Two Sides of Security

Peter Wild
Information Security Group
Royal Holloway, University of London

Thanks to:
Kenny Paterson, Keith Martin, Simon Blackburn, Martin Albrecht, Gaven Watson, Maura Paterson and Tuvi Etzion
Two Sides

Two issues in Information Security: Key Management and Protocol Failure.

1. Efficient Key Predistribution for Grid-Based Wireless Sensor Networks

2. Plaintext Recovery Attacks against SSH
Part One

Efficient Key Predistribution for Grid-Based Wireless Sensor Networks
Geographically distributed sensors with a requirement for communication.
Wildlife monitoring, natural resources, pollution, military applications.
Wireless communication with limited range in adhoc network.
Availability of data is important. Confidentiality and/or Integrity may be required – need for cryptographic channel.
Wireless Sensor Network

Restricted memory, restricted battery power, restricted computational ability, vulnerable to compromise.

Requirement for symmetric key agreement.
Sensor $\Psi$ communicates with other sensors within range $r$. 
Requirement for Light-weight Key Management

Fixed Location Information can be used to lower key storage requirement and improve resilience if sensors are lost/compromised.
Key Predistribution Scheme (KPS):
Nodes are assigned keys before deployment

- Nodes that share keys can communicate securely
- Two-hop path: nodes communicate via intermediate node
KPS in Grid-Based Network: Goals

• Enable any two neighbours to communicate securely (directly or via a two-hop path)
• Minimize Storage
• Be Resilient against Node Compromise

Observation: it is not necessary for two nodes to share more than one key.
KPS in Grid-Based Network: Goals

Each node shares a key with as many of its neighbours as possible

Restricting the extent to which each key is shared in case of a node (and its keys) being compromised
Costas Arrays

One dot per row/column

Vector differences between dots are distinct
Translates of Costas Arrays overlap in at most one point
Costas Array Translates
Costas Array Translates
Key Predistribution using Costas Arrays

Sensor: Cell in Square Grid
Key: Translate of Costas Array Pattern

Sensor assigned Key if a dot occupies the Cell in the Translate

Cells assigned Key A
Centre Cell holds Keys B, D, I.
Key Predistribution using Costas Arrays

Each Sensor stores $n$ Keys

Each Key is assigned to $n$ Sensors

Two Sensors share at most one Key

The distance between two sensors that share a key is at most $\sqrt{2(n-1)}$
Distinct-Difference Configuration

\( \text{DD}(m,r) \)

Generalization of Costas Array

• \( m \) dots are placed in a \( n \times n \) square grid
• The distance between any two dots is at most \( r \)
• Vector differences between dots are all distinct

\[
\text{DD}(5,8)
\]
Distinct-Difference Configurations

Can be used for Key Predistribution in the same way as Costas Arrays

More general than a Costas Array –
More flexible choice of parameters to reduce $r$ for given grid size $n$ and key storage $m$. 
DD(9,12)

http://www.isg.rhul.ac.uk/~martin/wsn.html
Part Two

Plaintext Recovery Attacks against SSH
Secure Shell or SSH is a network protocol that allows data to be exchanged using a secure channel between two networked devices.

Established over TCP/IP network (Transport Control Protocol, Internet Protocol)
SSH Session

Establish Connection

Agree Encryption Algorithm and Message Authentication Code (MAC) Algorithm

To provide Confidentiality and Integrity
Binary Packet Protocol

Messages are encoded into Packets transmitted in sequence during the session

Padding is appended to the message to produce a packet length 4 bytes short of a multiple of the block length (in bytes) of the cipher.

The length of the packet is encoded as a 32 bit value
The SSH BPP

- Encode-then-Encrypt&MAC construction.
- Sequence number is not sent on the wire.
- Packet length is encrypted to hide true length of SSH packets on the wire.
CBC mode encryption

\[ C_i = E_K(P_i \oplus C_{i-1}) \]
CBC mode decryption

\[ P_i = D_K(C_i) \oplus C_{i-1} \]
Initialization Vector IV

IV is agreed at start of SSH session for the first packet.

