Week of September 26, 2017

Question 1  Certificate Authorities and Certificates  
(30 min)

An agreement has been reached where Trusty Tristan was chosen to serve a special role: he will sign anyone’s public key with their name if they can prove that it is their key.

\[ S_{\text{Name}} = \text{SIGN}_{K_d^T}(\text{Name}||K_{\text{Name}}^e) \]

* Names can be abbreviated by initial when using them as sub/superscripts

To sign these keys, Tristan distributed his public key, \( K_e^T \), in person to everyone at the meeting where he was chosen for this role. They have a website, www.keys-keys-everywhere.gov, which can host public keys and other information, but anyone can post on this site, so the information alone cannot be trusted.

(a) Bob receives a message \( M \) along with a signature, \( S_M \). The message says “Hey this is Alice, check out my website at www.totally-safe.berk.ga” Now, Bob is understandably apprehensive of this URL and wants to verify that this message was in fact sent from Alice, but this is the first time they have ever talked, and he doesn’t know her public key.

What could Bob do to verify if this message was in fact sent by Alice?

**Solution:** Bob will look up Alice’s public key on www.keys-keys-everywhere.gov to get what is supposedly Alice’s public key, \( K_e^A \), along with Tristan’s signature of this public key, \( S_A \). Now Bob will verify that this public key is in fact Alice’s by verifying the signature with Tristan’s public key:

\[ \text{VERIFY}_{K_e^T}(S_A, \text{Alice}||K_e^A) \]

If it checks out, we now are sure of Alice’s public key which we can use to check the signature on the message sent to us:

\[ \text{VERIFY}_{K_e^A}(S_M, M) \]

This is the general procedure when visiting a website using HTTPS which you will learn more about later: a website provides a certificate which verifies that they own the site and their public key. We will check the signature on the certificate that the Certificate Authority has provided to verify that is valid, then we can be sure of the website’s public key which will allow us to exchange symmetric keys before we can start communication.
(b) Considering how Bob determined if the message was actually sent from Alice or not, why is it important that Tristan is in fact trustworthy?

**Solution:** If Tristan was actually malicious, he may sign keys that don’t actually belong to people.

\[ S_{Alice} = \text{SIGN}_{K^T_d} (Alice||K^e_{Mallory}) \]

In this example, Tristan has signed a key owned by Mallory, but with Alice’s name on it. Then Mallory could use this key to pretend she was Alice when sending a message such as telling Bob what account to send money to, or what website URL to visit. This is why it is so important who we allow to sign certificates on our computers (and who we allow to put those signing certificates on our computer, such as who controls the OS like Apple or Microsoft).

(c) Now, Calvin missed the meeting where Tristan was nominated and gave out his key. So he asks for Tristan’s key, but Mallory sees this requests.

What does Mallory do?

**Solution:** Mallory sends her public key to Calvin and now whenever Calvin may receive a message, Mallory has two options. She can sign their actual public key and act exactly as Tristan would, or she can impersonate anyone by signing a new key she generates using the key pair that she provided to Tristan.

(d) Why is it important in this scheme that we use public key signatures instead of a keyed MAC when both guarantee integrity?

**Solution:** An important property when talking about integrity is non-repudiation which is the property that an author of a message cannot deny that they were the one who authored it. Public key signatures provide universal non-repudiation which means that anyone can confirm the author of a message because verify uses the public key. The difference with a keyed MAC is that only people who have the key have the non-repudiation properties, and if everyone knows the key, then anyone can forge a MAC. As a side note, non-repudiation is in fact a legal term, but this form won’t hold in court.

(e) Mallory wore a disguise and has tricked Tristan into signing her key as if it belonged to Alice. We know this is a bad scenario, and we have to do something to stop Mallory from impersonating Alice. Propose what could be done to revoke this bad signature.
Solution: Because we have been modeling certificates and certificate authorities, we will use them to draw inspiration for this answer. A common problem is a compromised certificate whether the private key is leaked, or something similar as what is presented in the problem. Now the safest option, nuke them all isn’t realistic because everyone will flood Tristan for new signatures which will destroy his ability to churn out new certificates (availability is a real issue when considering the scale of the internet), and requires that everyone can receive a new public key that they know belongs to Tristan which can be hard if we can’t trust his current public key. There are two solutions that work in tandem: expiration dates and a revocation list.

