Using Formal Methods to Analyze Encryption Protocols

A secure network with encryption uses

- Encryption algorithms
- Encryption protocols

Encryption algorithm
converts clear text into cipher text or cipher text into clear text

Encryption protocol
A set of rules or procedures for using an encryption algorithm to send and receive messages in a secure manner over a network

Approach

Formally specify
- components of the facility
- cryptographic operations

Express desired properties of the protocol as state invariants

Verification system automatically generates the necessary proof obligations to guarantee that the desired properties are preserved

Nothing is proved about the encryption algorithms being used

Formal Specification

Used Ina Jo

System modeled as a state machine
- components are state variables and constants
- operations are state transitions (transforms in Ina Jo)
- Desirable properties are state invariants (criterion in Ina Jo)

Example System

Based on IBM's System Network Architecture (SNA), which supports session-level cryptography

Single-domain communication system using dynamically generated primary keys and two secret master keys

Example System

Session Keys
- One per terminal
- Dynamically generated by the host for each session
- Primary communication key


```
Variable
    session_key(terminal):key
```


Example System

Session Keys
- One per terminal
- Dynamically generated by the host for each session
- Primary communication key

Terminal Keys
- One per terminal
- Permanent
- Used to distribute new sessions keys


```
Variable
    session_key(terminal):key

Constant
    terminal_key(terminal):key
```


Example System

Session Keys
- One per terminal
- Dynamically generated by the host for each session
- Primary communication key

Terminal Keys
- One per terminal
- Permanent
- Used to distribute new sessions keys

Two Master Keys

KMH0, KMH1

```
Variable
    session_key(terminal):key

Constant
    terminal_key(terminal):key,
    KMH0, KMH1:key
```


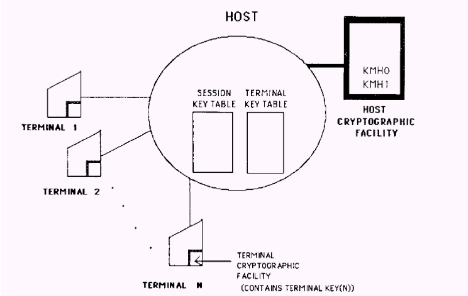

Fig. 1. System architecture.

- Terminal keys are stored in the terminal's cryptographic facility
- Host keys are stored in the host's cryptographic facility
- Terminal keys and session keys are stored in the host in encrypted form

- Terminal keys are stored in the terminal's cryptographic facility
- Host keys are stored in the host's cryptographic facility
- Terminal keys and session keys are stored in the host in encrypted form

- Session Key Table - current session keys encrypted using KMH0
- Terminal Key Table - terminal keys encrypted using KMH1

- Session Key Table - current session keys encrypted using KMH0
  - E_{KMH0}(session_key(i))

- Terminal Key Table - terminal keys encrypted using KMH1
  - E_{KMH1}(terminal_key(i))

Operations Provided

- Encipher Data (ECPH)
- Decipher Data (DCPH)
- Reencipher From Master Key (RFMK)

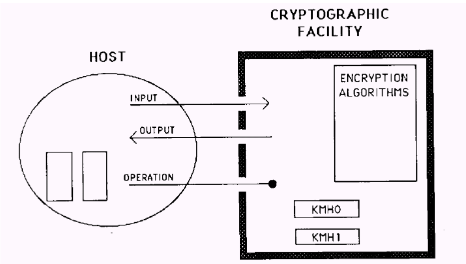

Fig. 2. Host and cryptographic facility.

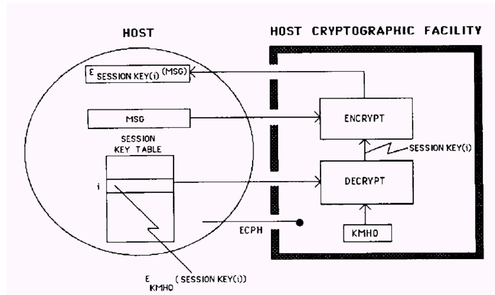

Fig. 3. The encipher operation (ECPH).

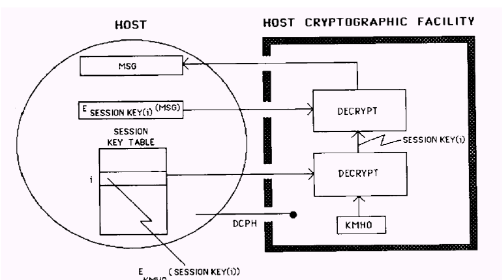

Fig. 4. The decipher operation (DCPH).

