NIST
National Institute of
Standards and Technology
U.S. Department of Commerce

Cryptography Classes in Bugs Framework (BF):

Encryption Bugs (ENC), Verification Bugs (VRF), and Key Management Bugs (KMN)

Irena Bojanova[1], Paul E. Black[1], Yaacov Yesha[1,2], Farhan Nadeem[1] ([1]NIST, [2]UMBC)

UMBC
AN HONORS UNIVERSITY IN MARYLAND

Advances in scientific foundations of cybersecurity rely on availability of precise definitions of software bugs and clear descriptions of software vulnerabilities and attacks. The Bugs Framework (BF) builds rigorous definitions and taxonomy for expressing software vulnerabilities through bugs attributes, causes, and consequences. The ability to understand, avoid, and correct software bugs would promote development of reliable systems and societal improvements.

Cryptography

Cryptography is a broad, complex, and subtle area.
It incorporates clearly separate processes, such as:

- encryption/decryption
- verification of data or source
- key management.

There are bugs if the software does not properly:

- transform data into unintelligible form
- verify authenticity or correctness
- manage keys, or perform other related operations.

Some transformations require keys, for example encryption and decryption, while others do not, for example secret sharing.

Authenticity covers data integrity, data source identity, origin non-repudiation, and secret sharing content. Correctness is verified for uses such as zero-knowledge proofs.

Cryptographic processes use particular algorithms to achieve particular security services.

Model of Cryptographic Store or Transfer

To illustrate our ENC, VRF, and KMN classes, we use cryptographic store or transfer:

Cryptographic Store/Transfer Bugs: The software does not properly encrypt/ decrypt, verify, or manage keys for data to be securely stored or transferred [1].

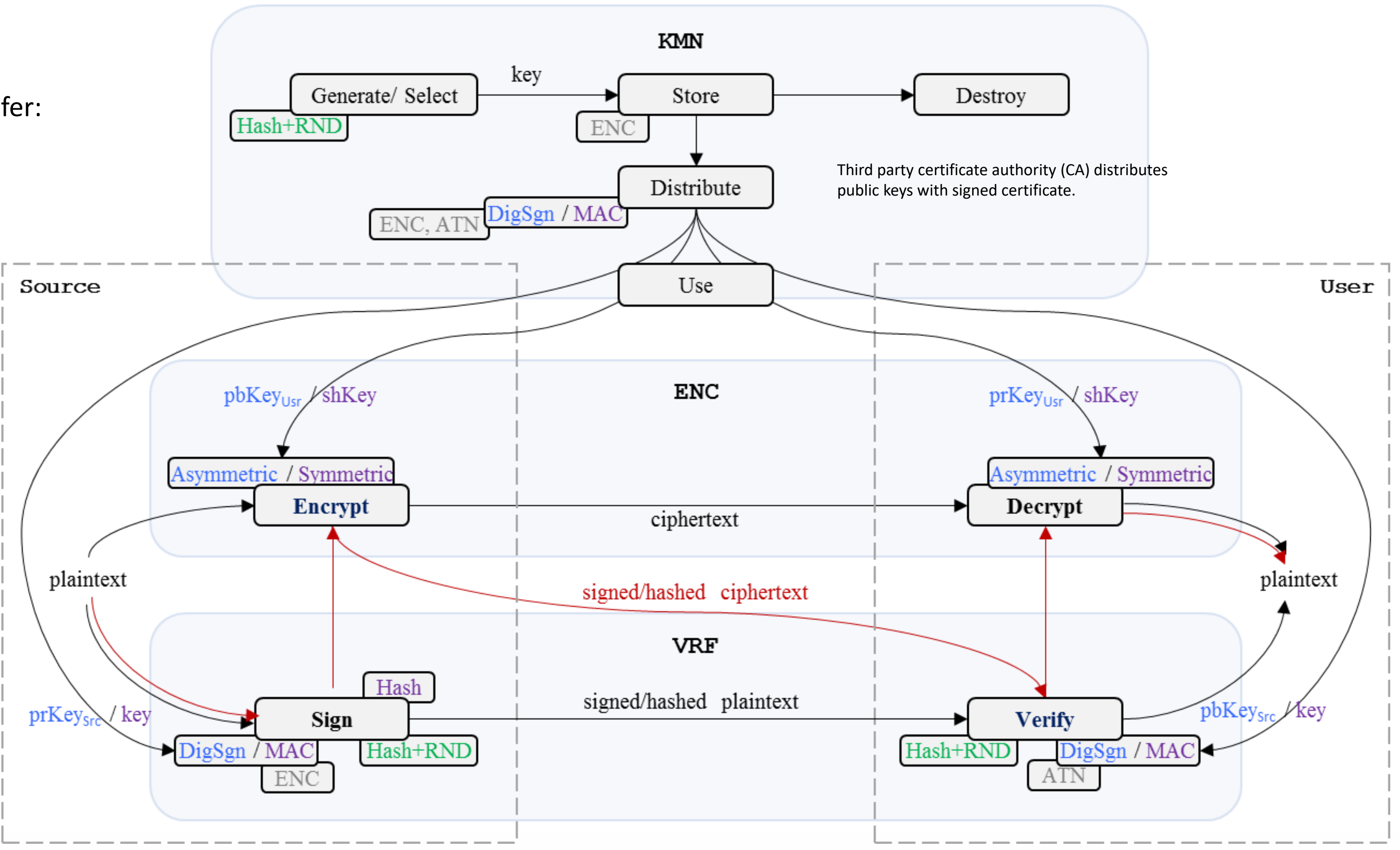
✓ Encryption may occur in tandem with Verification or it may precede Verification serially, if the cipertext is signed or hashed.
✓ Encryption uses Key Management, and Key Management likely uses Encryption and Verification to handle keys.
✓ Key management could be by third party, source, or user – thus KMN area intersects Source and User areas.

