Tutorial I
Computer Security Module

Mike Just

School of Informatics
University of Edinburgh

February 2010

Purpose of Tutorial

- More time (detail) on some areas of Computer Security
- Allow for more interaction with a smaller class
  - More interaction
  - More time for questions
- Method for tutorial
  - Learn additional, general concepts related to Tutorial themes
  - Interactively review answers to Practical I questions

General Lessons

- Things aren't always (as secure) as they seem
- Don't believe everything you're told/sold
- Be careful taking *short cuts*
- Anticipate all possible behaviour

Tutorial I Themes

1. Hashes, MACs and Integrity
2. PRNGs and Key Entropy
3. Protocols

Tutorial I Themes

1. Hashes, MACs and Integrity
2. PRNGs and Key Entropy
3. Protocols

Hashes, MACs and Integrity (2)

- Question 1
  - Short-cuts in MAC computation can prove costly

Hashes, MACs and Integrity (3)

- Question 2
  - Assumed security can be deceiving

Hashes, MACs and Integrity (4)

- Question 3
  - Unexpected usage means unintended side effects

Tutorial I Themes

1. Hashes, MACs and Integrity
2. PRNGs and Key Entropy
3. Protocols

PRBGs

- Talked of sources of (pseudo)random bits in class
- Also important to *test* that a sequence of bits is “suitably random”
- Ideally, a sequence of bits is suitable if the sequence is *indistinguishable* from a random sequence
  - If no polynomial time algorithm can distinguish, then PRBG will pass all *polynomial-time statistical tests*
  - If no polynomial time algorithm, with first *t* bits, can predict the *(t+1)*st bit, then PRBG passes *next-bit test*
  - Pass next-bit ↔ Pass all poly-time statistical tests

PRBGs (Next-Bit Test Example)

- Generate pseudo-random bits with RSA
  - Random seed x_0
  - For i from 1 to t
    - $x_i = (x_{i-1})^e \bmod n$
    - $z_i = \text{lsb}(x_i)$
  - Pseudo-random bits are z_i, $i=1,\ldots,t$
  - Predicting next-bit is equivalent to breaking RSA

PRBGs (Statistical Tests)

- High-level: Ensure #*0*s = #*1*s
- Golomb's randomness postulates [HAC] (for sequence *s*)
  1. #*0*s ≈ #*1*s
  2. Measures of repeated *0*s and *1*s
     - At least half the runs have length 1, at least one-fourth have length 2, at least one-eighth have length 3, etc., as long as the number of runs so indicated exceeds 1.
     - Moreover, for each of these lengths, there are (almost) equally many gaps (run of 0's) and blocks (run of 1's).

PRBGs (Statistical Tests ...)

- Golomb's randomness postulates [HAC] (for sequence *s*)

  3. For autocorrelation function C(t) and some integer K

  $$N\,C(t) = \text{Sum}\,(i=1,N-1)\,\{(2s_i - 1)(2s_{i+t} - 1)\} = N \quad \text{if } t=0$$

  $$K, \text{ if } 1 \le t \le N-1$$

- Intuitively, (3) measures the difference between the sequence *s*, and the sequence *s* shifted by *t* positions

-

s_i	s_{i+t}	$(2s_i - 1)(2s_{i+t} - 1)$
0	0	1
0	1	-1
1	0	-1
1	1	1

PRBGs

- Question 1
  - Getting something from nothing (with a cost)

Key Entropy

- *Actual key size*: Physical space for key storage
- *Effective key size*: Work required to guess key
- *Entropy* (information content / uncertainty) is often used as a measure of effective key size

Key Entropy (2)

- Let *X* be a random variable which takes on a finite set of values $x_1, x_2, \ldots, x_n$ with probability $P(X=x_i) = p_i$ $0 \le p_i \le 1$, where $1 \le i \le n$ and where $Sum(i=1,n) \{p_i\} = 1$.
- The entropy of *X* is a mathematical measure of the amount of information provided by an observation of *X*. Equivalently it is the uncertainty about the outcome before an observation of *X*. The entropy, or uncertainty, of *X* is mathematically defined as

$$H(X) = - [Sum(i=1,n) \{p_i \ lg(p_i)\}] = [Sum(i=1,n) \{p_i \ (1/ lg(p_i))\}]$$

- Examples
  - Set of 8 values with p=1/8 gives H(X) = 3
  - Set of 8 values with p1=1/2 and p2=p3=1/4 gives H(X) = 1.5

Key Entropy (3)

- Question 2 – RSA vs AES key size
  - Size doesn't matter

Key Entropy (4)

- Question 3 – Selling “Snake Oil”
  - Don't believe everything you're told/sold

Tutorial I Themes

1. Hashes, MACs and Integrity
2. PRNGs and Key Entropy
3. Protocols

Protocols

- Protocols are challenging to design properly
  - Similar to previous Hash and MAC design failures
- Must assume complete freedom of attacker to perform various actions, outside the normal, expected protocol execution
  - Middle-person attack
  - Reflection (“bounce-back”) of protocol messages
  - Interleaving of protocol messages
- Multi-party protocols are often complicated, difficult to assess, and ambiguous wrt responsibility (e.g., Chip & PIN)

Protocols

- Question 1
  - Just because you encrypt, doesn't mean it's secure

General Lessons (Again)

- Things aren't always (as secure) as they seem
- Don't believe everything you're told/sold
- Be careful taking *short cuts*
- Anticipate all possible behaviour