Generate Session Key Operation

Not part of the cryptographic facility

Can use pseudo-random number generator to generate a value interpreted as encrypted version of new session key E_{KMH0}(session_key(j))

Meyer-Matayas 1980

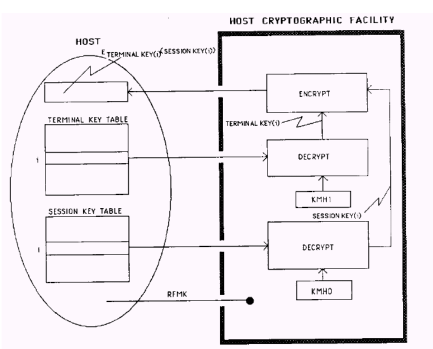

Fig. 5. Reencipher from master key (RFMK).


```
Variable
    session_key(terminal):key,
    intruder_info:information,
    keys_used:information
```

```
Variable
    session_key(terminal):key,
    intruder_info:information,
    keys_used:information

Transform ECPH(K:key,T:text)
  Effect
   N"intruder_info =
    intruder_info U
     {Encrypt(Decrypt(KMH0,K),T))}
```


Properties to be assumed about the encryption algorithms are represented as axioms in Ina Jo

Properties to be assumed about the encryption algorithms are represented as axioms in Ina Jo

For example the commutativity of the encryption and decryption functions would be represented as:

Axiom ∀ T:Text, K1,K2:key
(Encrypt(K1,Decrypt(K2,T))
= Decrypt(K2,Encrypt(K1,T)))

Analysis Technique

Write the formal specification

Generate the necessary proof obligations to guarantee preservation of the desirable properties

If the proof obligations can be proved and the encryption algorithms satisfy the specified axioms, then the system will satisfy its critical requirements

Analysis Technique

Write the formal specification

Generate the necessary proof obligations to guarantee preservation of the desirable properties

If the proof obligations can be proved and the encryption algorithms satisfy the specified axioms, then the system will satisfy its critical requirements

If the proofs fail, then the unproved parts of the proof obligations often indicate weaknesses in the protocol or incompleteness in the specification

Cryptoanalytic Assumptions

- What does the intruder know about the system
- What does the intruder observe
- What can the intruder do

Assume the intruder can

- Obtain any information communicated between the host and the terminals
- Masquerade as an authorized user
- Invoke operations of the host's cryptographic facilities

Also need a definition of security

An obvious desirable property is

Intruder never gets a key in the clear


```
Variable
    session_key(terminal):key,
    intruder_info:information,
    keys_used:information
Transform ECPH(K:key,T:text)
  Effect
    N"intruder_info =
      intruder_info U
       {Encrypt(Decrypt(KMH0,K),T))
Criterion
  ∀K:KEY(K ∈ intruder_info
         → ~ (K ∈ keys_used) )
```


Example of incompleteness in the specification

The original specification did not consider the possibility of an encrypted key being coincidentally equal to one of the keys used

E_{KMH0}(session_key(j)) ∈ keys_used

Add to initial condition

```
∀K1,K2:key (
 K1∈ intruder_info &
 K2 ∈ keys_used
    →K1 ≠ K2 )
```

(i.e., the encrypted version of a key cannot coincidentally be equal to some other key that was used)

In Generate_Session_Key transform one also needs to add

```
  ~(Encrypt(KMH0,K)∈ keys_used)
& ~(Encrypt(terminal_key(ter),K)
       ∈ keys_used)
```

Since these encrypted versions will become available to the intruder

Example of a Weakness in the Protocol

Weakness: two master keys are equal

Scenario:

- Intruder gets current session key for terminal T encrypted under T's terminal key
- Intruder gets terminal T's terminal key encrypted under KMH1
- Intruder invokes decipher operation using encrypted terminal key instead of the session key and encrypted session key in place of an encrypted message

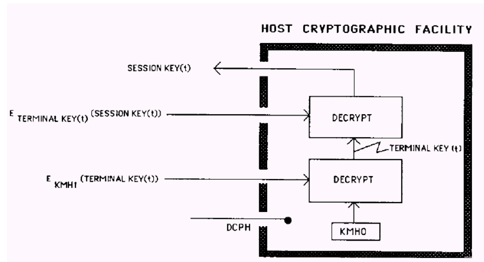

Fig. 6. Protocol flaw using DCPH.


```
Variable
    session_key(terminal):key

Constant
    terminal_key(terminal):key,
    KMH0, KMH1:key

Axiom
    KMH0 ≠ KMH1
```


