Crypto Rodeo

A round up of some interesting recent events in cryptography.

amy beth corman
cryptogirl@snorkel.rtfm.net.au

Overview

- Quantum computing, cryptography and key exchange
- SHA-1 hash function break
- Side channel attack on AES
- Blue tooth device pairing PIN crack

Quantum computing

A bit is not a qubit

- A bit is discrete either 1 or 0
- A qubit is a both a 1 and a 0 until measured (after computation)

- Serious difficulties in engineering quantum computers - current best implementation has 7 qubits.

Quantum computing and classical cryptography

- Shor's factoring algorithm - probabilistic, polynomial time
- Digital cryptographic systems that depend on factoring (or the related discrete logarithm problem) being hard:
  - RSA
  - ElGamal
  - DSA
  - Diffie-Hellman key exchange
  - Blum-blum shub pseudo-random number generation

Examples

- To factor a number we need about twice as many qubits as the number takes to express in bits.
  For example: to factor 15 into 5x3 you'd need 7 qubits
  To factor a typical RSA modulus of 1024 bits you'd need more than 2000 qubits.
- In addition to faster factoring, brute force searching for keys would also be significantly faster.
  For example: to brute force a 128 bit AES key, would currently take an average of 2^{64} guesses
  with a quantum computer it would take on the order of $2^{\sqrt{128}}$ (which is about 2^{12}) guesses.

Quantum cryptography

- Even though we have found algorithms that make some hard problems easier with quantum computing, it is not clear yet whether all hard problems will be easier.
- It is expected that classical computers will always have far more bits than quantum computers have qubits. This means that by using larger keys we can put them out of the reach of quantum computers.
- It is also possible for us to depend on information theoretical security (one-time pad) instead of computational complexity security.

Quantum key exchange

- Quantum physics can be used to transmit keys over a direct link such that it is impossible to eavesdrop on the key exchange and not be detected.
- Three basic methods for this:
  - Photon polarisation 1 km
  - Quantum entanglement 4 km
  - Single photon interference fringe 10 km
- Data rate 20,000 bits/sec. which is 2.4kb/sec.
- Although tampering is detected it can cause denial of service
- If this is going to be used for a one-time pad you still need a truly random source!

Cryptographic hash function properties

- Cryptographic hash functions are one-way functions with fixed length output.
- It is important that they have the following properties:
  - Computationally easy to calculate the hash of a value.
  - Computationally difficult to find a value which maps to a particular hash.
  - Computationally difficult to find two or more values which map to the same hash (this is called a collision).
- Collisions must exist because we are mapping a very large range of input to a fixed length, finite range of output.

Cryptographic hash function uses

- Cryptographic hash functions are usually used to ensure data integrity.
- Many signature schemes generate a hash of the data to be signed and sign only the hash.
- Cryptographic hash functions are also used to prove the possession of data without revealing the data itself.

SHA-1 Break

- In cryptography a break is any attack which takes less time than brute-forcing the function. Since SHA-1 is 160 bits, brute-forcing it takes 2^{80} operations on average.
- Feb. 2005 - Wang, Yin & Yu find an attack that can find collisions in 2^{69}
- Aug. 2005 - Wang, Yao & Yao announce improvement on the above attack to 2^{63}
- Although 2^{63} is still difficult, more improvements could be made in the future weakening the function even further.

What about other functions?

- List of common cryptographic hash functions and their length:

Name	*Length in bits*
MD5	128
SHA-1	160
SHA-256	256

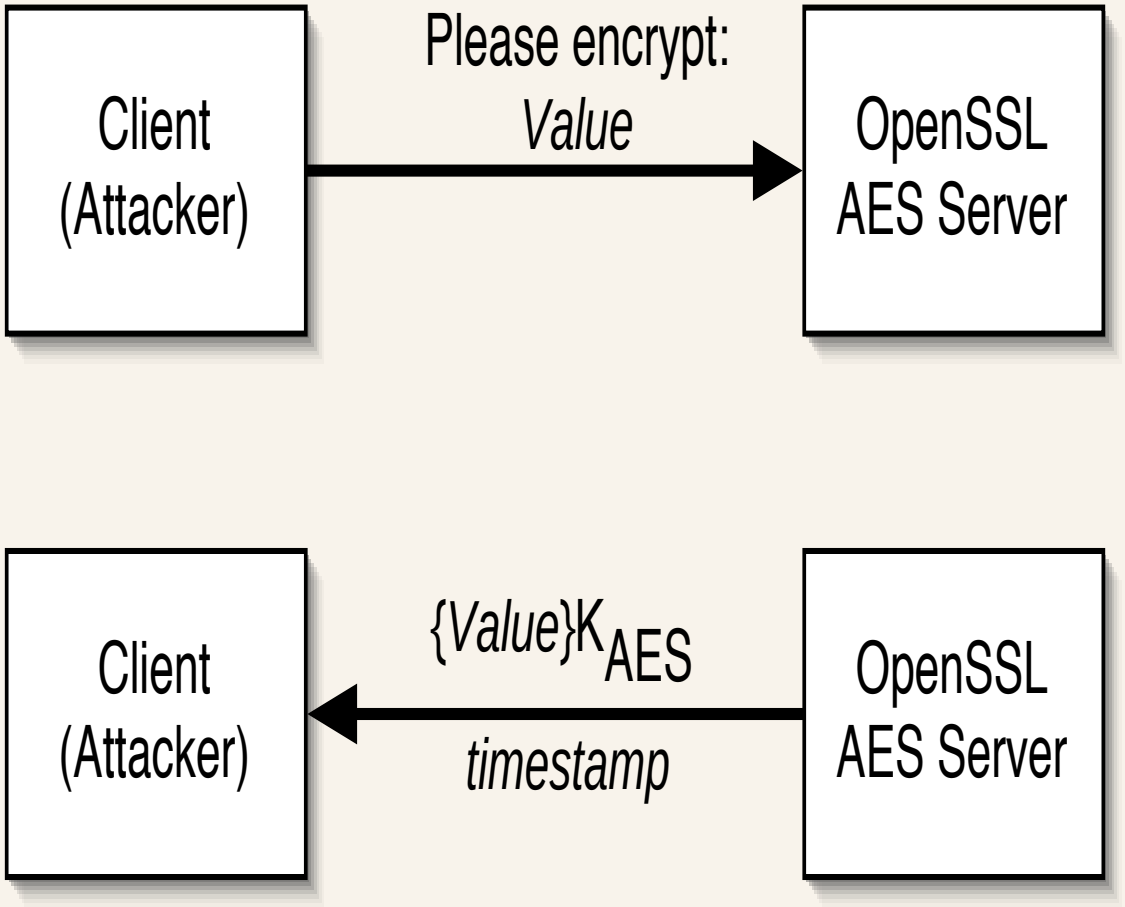
- MD5 is very badly broken and should not be used for anything security critical!
  - Two certificates with different public keys and the same hash have been constructed showing the practical implications of the attack.
  - Can find a collision in a few hours on a normal computer.