Asymmetric—two related keys (public, private)
Source ($pbKey_{Srs}$, $prKey_{Srs}$), User ($pbKey_{Usr}$, $prKey_{Usr}$)

- Source encrypts with $pbKey_{Usr}$
- User decrypts with $prKey_{Usr}$
- Source signs with $prKey_{Srs}$
- User verifies with $pbKey_{Srs}$

Symmetric—one secretly shared key shKey

- Source encrypts with shKey
- User decrypts with shKey

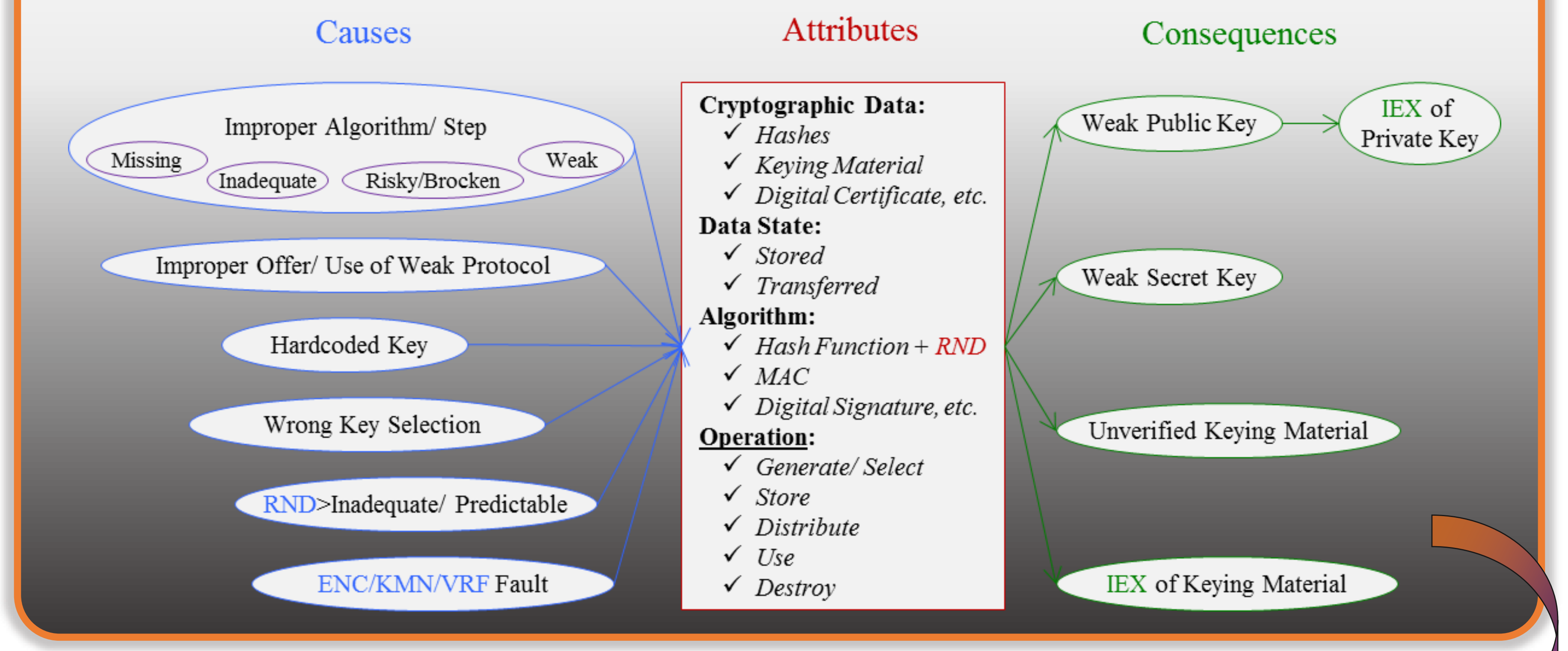


Third party certificate authority (CA) distributes public keys with signed certificate.

[1] Bojanova, I., Black, P. E., Yesha, Y., Cryptography Classes in Bugs Framework (BF): Encryption Bugs (ENC), Verification Bugs (VRF), and Key Management Bugs (KMN). IEEE Software Technology Conference (STC 2017), NIST, Gaithersburg, USA. September 25-28.

