EXHIBIT I
Volume 9

	Xbox	Wii	360	PS3
On-die bootROM	✓	✓	✓	✓
On-die key storage		✓	✓	
Public-key crypto	✓	✓	✓	✓
Chain of trust	✓		✓	✓
Per-console keys		✓	✓	✓
Signed executables	✓		✓	✓
Security coprocessor		✓		✓
Full media encryption and signing		✓		
Encrypted storage		✓		✓
Self-signed storage		✓		
Memory encryption/hashing			✓	
Hypervisor			✓	✓
User/kernelmode				✓
Anti-downgrade eFUSEs			✓	

BROKEN

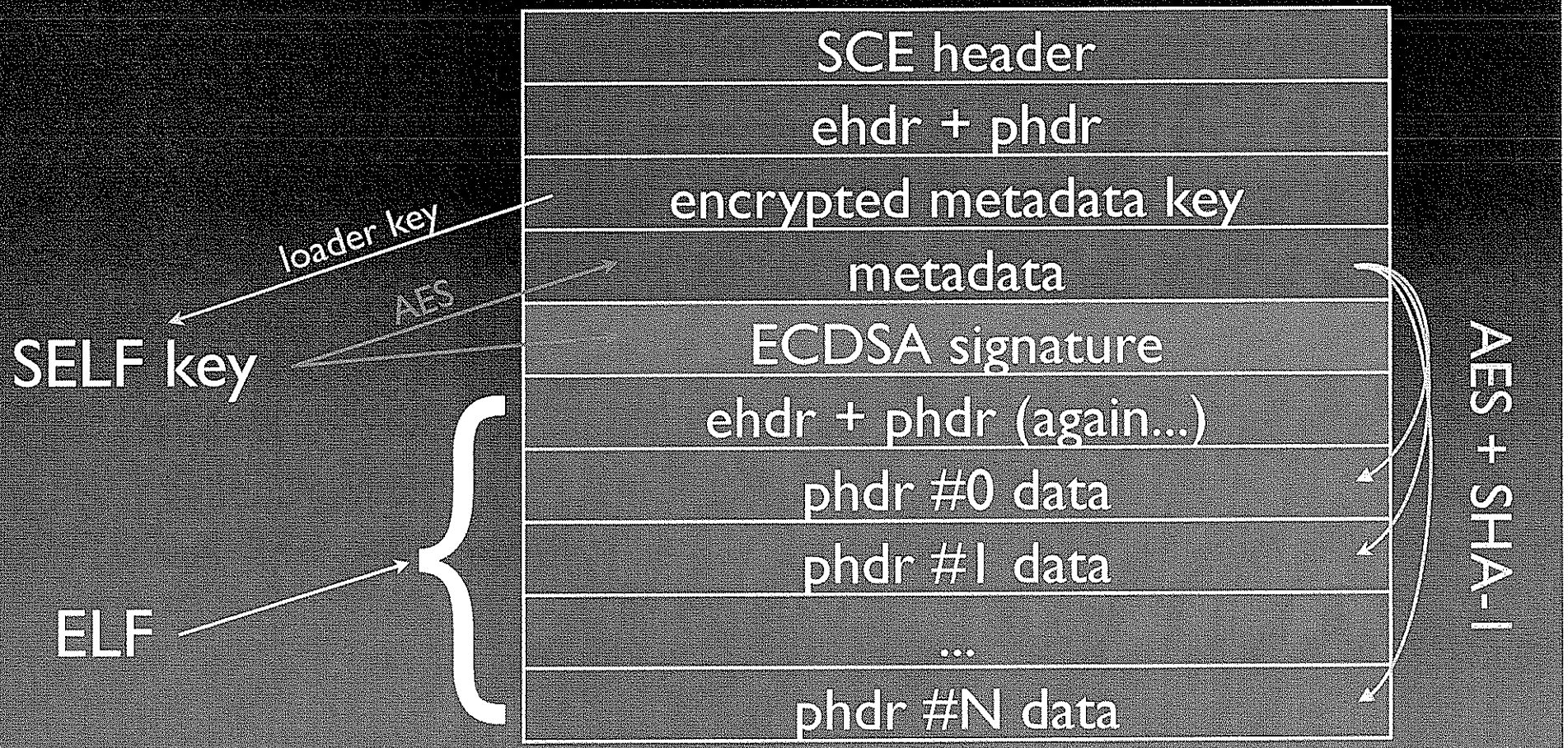
INEFFECTIVE

POINTLESS

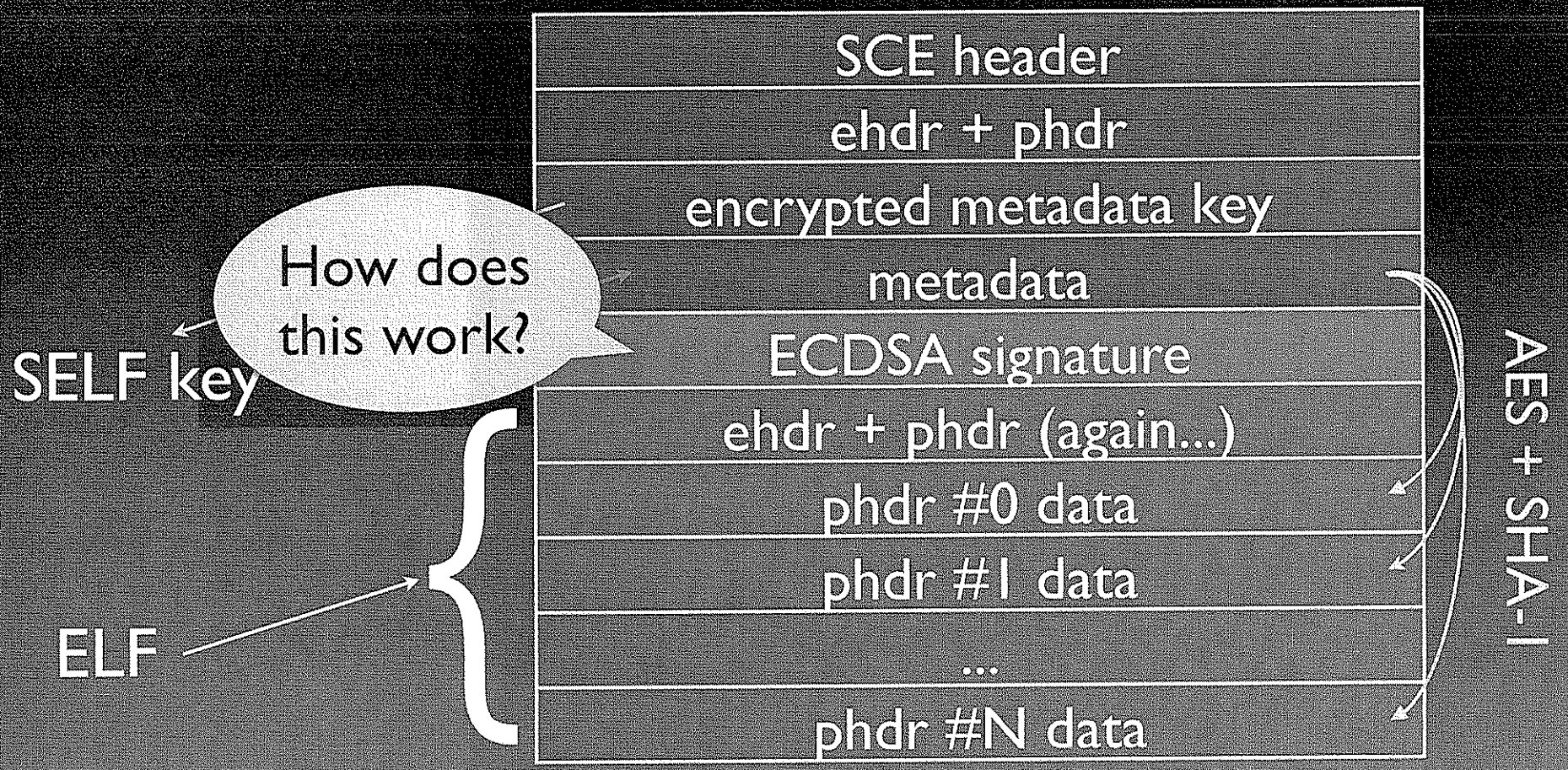
BYPASSED

USELESS

SELFs


SCE header
ehdr + phdr
encrypted metadata key
metadata
ECDSA signature
ehdr + phdr (again...)
phdr #0 data
phdr #1 data
...
phdr #N data
loader key
AES
SELF key
ELF
AES + SHA-1

SELFs


SCE header
ehdr + phdr
encrypted metadata key
metadata
ECDSA signature
ehdr + phdr (again...)
phdr #0 data
phdr #1 data
...
phdr #N data
How does this work?
SELF key
ELF
AES + SHA-1

ECDSA

These are public:

$$p, a, b, G, N \quad \text{(elliptic curve params)}$$

$$Q = \text{public key}$$

$$e = \text{hash of data}$$

$$R, S = \text{signature},$$

and these are private:

$$m = \text{random}$$

$$k = \text{private key}.$$

A signature is a pair of numbers R, S computed by the signer as

$$R = (mG)_x$$

$$S = \frac{e + kR}{m}.$$

It is imperative to have a random m for every signature: from a pair of signatures that use the same m, we can compute m and k.

