Winter 2018 CS 485/585 Introduction to Cryptography

Lecture 18

Portland State University
Lecturer: Fang Song

Mar. 8, 2018

Draft note. Version: March 8, 2018. Email fang.song@pdx.edu for comments and corrections.

Agenda

- (Last time) Hash-based signature, RO, OAEP, FDH
- PKI, TLS, real-world attacks
- Final Review
- Quiz 4

Digital certificate and PKI

We have seen many nice features of public-key cryptography:

- Key-exchange: two users can agree on a secret key via public communication and without pre-shared secret.
- public-key encryption: no need of pairwise key; anyone can encrypt with the receiver's public key.
- digital signature: authentication that is publicly verifiable, non-reputable, etc.

However, PubKE and DS rely on safely distributing the public keys.[1]

So, how to make sure the right public keys are distributed? This can be solved by PKC, digital signature, more specifically. The key gadget is a *digital certificate*: a signature that binds a public key to some identity.

$$\mathsf{cert}_{A\to B} := S_{sk_A}(\text{“Bob's public key is } pk_B\text{”})\,.$$

Namely A certifies that Bob has public key pk_B, and anyone who knows Alice's public key can verify and accept Bob's valid public key.

But how to get Alice's pubkey, and why should we trust Alice in the first place? There is no perfect resolution to this question, and we rely on a *public key infrastructure (PKI)* to get a workable solution. [2]

Some main models of PKI:

[1] For instance, if an attackers fools you to use his pub-key as Amazon's public key, then it can decrypt all your information (credit card etc.). Similarly, if an attacker fools you about Apple's public key, it can sign a malicious update package using its own key and inject malware on your phone.

[2] A PKI is a set of roles, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates and manage public-key encryption.

- Certificate authorities. Everyone trusts one or a group of authorities, who are in charge of certifying and maintaining public keys for all. This can be a hierarchy of CAs (root CA, domestic CA, etc.). Some popular ones: Comodo, Verisign (acquired by Symantec), DigiCert.
- Web of trust. This model is adopted in the email encryption application PGP (Pretty Good Privacy)[3] Each user can issue certificates for anyone else, and also collects certificates for its own public key. For instance, Alice may have $\mathsf{cert}_{C\to A}, \mathsf{cert}_{D\to A}, and \mathsf{cert}_{E\to A}$. She can send all to Bob. Suppose Bob already has pk_C and pk_E. He can verify these two, and decide if he wants to accept Alice's public key. He can assign different trust level to Charlie and Elli (e.g., Bob totally trusts Elli and accepts certificate issued by Elli with certainty, or Bob only accepts a public key if both certificates by Charlie and Elli verify). These can also be store in a central database (e.g., `http://pgp.mit.edu`).
- Block-chain based: under study.

[3] Read about some concern with PGP `https://blog.cryptographyengineering.com/2014/08/13/whats-matter-with-pgp/`.

Demonstrate an example of certificate in browser: amazon.com. The format of a certificate follows the standard X.509[4]

[4] `https://tools.ietf.org/html/rfc4158`.

SSL/TLS

- Handshake Protocol
- Record-Layer Protocol

Real-world attacks

CRIME (Compression Ratio Info-leak Made Easy). CPA does not hide the length of the message. Learn secret cookie by Chosen-Plaintext Attack on an encryption that compress the input message. Very clever.

Heartbleed. A mundane implementation flaw in OpenSSL 1.0.1 through 1.0.1f. No bound check: in heartbeat mechanism[5], client sends an "ack" msg that has a payload length; Server bounces the same. It allocates a buffer for its response, and copies "payload" data bytes in its memory. Unfortunately, there's no check to make sure that there are actually "payload" bytes in data, or that this is in bounds. This can leak up to 64KB of the main memory, which could contain all kinds of sensitive data.

[5] to keep connections alive even when no data is being transmitted.

KRACK. Break WPA2 802.11i. [6]

[6] Learn more at `https://www.krackattacks.com/`. Vanhoef's slides at CCS 17 `https://papers.mathyvanhoef.com/blackhat-eu2017-slides.pdf`.