Key Management Bugs (KMN)

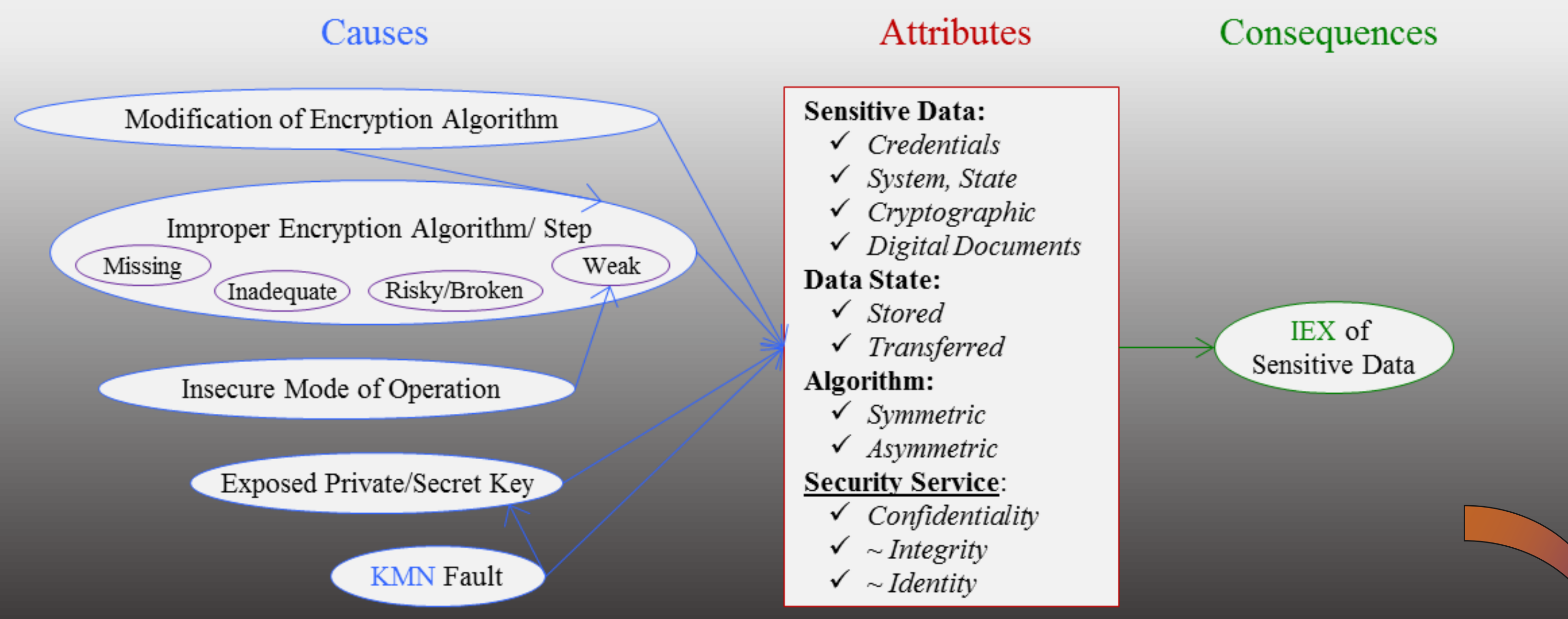
The software does not properly generate, store, distribute, use, or destroy cryptographic keys and other keying material.



Cryptographic Data:
✓ *Hashes*
✓ *Keying Material*
✓ *Digital Certificate, etc.*
Data State:
✓ *Stored*
✓ *Transferred*
Algorithm:
✓ *Hash Function + RND*
✓ *MAC*
✓ *Digital Signature, etc.*
Operation:
✓ *Generate/ Select*
✓ *Store*
✓ *Distribute*
✓ *Use*
✓ *Destroy*

Encryption Bugs (ENC)

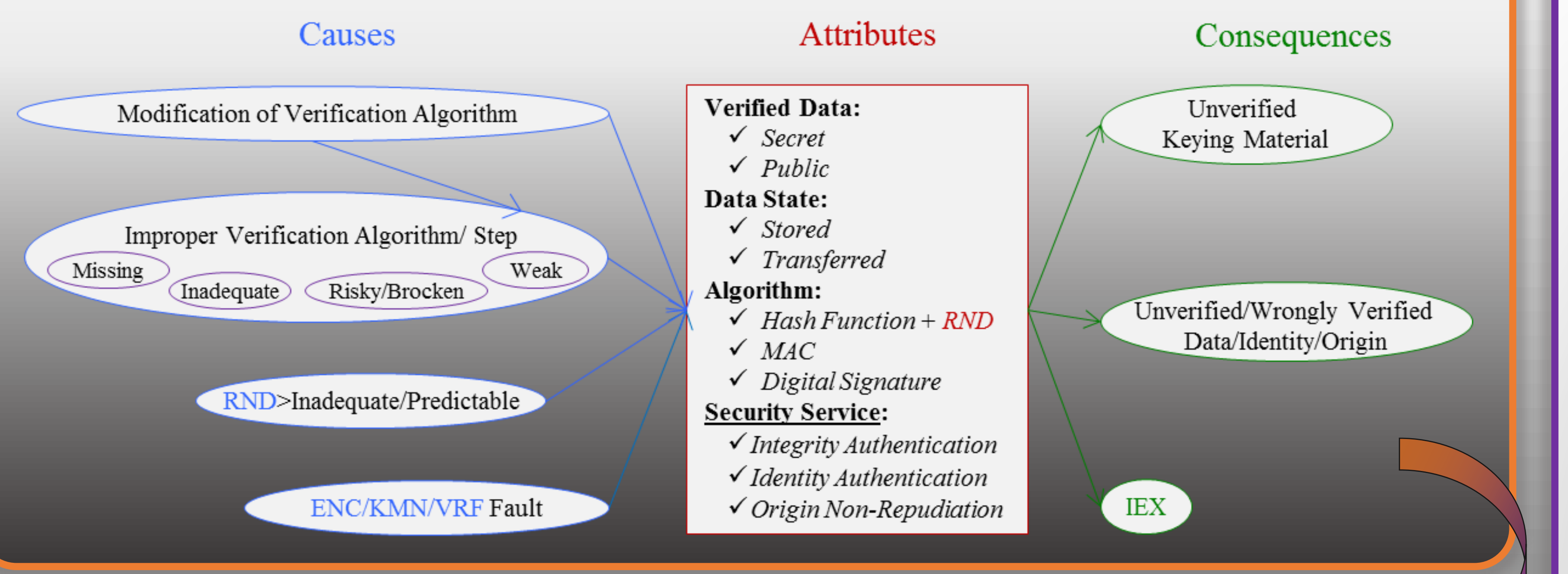
The software does not properly transform sensitive data (plaintext) into unintelligible form (ciphertext) using cryptographic algorithm and key(s).