$$R = (mG)_x \qquad R = (mG)_x$$

$$S_1 = \frac{e_1 + kR}{m} \qquad S_2 = \frac{e_2 + kR}{m}$$

When m is identical for two signatures, so is R, and

$$S_1 - S_2 = \frac{e_1 - e_2}{m}$$

$$m = \frac{e_1 - e_2}{S_1 - S_2}$$

$$k = \frac{mS_i - e_i}{R} \quad \left[= \frac{e_1 S_2 - e_2 S_1}{R(S_1 - S_2)} \right].$$

Our ECDSA code

Used for HBC's network update function

```
def generate_ecdsa(k, sha):
    k = bytes_to_long(k)
    e = bytes_to_long(sha)

    m = open("/dev/random","rb").read(30)

    if len(m) != 30:
        raise Exception("Failed to get m")
    m = bytes_to_long(m) % ec_N

    r = (m * ec_G).x.tobignum() % ec_N
    kk = ((r * k) + e) % ec_N
    s = (bn_inv(m, ec_N) * kk) % ec_N
    r = long_to_bytes(r, 30)
    s = long_to_bytes(s, 30)
    return r,s
```

Our ECDSA code

Used for HBC's network update function

```
def generate_ecdsa(k, sha):
    k = bytes_to_long(k)
    e = bytes_to_long(sha)

    m = open("/dev/random","rb").read(30)

    if len(m) != 30:
        raise Exception("Failed to get m")
    m = bytes_to_long(m) % ec_N

    r = (m * ec_G).x.tobignum() % ec_N
    kk = (e + r * k) * m % ec_N
    s = (pow_inv(m, ec_N) * (kk)) % ec_N
    r = long_to_bytes(r, 30)
    s = long_to_bytes(s, 30)
```

Sony's ECDSA code

```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```

With private keys you can SIGN THINGS

You have earned a trophy.
Public Private Keys

With private keys you can SIGN THINGS

	Xbox	Wii	360	PS3
On-die bootROM	✓	✓	✓	✓
On-die key storage		✓	✓	
Public-key crypto	✓	✓	✓	✓
Chain of trust	✓		✓	✓
Per-console keys		✓	✓	✓
Signed executables	✓		✓	✓
Security coprocessor		✓		✓
Full media encryption and signing		✓		
Encrypted storage		✓		
Self-signed storage		✓		
Memory encryption/hashing			✓	
Hypervisor			✓	✓
User/kernelmode				✓
Anti-downgrade eFUSEs			✓	

BROKEN

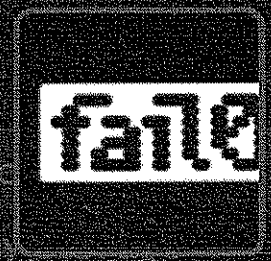
INEFFECTIVE

POINTLESS

BYPASSED

USELESS

	Xbox	Wii	360	PS3
On-die bootROM	✓	✓	✓	✓
On-die key storage		✓	✓	
Public-key crypto	✓	✓	✓ EPICFAIL	✓
Chain of trust	✓		✓ BROKEN	
Per-console keys		✓	✓	✓
Signed executables	✓		✓ INEFFECTIVE	✓
Security coprocessor		✓	POINTLESS	✓
Full media encryption and signing		✓		
Encrypted storage		✓	BYPASSED	✓
Self-signed storage		✓		
Memory encryption/hashing			✓	
Hypervisor			✓ USELESS	✓
User/kernelmode				✓
Anti-downgrade eFUSEs			✓	



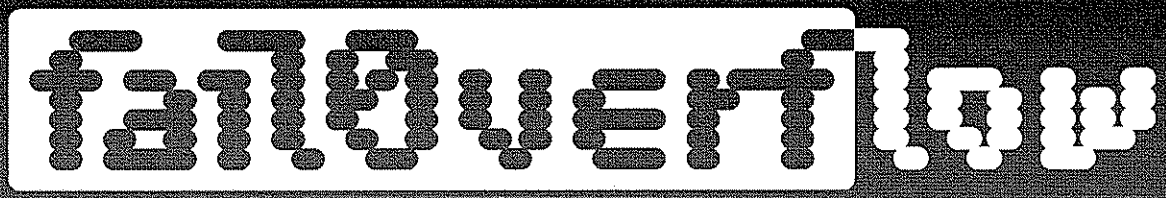
You have earned a trophy.

Fail0verflow

On-die				
On-die k				
Public-key crypto	✓	✓	✓	✓
Chain of trust	✓		✓	✓
Per-console keys		✓	✓	✓
Signed executables	✓		✓	✓
Security coprocessor		✓		✓
Full media encryption and signing		✓		
Encrypted storage		✓		✓
Self-signed storage		✓		
Memory encryption/hashing			✓	
Hypervisor			✓	✓
User/kernelmode				✓
Anti-downgrade eFUSEs			✓	

EPIC FAIL
BROKEN
INEFFECTIVE
POINTLESS
BYPASSED
USELESS

Thanks, Sony!



http://fail0verflow.com