SSH uses a ‘chained IV’

– IV for a subsequent packet is the last ciphertext block $C_n$ from the previous packet.
– Effectively creates a single stream of data from multiple SSH packets.
The packet length field encodes how much data needs to be received before the MAC should be checked. Thus this field *must* be decrypted before the MAC is checked.
Plaintext Recovery

• The attacker intercepts a target ciphertext block from the SSH connection.
• The attacker sends the target block as the first block of a new SSH packet on the connection.
• The recipient will treat the first 32 bits of the plaintext corresponding to that block as the packet length field.
• If the attacker obtains information about the packet length then he obtains information about the original plaintext of the target block.
Side Channel Information

Once enough data has arrived, as determined by the packet length, the recipient checks the MAC. If this check fails then the connection is terminated with an error message. The number of blocks required to trigger the error message reveals the content of the 32-bit packet length field.
The Attack

The attacker feeds 4 random bytes (the size of the MAC) followed by random ciphertext blocks (16 bytes) into the SSH connection.

– One block at a time, waiting to see what happens with each new block.

With overwhelming probability the MAC check fails and an error message is returned.
Plaintext Recovery

Because of CBC mode, this reveals 32 bits of the target plaintext block.

Target ciphertext block: \( P_i = D_k(C_i) \oplus C_{i-1} \)

First block of new packet: \( P = D_k(C_i) \oplus IV \)

Thus, \( P = P \oplus C_i \oplus IV \)
Performance of the attack

The attack would succeed with probability 1, but would require the injection of $2^{27}$ random blocks (of 16 bytes) on average.

In practice, various sanity checks on the packet length field are performed by implementations and so not all packet lengths are possible.

“… implementations SHOULD check that the packet length is reasonable in order for the implementation to avoid denial of service and/or buffer overflow attacks.”
In the OpenSSH implementation, two sanity checks are carried out.

- Packet Length must be between 5 and $2^{18}$
- $4 + \text{Packet Length}$ must be a multiple of the Block Length 16.

When each of the checks fails, the SSH connection is terminated before the MAC check - and in subtly different ways.
Failure Differentiation

If the Packet Length check fails the SSH connection terminates and returns an error message.

Else If the Divisibility check fails then the SSH connection terminates with no error message. Observed via TCP FIN packet.

Else the SSH connection waits for further Blocks (and eventually checks the MAC).
Attack Failure

If the SSH session terminates with a Packet Length error message then the attack reveals virtually no information about the target plaintext block.
Packet Length Check

If the session terminates without error message or waiting for data then the Packet Length check was passed, i.e. the first 14 bits are 0s

Success occurs with probability $2^{-14}$ (assuming that $C_{i-1}$ is random) and reveals 14 bits of the target plaintext block.
Waiting State

If the SSH session enters the Waiting for Blocks state then the first 14 bits of the Packet Length are 0s and the last 4 bits are 1100 (by the divisibility condition).

This reveals 18 bits of the target plaintext block and success occurs with probability $2^{-18}$. 
MAC Check Failure

In the Waiting state the attacker injects up to $2^{18}/16 = 2^{14}$ Blocks before the MAC check fails, revealing 32 bits of the target plaintext block.
An Iterated Attack

If a plaintext is repeated at a fixed position in SSH packets over multiple connections, then the attack can be iterated to boost the success rate.

Some clients automatically reconnect on session termination.
Application to password extraction.
Random IVs per Packet

The attack applies even if a fresh random IV is used for each packet.

Assuming that this is sent in clear on the connection, an iterated attack becomes more effective.
An Iterated Attack

1. Watch for IVs with distinct first 14 bits. One will pass the Packet Length check yielding 14 bits of the target plaintext

2. Watch for IVs with these first 14 bits and distinct last 4 bits. One will pass the divisibility check yielding 18 bits of the target plaintext

3. Inject Blocks one by one until the MAC check fails yielding 32 bits of the target plaintext
MAC check error reveals the amount of data expected in the packet corresponding to the Packet Length field.

CBC mode encryption reveals the plaintext of the target block:

- Encrypted length field
- CBC mode
- Reliable transport (injecting block by block)
- Error signalling
The End

Thank you!