Causes
Attributes
Consequences
Modification of Encryption Algorithm
Improper Encryption Algorithm/ Step
Missing
Inadequate
Risky/Broken
Weak
Insecure Mode of Operation
Exposed Private/Secret Key
KMN Fault
IEX of Sensitive Data

Sensitive Data:
✓ *Credentials*
✓ *System, State*
✓ *Cryptographic*
✓ *Digital Documents*
Data State:
✓ *Stored*
✓ *Transferred*
Algorithm:
✓ *Symmetric*
✓ *Asymmetric*
Security Service:
✓ *Confidentiality*
✓ *~ Integrity*
✓ *~ Identity*

Verification Bugs (VRF)

The software does not properly sign data, check and prove source, or assure data is not altered.


Causes
Attributes
Consequences
Modification of Verification Algorithm
Improper Verification Algorithm/ Step
Missing
Inadequate
Risky/Brocken
Weak
RND>Inadequate/Predictable
ENC/KMN/VRF Fault
Unverified Keying Material
Unverified/Wrongly Verified Data/Identity/Origin
IEX

Verified Data:
✓ *Secret*
✓ *Public*
Data State:
✓ *Stored*
✓ *Transferred*
Algorithm:
✓ *Hash Function + RND*
✓ *MAC*
✓ *Digital Signature*
Security Service:
✓ *Integrity Authentication*
✓ *Identity Authentication*
✓ *Origin Non-Repudiation*

ENC Examples

CVE-2007-5460 → ENC

Cause: Weak Encryption Algorithm (XOR cipher with fixed key)
Attributes:
Sensitive Data: Credentials (PINs/passwords)
Data State: Transferred (over network)
Algorithm: Symmetric (that allows obtaining shared key, by sniffing or spoofing the docking process, and decryption)
Security Service: Confidentiality
Consequence: IEX of Sensitive Data (credentials)

Use of weak encryption algorithm (XOR cipher with fixed key) allows obtaining the shared symmetric key (by sniffing or spoofing the docking process) and decryption of transferred (over network) credentials (PINs/passwords), which is confidentiality failure and IEX of sensitive data (PINs/passwords).

