Lecture 7 - Applied Cryptography

CMPSC 443 - Spring 2012
Introduction Computer and Network Security
Professor Jaeger

www.cse.psu.edu/~tjaeger/cse443-s12/
Applied Cryptography

• Applied Cryptographic is the art and science of using cryptographic primitives to achieve specific goals.
  – The use of the tools is called a construction
  – e.g., encryption (achieves confidentiality)

\[ E(k, d) = c \]

• Much of network and systems security is based on the integration of constructions with the system.
Some notation …

• You will generally see protocols defined in terms of exchanges containing some notation like
  – All players are identified by their first initial
    • E.g., Alice=\( A \), Bob=\( B \)
  – \( d \) is some data
  – \( \text{pw}^A \) is the password for A
  – \( k^{AB} \) is a symmetric key known to A and B
  – \( A^+, A^- \) is a public/private key pair for entity A
  – \( E(k, d) \) is encryption of data \( d \) with key \( k \)
  – \( h(d) \) is the hash of data \( d \)
  – \( S(A^-, d) \) is the signature (using A’s private key) of data \( d \)
  – “+” is used to refer to concatenation
Providing Authenticity/Integrity

• Most of what we have talked about so far deals with achieving \textit{confidentiality} using encryption.

• However, and often equally or more important property is authenticity
  – \textit{authenticity} is the property that we can associate a data with a specific entity from whence it came/belongs to
  – Integrity is the property that the data has not been modified
  – Note that \textit{integrity} is a necessary but not sufficient condition for authenticity (why?)

• \textbf{Q:} How do we use cryptography for these goals?
Hashed Message Authentication Codes

• HMAC
  – Authenticates/integrity for data d in symmetric key system
  – Uses some key $k$ and hash algorithm $h$
  – To simplify,

$$hmac(k, d) = h(k \cdot d)$$

• Why does this provide authenticity?
  – Cannot produce $hmac(k, d)$ unless you know $k$ and $d$
  – If you could, then can break $h$
  – Exercise for class: prove the previous statement

• Used in protocols to authenticate content
Using HMACs

• Assume I am going to send you a random number $r$ over a network, and that we share a key $k$

• I could send you

$$E(k, r)$$

• .... over the network.

• **Q:** Is there anything wrong with this approach?
  – *Hint:* think of an active attacker.
Using HMACs (cont.)

• An active attacker could replace the value \( E(k,r) \) with any random bits and I would not know it.
  – The central point is that I cannot tell one decrypted random value from another
  – Attacker can change the cipher, but not know the result (e.g., confidentiality is preserved)

• A fix:

\[
E(k,r), \text{HMAC}(k,r)
\]

• Now, the adversary cannot generate a HMAC that will properly validate without knowing \( k \)

• Extra credit: how would you prevent a \textit{replay attack}?
Digital Signatures

- Models physical signatures in digital world
  - Association between private key and document
  - … and indirectly identity and document.
  - Asserts that document is **authentic** and **non-repudiable**

- To sign a document
  - Given document \(d\), private key \(k^-\)
  - To simplify,

\[
S(k^-, d) = E(k^-, h(d))
\]

- Validation
  - Given document \(d\), signature \(S(d)\), public key \(k^+\)
  - To simplify,

\[
D(k^+, S(d)) = h(d)
\]
Using Signatures ... 

• Assume you want to certify/endorse some data d
• You want anyone to be able to validate the item after the fact, even when you are not around
• Just sign the document, leave it with your public key

\[ d, S(d), k^+ \]

• Of course, then you have the problem of securely identifying which key belongs to you.
  – This is the purpose of a public key infrastructure, covered in future lectures.
• This is the approach taken in most electronic commerce systems
  – e.g., signing receipts, transactions, etc. ...
Meet Alice and Bob ....

- **Alice** and **Bob** are the canonical players in the cryptographic world.
  - They represent the end points of some interaction
  - Used to illustrate/define a security protocol

- Other players occasionally join  ...
  - Trent - trusted third party
  - Mallory - malicious entity
  - Eve - eavesdropper
  - Ivan - an issuer (of some object)
Using hash values as authenticators

- Consider the following scenario
  - Alice is a teacher who has not decided if she will cancel the next lecture.
  - When she does decide, she communicates to Bob the student through Mallory, her evil TA.
  - She does not care if Bob shows up to a cancelled class
  - Alice does not trust Mallory to deliver the message.

