx2: Stream cipher/PRF



Encryption after
$n = 2sH$ rounds
every $2s$ rounds

$w = 32$ or 64
$s = 8$ or H

```
x[r+2]^=f=rotr(2*x[r^1]^x[r+4]^d[r&1]^r,8)*9;  c=p^(d[1]^=f^x[r]);
```

ïrRUPTx2 in pseudocode

ïrRUPT*w*x2-*h*/*s*	input m bits of message p and location for h bits of hash o ; set $p_{m/w}$ = (1 << (¬ m & (w–1))) | $p_{m/w}$ & (–1 << (¬ m & (w–1))); set H = (2*h+2*w–1)/w/2*2; set $x_{0..H-1}$ = $d_{0..P-1}$ = 0; for i = 0 to (m+w–1)/w execute ïr2s(p_i), set i += 1; execute ïr2s(h); for i = 0 to H–1 execute ïr2s(0), set i += 1; for i = 0 to (h–1)/w set o_i = ïr2s(0), set i += 1; Return h bits of o as the final hash value.
ïr2s(p)	execute **(ïr1)** 2*s times; set d_1⊕=p; return d_1;
(ïr1)	set $x_{(r+2)\%H}$ ⊕= f = rotr(2*$x_{(r⊕1)\%H}$ ⊕ $x_{(r+4)\%H}$ ⊕ $d_{r\&1}$ ⊕ r, w/4)*9, set $d_{r\&1}$ ⊕= f ⊕ x_r, set r += 1;

A **complete** ïrRUPTx2 implementation.

Recent Collision Cryptanalysis

n — broken

n — could be attacked

unbroken

provably resistant to linearized collisions

min proposed *s*

s=H

Linearized Collision Attacks:

Complexity, bits	*s*=4	5	6	7	8	9	10	11	12	13	14	15	16
Indesteege+Preneel Attack:													
ïrRUPT32-128	36												
ïrRUPT32-160	38												
ïrRUPT32-192	38												
Generic Linearized Search:													
ïrRUPT32-128	65	86	107	129	150	171	192	213	234	255	277	298	319
ïrRUPT32-160	65	86	107	129	150	171	192	213	234	255	277	298	319
ïrRUPT32-192	65	86	107	129	150	171	192	213	234	255	277	298	319

Complexity, bits	*s*=4	5	6	7	8	9	10	11	12	13	14	15	16
Indesteege+Preneel Attack:													
ïrRUPT64-256	37		110										
ïrRUPT64-384	39												
ïrRUPT64-512	38												
Generic Linearized Search:													
ïrRUPT64-256	137	182	227	273	318	363	408	453	498	543	589	634	679
ïrRUPT64-384	137	182	227	273	318	363	408	453	498	543	589	634	679
ïrRUPT64-512	137	182	227	273	318	363	408	453	498	543	589	634	679

Recent Preimage Cryptanalysis

Hash	Attack Memory	Attack Time
ïrRUPT64x2-512/4	2^{480}	2^{480}

Meet-in-the-middle attacks are natural to stream hashes. Such high attack complexity using memory the size of the universe does not invalidate EnRUPT's security claims. Parallel brute-force is approximately 2^{448} times cheaper. If 2^h time * 2^h memory attack resistance is required, the H parameter should be doubled.

Currently, the fastest unbroken variant is EnRUPTx2/5. There are also no attacks against stream processing with s=2 in any of the keyed modes when $s \geq 5$ is used for the more sensitive initialisation and finalisation.

If NIST allows tuning security parameters up and not only down, we propose a choice between the more secure s=H and the faster s=8 for EnRUPT64x2 and between the more secure s=H and the faster s=H/2+1 for EnRUPT32x2. The following updated performance figures are for s=H.

Performance

Hash	ASIC Area KGE	ASIC Freq MHz	ASIC Speed Gbps	8-bit CPU CPB	32-bit SSE CPB	64-bit Intel C CPB	Memory Bytes
ïrRUPT64x2-256/8	57.8	100	6.4	200	26	10	81-88
ïrRUPT64x2-384/12	87.6	75	4.8	300	39	15	113-120
ïrRUPT64x2-512/16	117.5	50	3.2	400	52	20	145-152

Even with s=16, EnRUPT is one of the fastest SHA-3 submissions at 20 CPB.

Disadvantages

1. Not the fastest: Some of the speed was traded in favor of simplicity and flexibility (although hardware efficiency was not sacrificed and it could also turn out to be the fastest algorithm on 8-bit and 16-bit CPUs). Limited parallelisation.
2. Appears too simple to be secure: Appearances are deceiving, but the initial resistance of the professionally paranoid cryptologists to simplicity is expected and understandable.
3. Not a traditional design: Security of stream hashing is largely under-researched. Meet-in-the-middle attacks are a concern, while not being a threat to block hashes. However, MITM attacks are naturally managed by the large state required of a stream hash, which is also naturally resistant to other “odd” or even “exotic” attacks such as length extension, herding and multiple collision/preimage attacks.

Advantages

1. The simplest submission: Can be easily memorized. No constants, no s-boxes, no permutations. Lower implementation/debugging cost. Minimal structure: Less opportunities for the attacker means faster growth of trust as the algorithm remains unbroken. It is harder to expect a new attack or a new optimization.
2. 8-bit CPU, 16-bit CPU and Web friendly: Minimal RAM and code, no ROM, no rotations, no complex operations. Network router friendly: Minimal latency. Hashes 1 word at a time. Input block size is often omitted from the performance figures as it is expected to be always present. It is only one word in EnRUPT, and it does not need storage.
3. FPGA/ASIC friendly: Possibly the most area efficient submission. According to the hardware guys it is more efficient than SHA-2, MD6, Grøstl, Blake, Whirlpool, AES s-box based hashes… Faster than SHA-2. RFID friendly: Fits in under 500 gates.
4. Not a block hash: No additional block chaining mode introducing potential security flaws is required. Adds variety to the standards.

Thank you!

www.enrupt.com

Sean O'Neil

Special thanks to: Luca Henzen, Karsten Nohl,
Sebastiaan Intesteege, Bart Preneel, Dmitry Khovratovich, Ivica Nicolić