CVE-2002-1697 → ENC

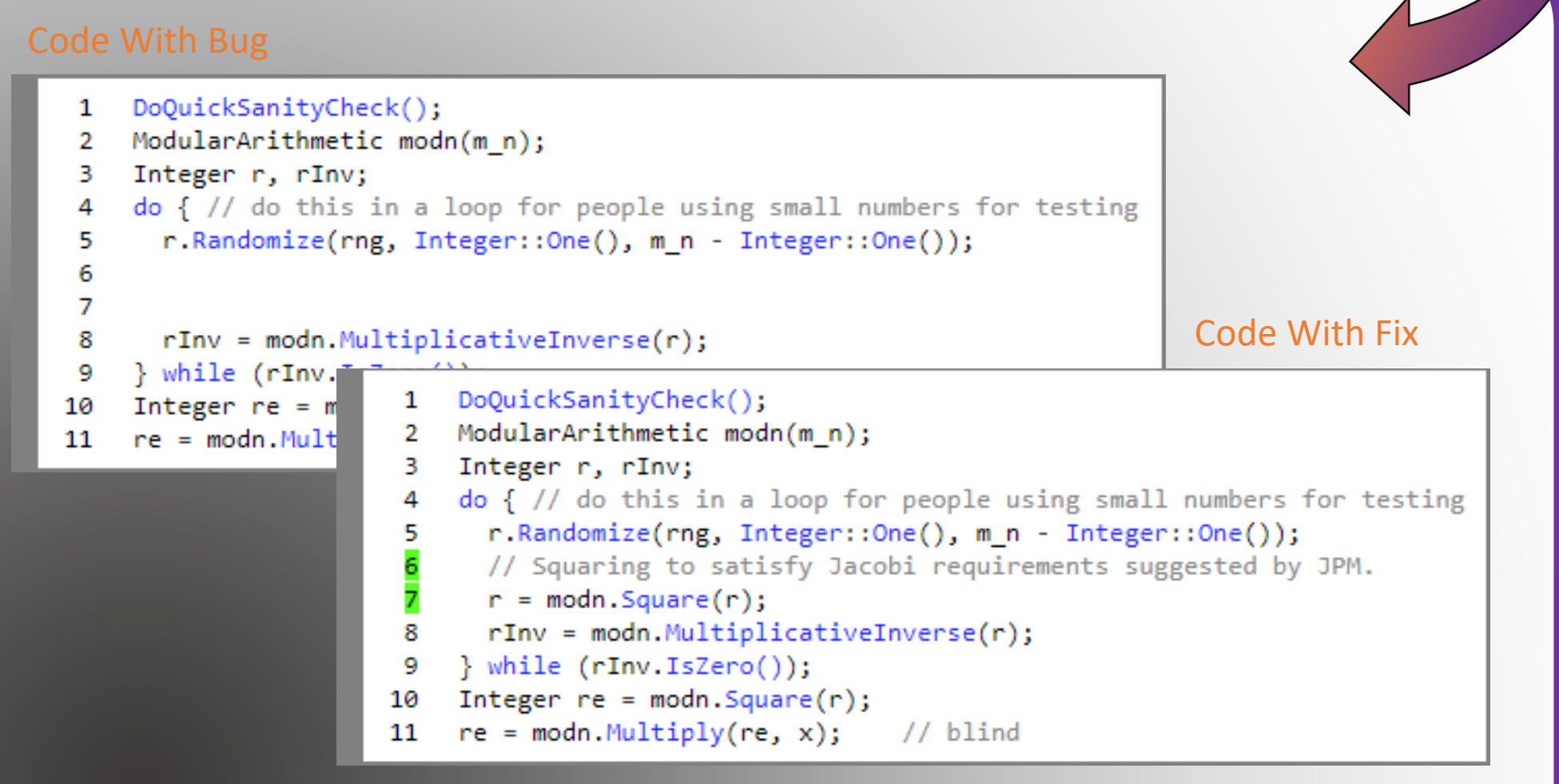
Causes: Insecure Mode of Operation (ECB) leads to Weak Encryption Algorithm (for same shared key produces same ciphertext from same plaintext)
Attributes:
Sensitive Data: Any (Credentials, Cryptographic, …)
Data State: Transferred (over network)
Algorithm: Symmetric (that allows identifying patterns and data recovery)
Security Service: Confidentiality
Consequence: IEX of Sensitive Data

Use of insecure mode of operation (ECB) leads to weak symmetric encryption algorithm (for same shared key produces same ciphertext from same plaintext) that allows identifying patterns and recovery of transferred (over network) sensitive data, which is confidentiality failure and IEX of that sensitive data.

VRF Examples

CVE 2015-2141 → VRF

Cause: Modification of Verification Algorithm by adding a step (blinding)
Attributes:
Verified Data: Any (Secret/ Public)
Data State: Transferred (over network)
Algorithm: Digital Signature (Rabin-Williams) (that allows obtaining the private key in cases of incorrect unblinding)
Security Service: Identity Authentication
Consequence: IEX


Code With Bug

```
DoQuickSanityCheck();
ModularArithmetic modn(m_n);
Integer r, rInv;
do { // do this in a loop for people using small numbers for testing
  r.Randomize(rng, Integer::One(), m_n - Integer::One());


  rInv = modn.MultiplicativeInverse(r);
} while (rInv.
Integer re = m
re = modn.Mult
```

Code With Fix

```
DoQuickSanityCheck();
ModularArithmetic modn(m_n);
Integer r, rInv;
do { // do this in a loop for people using small numbers for testing
  r.Randomize(rng, Integer::One(), m_n - Integer::One());
  // Squaring to satisfy Jacobi requirements suggested by JPM.
  r = modn.Square(r);
  rInv = modn.MultiplicativeInverse(r);
} while (rInv.IsZero());
Integer re = modn.Square(r);
re = modn.Multiply(re, x);    // blind
```

Modification of verification algorithm (digital signature, Rabin-Williams) by adding a step (blinding before signing) allows obtaining the private key in cases of incorrect unblinding, which leads to identity authentication failure and may be exploited for IEX.

CVE 2001-1585 → VRF

Cause Missing Verification Step (challenge-response) in public key authentication
Attributes:
Verified Data: Any (Secret/ Public)
Data State: Transferred (over network)
Algorithm: Digital Signature (not using such allows private key not to be verified by public key)
Security Service: Identity Authentication
Consequence: IEX

Missing verification step (challenge-response) in public key authentication allows private key for digital signature not to be verified by public key, which leads to identity authentication failure and may be exploited for IEX.