- She and Bob use the following protocol:
  1. Alice invents a secret $t$
  2. Alice gives Bob $h(t)$, where $h()$ is a crypto hash function
  3. If she cancels class, she gives $t$ to Mallory to give to Bob
     - If does not cancel class, she does nothing
     - If Bob receives the token $t$, he knows that Alice sent it
Hash Authenticators

• Why is this protocol secure?
  – $t$ acts as an authenticated value ( authenticator) because Mallory could not have produced $t$ without inverting $h()$
  – Note: Mallory can convince Bob that class is occurring when it is not by simply not delivering $t$ (but we assume Bob is smart enough to come to that conclusion when the room is empty)

• What is important here is that hash preimages are good as (single bit) authenticators.
• Note that it is important that Bob got the original value $h(t)$ from Alice (i.e., was provably authentic)
Hash Chain

• Now, consider the case where Alice wants to do the same protocol, only for all 26 classes (the semester)
• Alice and Bob use the following protocol:
  1. Alice invents a secret $t$
  2. Alice gives Bob $H^{26}(t)$, where $H^{26}()$ is 26 repeated applications of $H()$.
  3. If she cancels class on day $d$, she gives $H^{(26-D)}(t)$ to Mallory, e.g.,
     - If cancels on day 1, she gives Mallory $H^{25}(t)$
     - If cancels on day 2, she gives Mallory $H^{24}(t)$
     .......
     - If cancels on day 25, she gives Mallory $H^{1}(t)$
     - If cancels on day 26, she gives Mallory $t$
  4. If does not cancel class, she does nothing
     – If Bob receives the token $t$, he knows that Alice sent it
Hash Chain (cont.)

• Why is this protocol secure?
  – On day \( d \), \( H^{(26-d)}(t) \) acts as an authenticated value (authenticator) because Mallory could not produce \( t \) without inverting \( H() \) because for any \( H^k(t) \) she has \( k>(26-d) \)
  – That is, Mallory potentially has access to the hash values for all days prior to today, but that provides no information on today’s value, because they are all post-images of today’s value
  – Note: Mallory can again convince Bob that class is occurring by not delivering \( H^{(26-d)}(t) \)

• Important: chain of hash values are ordered authenticators

• Important that Bob got the original value \( H^{26}(t) \) from Alice directly (was provably authentic)
Key Distribution

• Key Distribution is the process where we assign and transfer keys to a participant
  – Out of band (e.g., passwords, simple)
  – During authentication (e.g., Kerberos)
  – As part of communication (e.g., skip-encryption)

• Key Agreement is the process whereby two parties negotiate a key
  – 2 or more participants
  – E.g., Diffie, Hellman

• Typically, key distribution/agreement occurs in conjunction with or after authentication.
  – However, many applications can pre-load keys
Simple Key Distribution

• (simplified view) Assume you have 4 participants
  – Distribute 3 out of 4 total keys to each participant
  – Any two participants can generate a unique key

\[\begin{align*}
  [k^2 \ k^3 \ k^4] \\
  [k^1 \ k^2 \ k^3] \\
  [k^1 \ k^3 \ k^4] \\
  [k^1 \ k^2 \ k^4]
\end{align*}\]

- How: pick XOR of the keys that are not held by the other participants

  • E.g., Assume A and C want to communicate
    – \(k^{AC} = k^2 \ XOR \ k^4\)
Simple Key Distribution (cont.)

• Why does this work?
  – B cannot eavesdrop because it does not know $k^2$
  – D cannot eavesdrop because it does not know $k^4$

• General construction
  – Create large set of keys \{k^1, k^2, \ldots, k^n\}
  – Give precisely 1/2 of keys to each participant
    • Make sure that no two sets of assigned keys are complements
  – Any two participants can communicate
  – The more keys you have, the more likely it is that two participants can generate a key

• Q: Can you attack this system?
Simple Key Distribution (cont.)

- **Collusion**: two or more adversaries attempt to circumvent the security services
- In the case of simple key distribution, if several of the participants are evil and collude, then they have the full set of keys and the game is up
  - E.g.,
    \[
    [k_1 \ k_2 \ k_3] + [k_1 \ k_3 \ k_4] = [k_1 \ k_2 \ k_3 \ k_4]
    \]
  - Topic area: simple key distribution is use in severely resource constrained environments (e.g., sensor networks) because of the low performance requirements
    - However, storage is often a problem