Expiration dates:
+ Fail safe, doesn’t require detection
+ Staggered expirations balance load of having to renew certs
- Can’t immediately revoke a known compromised cert
  - Short time-to-live would fix, but sacrifice availability of authority

Revocation lists:
+ Can immediately revoke a known compromised cert
- Could get quite lengthy as accumulate more out-of-date certs
- Requires download to update

When working together, we can provide long expiration dates to more secure/safe domains like google.com versus evil.com so maybe we don’t need to renew Google’s cert as often because we trust their credentials and security more than Evil’s. Now we can also provide longer expiration periods because we can immediately revoke by adding a cert to the revocation list (we still must maintain realistic expiration dates, otherwise there would be no point to having them). Finally, the revocation lists only needs to keep certs that haven’t already expired on it, so it can remain relatively short.
Question 2  **Facebook Messenger** *(20 min)*

Rather than use a pure peer-to-peer protocol, Facebook’s communication scheme places a Facebook server between two clients. In order to prevent eavesdropping (even from Facebook itself) and provide a very limited form of non-repudiation in order to verify abuse claims, Facebook uses a combination of public-key cryptography and HMACs. For a hypothetical sender and receiver, Sendy Cindy and Receiving Ricky respectively, the protocol functions as described below:

**Cindy**

1. Cindy queries Facebook’s servers for Ricky’s public key, \( K_{pub} \)
2. Cindy generates a random key, \( K_{rand} \)
3. Cindy encrypts \( M \) and \( K_{rand} \) using \( K_{pub} \)
   - \( C = E_{K_{pub}} \{M, K_{rand}\} \)
4. Cindy computes a MAC on \( M \) using \( K_{rand} \)
   - \( MAC = HMAC_{K_{rand}} \{M\} \)
5. Cindy sends \( X = \{C, MAC, To = Ricky\} \) to Facebook

**Facebook**

1. Upon receipt of \( X \), Facebook adds additional information to get \( X' \)
   - \( X' = \{C, MAC, To = Ricky, From = Cindy, Time = Now, FBMAC = HMAC_{K_{FB}} \{MAC, From = Cindy, To = Ricky, Time = Now\}\} \)
   - \( K_{FB} \) is never shared with anyone
2. Facebook then sends \( X' \) to Ricky

**Ricky**

1. Upon receipt of \( X' \), Ricky extracts \( C \) and decrypts it using his private key
2. Ricky verifies MAC using \( K_{rand} \)
3. If the message is abusive, Ricky sends an abuse report, \( A \), to Facebook
   - \( A = \{M, K_{rand}, MAC, From = Cindy, To = Ricky, Time = Now, FBMAC\} \)

**Facebook**

1. When Facebook receives \( A \), it first computes FBMAC again using \( K_{FB} \)
   - \( FBMAC' = HMAC_{K_{FB}} \{MAC, From = Cindy, To = Ricky, Time = Now\} \)
   - If \( FBMAC' \neq FBMAC \), reject
2. Facebook then computes and verifies MAC with \( K_{rand} \)
   - \( MAC = HMAC_{K_{rand}} \{M\} \)
3. Having verified the contents and sender of the abusive message, Facebook can act
(a) Who is able to validate Ricky’s claim of abuse? Why?

**Solution:** Only Facebook. Facebook is the only entity with $K_{FB}$

(b) What if Facebook used a signature instead of an HMAC? Who would be able to verify then?

**Solution:** Anyone with Facebook’s public signing key would be able to verify the message. Since one of the goals of the protocol is to prevent the receiver from publicly accusing a sender of abuse, this is problematic.

(c) For this protocol to work, Facebook needs to be trusted by both the sender and receiver. Suppose Facebook is taken over by a malicious entity. Which of the following could Facebook accomplish? Assume Facebook’s key server hasn’t been compromised.

(a) Spoof a message to Ricky

(b) Decrypt a message from Cindy to Ricky

**Solution:**

(a) Yes

(b) No

(c) In the provided example, Cindy generates a new random key every time she wants to send a message. What if she reused a key? Does the message lose any valuable properties?

**Solution:** Loses confidentiality, since a duplicate message would have the same HMAC. Facebook can now tamper with the contents of the message if the original had been reported for abuse, since $K_{rand}$ had to be shared with Facebook.
Question 3  **OAEP**  

When talking about RSA, there’s actually a few common attacks: because of a low $e$ the actual output space for small messages ($m^e$) is strictly less than $n$ which means taking the $e$-th root over the integers could be possible, if we send the message to too many people that share the same $e$ ($e$ or more recipients) then it is possible to decrypt using the Chinese remainder theorem, or because RSA is deterministic an attacker can launch plaintext attacks quite easily. Luckily, we have a possible answer:

$$\begin{array}{c}
M \\
\oplus \\
H \\
\oplus \\
H \\
X \\
\end{array} \quad \begin{array}{c}
r \\
\oplus \\
Y \\
\end{array}$$

Above is a slightly simplified feistel network modeling Optimal Asymmetric Encryption Padding (OAEP). Here, $r$ is generated randomly and $M$ is our message that we want to send whereas $X || Y$ is what we will actually encrypt using RSA.

(a) Given $X$ and $Y$, how would you recover the original message $M$?

**Solution:**

$$r = H(X) \oplus Y$$

$$M = H(r) \oplus X$$

(b) What purpose does $r$ actually serve in this scheme? (Think back to block ciphers)

**Solution:** It is acting similarly to a nonce or initialization vector so that the same message will not produce the same ciphertexts, and when configured correctly can prevent chosen plaintext attacks. It also could be used to prevent replay attacks in some small way by monitoring the occurrence of previous $r$ values, but because of its usually small bit-length this isn’t perfect.