KMN Example

CVE-2015-0204, 1637, 1067 (FREAK) → KMN & ENC

An inner KMN leads to an inner ENC, which leads to an outer ENC.
Inner KMN:
Cause: Improper Offer of Weak Protocol (Export RSA – offered from MITM-tricked server and accepted by client)
Attributes:
Cryptographic Data: Keying Material (pair of private and public keys)
Data State: Transferred (over network)
Algorithm: Export RSA (512-bits key generation based on prime numbers, such that private key can be obtained from public key through factorization)
Operation: Generate
Consequence: IEX Keying Material (private key)

Inner ENC:
Causes: KMN Fault leads to Exposed Private Key
Attributes:
Sensitive Data: Cryptographic (Pre-Master Secret)
Data State: Transferred (over network)
Algorithm: Asymmetric (RSA) (that allows decryption of Pre-Master Secret using exposed private key and computation of Master Secret)
Security Service: Confidentiality
Consequence: IEX of Sensitive Data (Master Secret)

Outer ENC:
Causes: KMN Fault leads to Exposed Secret Key (Master Secret)
Attributes:
Sensitive Data: Credentials (passwords, credit cards)
Data State: Transferred (over network)
Algorithm: Symmetric (key is known)
Security Service: Confidentiality
Consequence: IEX of Sensitive Data (credentials)

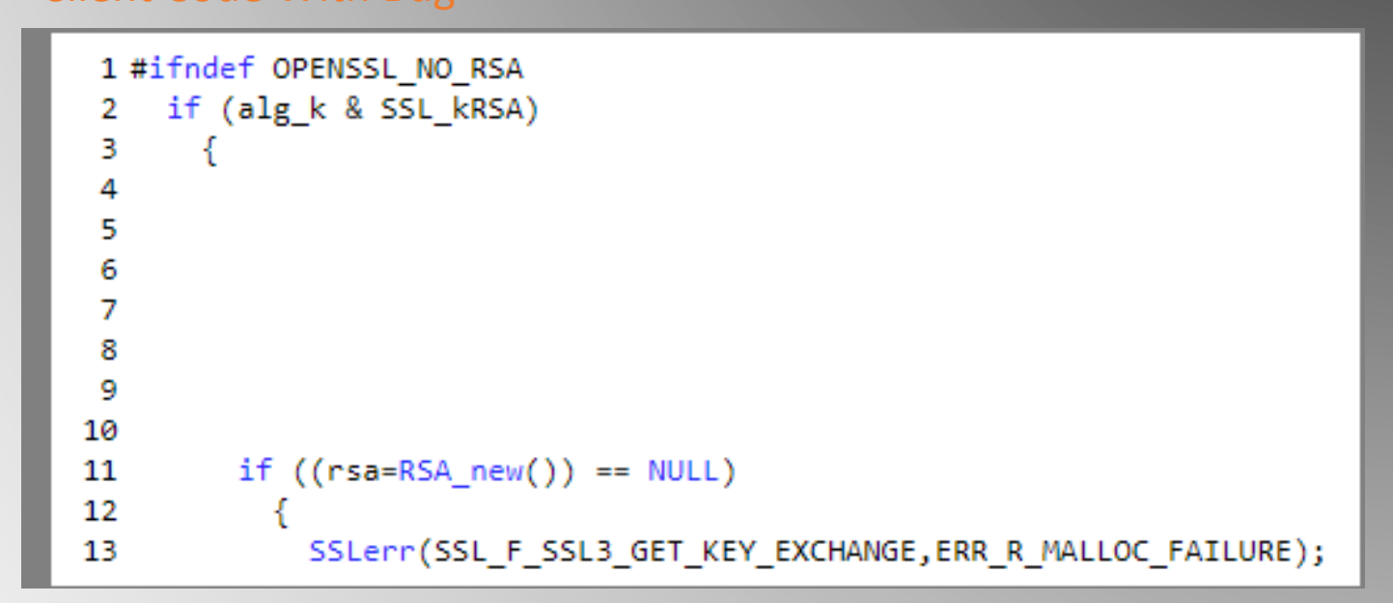
An inner KMN leads to an inner ENC, which leads to an outer ENC.

Inner KMN: Improper offer of weak protocol (Export RSA from MITM-tricked server and accepted by client) allows use of 512-bits key generation (based on prime numbers) such that the private key can be obtained from the public key through factorization, which may be exploited for IEX of keying material (private key).

Inner ENC: KMN fault leads to exposed private key for asymmetric encryption (RSA) that allows decryption of transferred (over network) cryptographic data (Pre-Master Secret) and computation of other cryptographic data (Master Secret), which is confidentiality failure and IEX of sensitive data (Master Secret).