Also looked at DES semi-weak and weak key pairs

Semi-weak key pairs k1, k2 have the property that for any clear text T, encryption with k1 followed by encryption with k2 results in the original clear text T

∀ T:text (encrypt(k2,encrypt(k1,T)) = T)

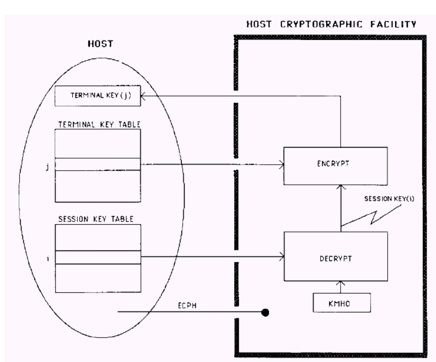

Fig. 7. Semiweak key DES key flaw using ECPH.

Summary

Discovered weakness in protocol and demonstrated the flaw using a specification testing tool

This was a previously known weakness

True value will be demonstrated when a flaw is discovered in a protocol previously assumed to be secure

Advantages of Approach

Only need to replace the axioms to analyze different algorithms using the same protocol

Properties of the cryptographic protocols and facilities can be tested before the system is built

Disadvantages of the Approach

Analyst needs to think of the flaw scenarios

Tool only validates the flaw

Tatebayashi-Matsuzaki-Newman (TMN) Protocol Analysis

- Used in digital mobile communication network
- Uses public key cryptosystem for uplink from the user to the key distribution center (KDC)
- Uses secret key cryptosystem for downlink channel
- Uses secret key cryptosystem for user to user communication with a new key for each session

Gus Simmon's Challenge

User Transforms

Request_Key(A,B:user, R1:key)
Respond_To_KDC(A,B:user, R2:key)
Get_Key(A,B:user, T:text)

KDC Transforms

Process_Request(A,B:user)
Return_Key(A,B:user, T:text)

Let

Enc(K,T) Dec(K,T)	Secret Algorithms
P-Enc(K,T) P-Dec(K,T)	Public Key Algorithms
e	KDC public key
d	KDC private key

Request_Key	A---B, P-Enc(e,R1)--->KDC
Process_Request	KDC---A--->B
Respond_To_KDC	B---A, P-Enc(e,R2)--->KDC
Return_Key	KDC---B, Enc(R1,R2)--->A
Get_Key	Dec(R1,Enc(R1,R2)) = R2

User Transforms
- Request_Key(A,B:user, R1:key)
- Respond_To_KDC(A,B:user, R2:key)
- Get_Key(A,B:user, T:text)

KDC Transforms
- Process_Request(A,B:user)
- Return_Key(A,B:user, T:text)

Cheater Transforms
- Cheater_Request(K:key)
- Partner_Response
- Compromise_Key(T:text)

Let

Enc(K,T)	Secret Algorithms
Dec(K,T)	
P-Enc(K,T)	Public Key Algorithms
P-Dec(K,T)	
e	KDC public key
d	KDC private key
C	Cheater
D	Partner
Cons	Agreed upon constant

Request_Key	A----B, P-Enc(e,R1)---->KDC
Cheater_Request	C----D, P-Enc (e,R3)*P-Enc (e,R1)---->KDC
Process_Request	KDC----C---->D
Partner_Response	D----C, P-Enc(e,Cons)---->KDC
Return_Key	KDC----D, Enc(R1*R3,Cons)---->C
Compromise_Key	

Return_Key KDC----D, Enc(R1*R3,Cons)---->C

Compromise_Key

Dec(Cons,Enc(R1*R3,Cons))/R3
Dec(Cons,Enc(Cons,(R1*R3))/R3
(R1*R3)/R3
R1


```
Axiom
  ∀T1,T2:text, K:key
    (P-Encr(K,T1)*P-Enc(K,T2)
       = P-Encr(K,T1*T2))
```



```
Axiom
  ∀T1,T2:text, K:key
    (P-Encr(K,T1)*P-Enc(K,T2)
       = P-Encr(K,T1*T2))

Axiom
  ∀K1,K2:key
    (Encr(K1,K2) = Encr(K2,K1)
```

Aslantest

A tool for symbolically executing Aslan specifications

Aslantest was used to test the TMN specification

"Analyzing Encryption Protocols Using Formal Verification Techniques," *IEEE Journal on Selected Areas in Communications*, vol. 7, No. 4, 1989

Kemmerer

"Three Systems for Cryptographic Protocol Analysis," *Journal of Cryptography*, vol. 7, No. 2, 1994

Kemmerer, Meadows, and Millen