What should we use now then?

- Moving to SHA-256, SHA-384, or SHA-512 does provide some breathing room. It is not clear yet if the attacks are transferable to SHA-2 variants.
- Recent breaks have stimulated hash function research, probably see new functions proposed soon.
- You could switch to a more obscure cryptographic hash function but these have not received the same scrutiny and might be equally flawed.

Cache timing attack on AES

- Daniel J. Bernstein (djb) announced earlier this year that he could successfully extract a full AES key based on a cache timing attack (with known plaintext).
- AES and other block ciphers (including all 15 of the other AES candidates) are vulnerable to timing attacks because fast implementations leak secret information.
- It is possible to implement AES such that it takes constant time but this is too slow for practical use.


Client
(Attacker)
Please encrypt:
Value
OpenSSL
AES Server
Client
(Attacker)
{Value}K AES
timestamp
OpenSSL
AES Server

How does the attack work? (Simplified)

- Tested against OpenSSL which can use AES for a session key
- There is a variable index array lookup early in the computation which uses the key bitwise binary exclusive or (now referred to as xor $\otimes$) with the plaintext as the index.
- Attacker notes time it takes for many different plaintexts. The attacker then knows both:
  - The array index value (for a particular byte) that takes the most time.
  - The plaintext value (for the same byte) that takes the most time.
- Since $index = plaintext \otimes key$, we can derive the key by $key = index \otimes plaintext$.

How do we fix it?

- Design block ciphers that are both fast and take constant time. Some of these exist already:
  - Tiny Encryption Algorithm by Wheeler and Needham
  - SHA-256 by NIST
  - Helix by Ferguson, Whiting, Schneier, Kelsey, Lucks & Kohno
  - Salsa20 by djb
- More research needs to be done in this area.

Bluetooth pairing PIN crack

- Applies to Bluetooth in security mode 3
- April 2004 - Whitehouse (@Stake) outlines PIN cracking attack
- June 2005 - Shaked and Wool implement attack with some algebraic optimisations that speed up by factor of 10

Overview of attack

- When two bluetooth devices want to talk to each other they must perform a pairing (bonding) operation. This pairing tests whether the same PIN number has been entered into each device.
- By observing the pairing process and storing the packets, it is possible to perform an offline brute force attack on the PIN.
- The other keys derived in the pairing process are also vulnerable because K_{init} is derived from the PIN and K_{ab} (the link key) is protected by K_{init}.

Details of protocol

- A $\rightarrow$ B: IN_RAND (128 bit random number)
  - Both derive K_{init} using E22 and the PIN, IN_RAND and B's Device Address
- A $\rightarrow$ B: $K_{init} \otimes LK\_RAND_A$ (128 bit random number)
- B $\rightarrow$ A: $K_{init} \otimes LK\_RAND_B$ (128 bit random number)
  - Both derive K_{ab} using E21 and LK_RAND_{AandB} and the Device Addresses
- The link key K_{ab} is stored and used for further communications

Details of optimisation

- Optimisation involves two insights:
  - viewing the linear parts of SAFER+ as a 16x16 matrix and exploiting structure within the matrix
  - exploiting structure within the matrix to use `shift left` for binary multiplication by 2
- This reduces the operations needed for the matrix from 512 (256 multiplication, 256 addition) to 200 (40 shift left, 128 addition, 16 load and 16 store).
- Cracks 4 digit PIN in 0.063 sec. and a 7 digit PIN in 76 sec. on Pentium IV 3 GHz

What can you do to protect yourself?

- Choose a bigger PIN (can be up to 8 bytes long)
- Pair your devices in a Faraday cage
- Be suspicious of repeated pairings, once two devices have negotiated a link key they should both store this key and use it (not need to re-pair). If you're not certain, don't enter the PIN again.

References

- Quantum computing, cryptography and key exchange
  - http://www.qubit.org
- SHA-1 Hash function break
  - http://cryptome.org/wang_sha1_v2.zip
- Side channel attack on AES
  - http://cr.yp.to/antiforgery/cachetiming-20050414.pdf
- Blue tooth device pairing PIN crack
  - http://www.cansecwest.com/csw04/csw04-Whitehouse.pdf
  - http://www.eng.tau.ac.il/~yash/shaked-wool-mobisys05/index.html

Melbourne, Australia
February 8-10, 2006

The 4th annual SECURECon features 1 day of tutorials/workshops and 2 days of FREE talks.

http://securecon.unimelb.edu.au