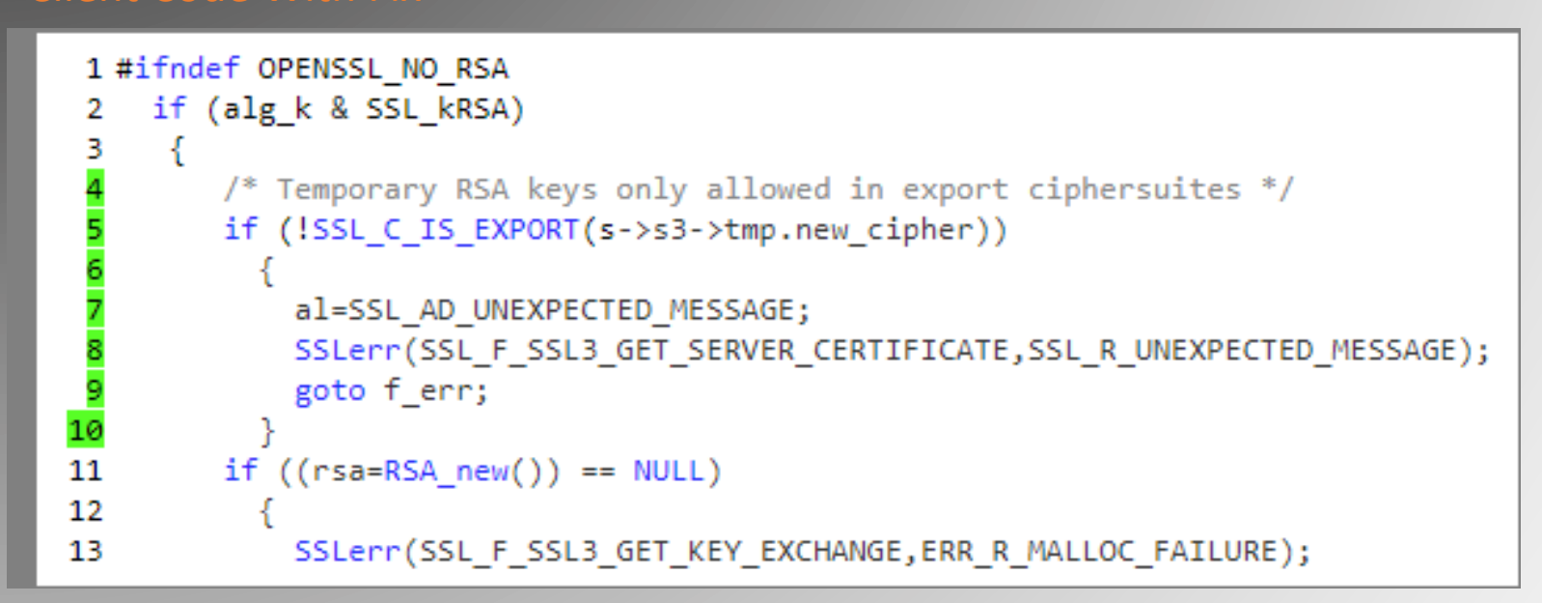
Outer ENC: KMN fault leads to exposed secret key (Master Secret) for symmetric encryption allows decryption of credentials (passwords, credit cards, etc.) transferred (over network), which is confidentiality failure and IEX of sensitive data (passwords, credit cards, etc.).

Client Code With Bug



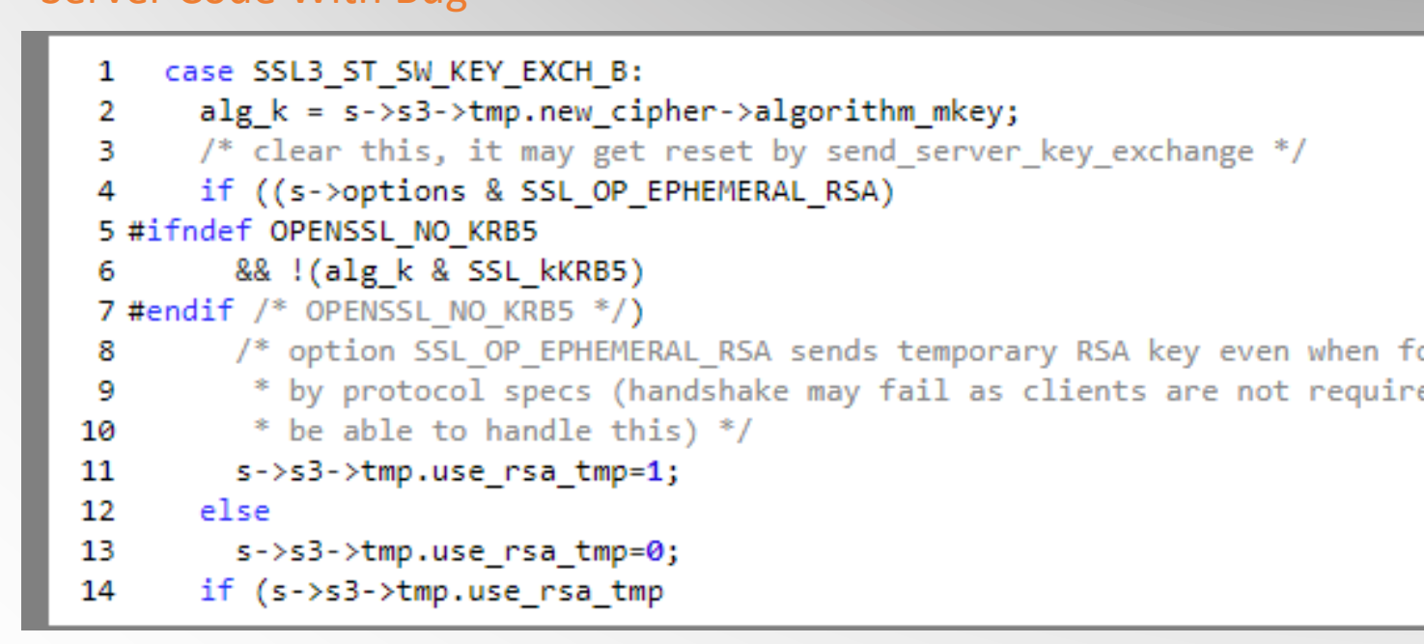
```
#ifndef OPENSSL_NO_RSA
  if (alg_k & SSL_kRSA)
    {







      if ((rsa=RSA_new()) == NULL)
        {
          SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,ERR_R_MALLOC_FAILURE);
```

Client Code With Fix



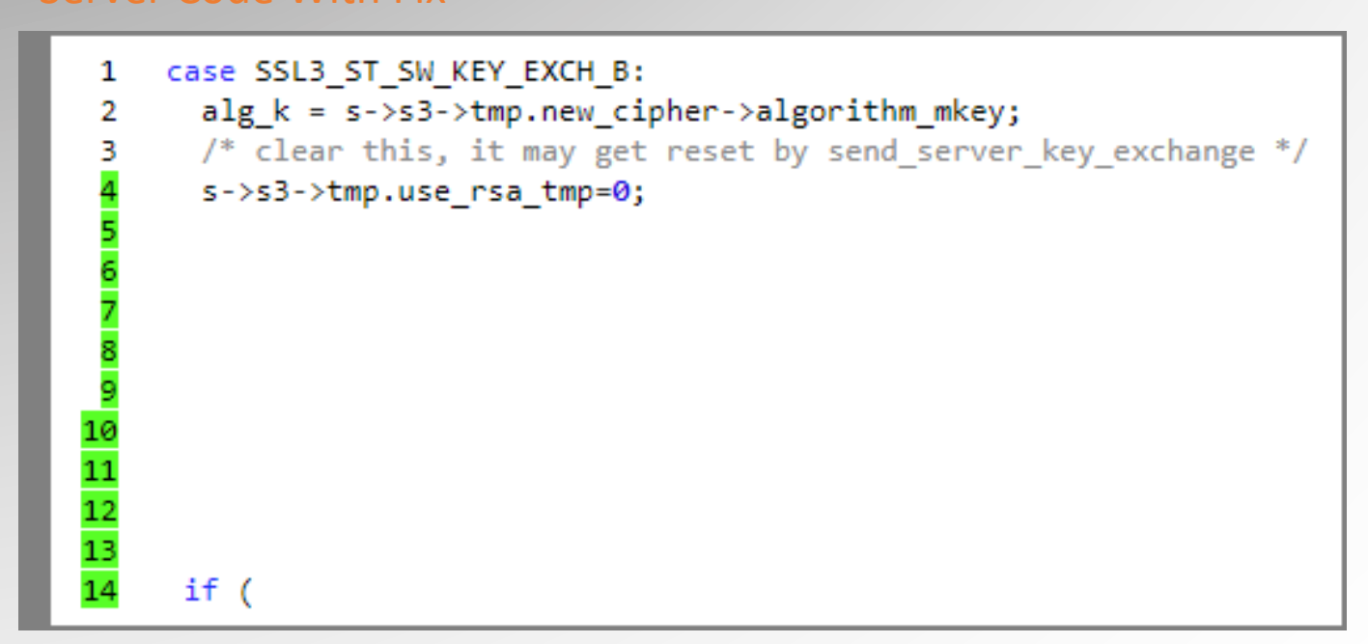
```
#ifndef OPENSSL_NO_RSA
  if (alg_k & SSL_kRSA)
   {
      /* Temporary RSA keys only allowed in export ciphersuites */
      if (!SSL_C_IS_EXPORT(s->s3->tmp.new_cipher))
        {
          al=SSL_AD_UNEXPECTED_MESSAGE;
          SSLerr(SSL_F_SSL3_GET_SERVER_CERTIFICATE,SSL_R_UNEXPECTED_MESSAGE);
          goto f_err;
        }
      if ((rsa=RSA_new()) == NULL)
        {
          SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,ERR_R_MALLOC_FAILURE);
```

Server Code With Bug



```
  case SSL3_ST_SW_KEY_EXCH_B:
    alg_k = s->s3->tmp.new_cipher->algorithm_mkey;
    /* clear this, it may get reset by send_server_key_exchange */
    if ((s->options & SSL_OP_EPHEMERAL_RSA)
#ifndef OPENSSL_NO_KRB5
        && !(alg_k & SSL_kKRB5)
#endif /* OPENSSL_NO_KRB5 */)
        /* option SSL_OP_EPHEMERAL_RSA sends temporary RSA key even when forbidden
         * by protocol specs (handshake may fail as clients are not required to
         * be able to handle this) */
        s->s3->tmp.use_rsa_tmp=1;
    else
        s->s3->tmp.use_rsa_tmp=0;
    if (s->s3->tmp.use_rsa_tmp
```

Server Code With Fix



```
  case SSL3_ST_SW_KEY_EXCH_B:
    alg_k = s->s3->tmp.new_cipher->algorithm_mkey;
    /* clear this, it may get reset by send_server_key_exchange */
    s->s3->tmp.use_rsa_tmp=0;










   if (
```