CSA E0 235: Cryptography 12 Apr 2015

Lecture 12

Instructor: Arpita Patra *Submitted by: Niranjan Singh*

Digital Signatures

In the public-key setting, digital signature is used to provide Integrity (or authenticity). It allows a signer who has established a public key pk to sign a message using the associated private key sk in such a way that anyone who knows pk (and knows that this public key was established by S) can verify that the message originated from S and was not modified in transit.

The pair (pk, sk) plays different roles in digital signatures and public key encryption. Digital signatures use sk for signing (the later one uses pk for encryption) and pk for verification (the later one uses sk for decryption). Digital signatures are often mistakenly viewed as the inverse of public-key encryption, with the roles of the sender and receiver interchanged. It was also suggested that digital signatures can be obtained by reversing public-key encryption. But, in most cases it is simply inapplicable, and if applicable it results in insecure signature schemes. Also, unlike public key encryption, digital signature can be constructed just based on the Hash function or a one-way function.

1 Definitions

Definition 1 A triple $\Pi = (Gen, Sign, Vrfy)$ is a **Digital Signature** Scheme, such that

1. The key generation algorithm Gen(1^n) generate a pair of keys (sk, pk) called private key and public key, respectively. The length of these keys is at least n, and this n can be determined from any of these keys.
2. The signing algorithm Sign takes private key sk and a message m from some message space as input and outputs a signature σ, can be written as $\sigma \leftarrow Sign_{sk}(m)$.
3. The deterministic verification algorithm Vrfy takes a public key pk, a message m, and a signature σ as input and outputs a bit b, with b = 1 meaning valid and b = 0 meaning invalid, can be written as $b = Vrfy_{pk}(m, \sigma)$.

$\diamondsuit$

It is required that except with negligible probability over (pk, sk) output by Gen(1^n), it is required that,

$$Vrfy_{pk}(m, Sign_{sk}(m)) = 1$$

Security of digital signatures : A forgery is a message m with a valid signature σ for a pk generated by S, where m was not signed by S. So, for digital signature to be secure the adversary should not be able to output a forgery even if it has the signatures of some of the messages of his choice.

Consider the scheme $\Pi = (Gen, Sign, Vrfy)$ with following experiment for an adversary $\mathcal{A}$ and parameter n,
The **signature experiment** $Sign - forge_{\mathcal{A},\Pi}(n)$:

1. The key pair (sk, pk) is produced by Gen(1^n).
2. The adversary $\mathcal{A}$ has an access to oracle $Sign_{sk}(.)$ with public key pk. $\mathcal{A}$ does $\mathcal{Q}$ queries to this oracle and outputs (m, σ).
3. $\mathcal{A}$ succeeds iff $Vrfy_{pk}(m, \sigma) = 1$ and $m \notin \mathcal{Q}$.

Definition 2 A signature scheme $\Pi = (Gen, Sign, Vrfy)$ is existentially unforgeable under an adaptive chosen-message attack, if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there is a negligible function negl() such that:

$$Pr[Sig - forge_{\mathcal{A},\Pi}(n) = 1] \leq negl(n).$$

$\diamondsuit$

2 Digital Signature vs. MAC

Digital signature and message authentication codes both are used for integrity assurance of messages. In MAC, the sender has to establish a secret key with each receiver and compute corresponding tag for each key, whereas a single signature works for all recipients. So, by using digital signatures rather than MAC simplifies the key distribution and management, specially in case of multiple receivers.

A major advantage of digital signatures over MAC is public verifiability of signatures. If a receiver verifies a signature to be legitimate, then all other parties who received the same message will also verify it as legitimate. But, this can not be done with MAC.

A signature σ by a signer S of message m can be shown to a third party which verifies that σ is a legitimate signature from S. It can then make a copy of σ and show it to some other party and convince that about the authentication of m. Thus, public verifiability also implies transferability.

Digital signatures also provide non-repudiation. It means that once S has signed a message he cannot deny it, as the public key is available to all and anyone can verify. But this is not the case with MAC. To see this, consider that S and R has shared a key k_{SR}. so, when S sends a pair (m, t) to R, there is no way for a judge J to verify that t is valid or not, as he does not know k_{SR}. Even if R shares k_{SR} with J, it still don't know that it is actual key shared between S and R or some fake key generated by R.

MAC's are shorter and roughly $2-3$ orders of magnitude more efficient to generate/verify than digital signatures. So, when public verifiability, transferability, non-repudiation are not needed and sender works with a single receiver, MAC has to be used.

MAC's are more suitable for closed environments like universities, private companies, etc. Whereas, digital signatures are more suitable for open environments like Internet.

Also, digital signatures relies on the fact that public key can be sent to the verifiers in an authenticated manner and requires some way to ensure this.

3 Digital Certificates and Public-key Infrastructure (PKI)

Public key cryptography can be used successfully once the public keys are distributed successfully. Thus, it requires a secure distribution of public keys. The public-key cryptography itself can be used to securely distribute public keys. Once a single public key, belonging to a trusted party, is distributed securely, that key can be used to bootstrap the secure distribution of arbitrarily many other public keys. Thus, the problem of secure key distribution need only be solved once. And digital signatures are used for this purpose.

Digital Certificate is simply a signature binding an entity to some public key and used to prove the ownership of the public key. To see this, consider a party T has generated a key-pair (pk_T, sk_T), and a party S has generated a key-pair (pk_S, sk_S) for a secure digital signature scheme, and T knows that pk_S is the public key of S. Then, T can compute the signature as,

$$cert_{T\to S} = Sign_{sk_T}(pk_S \text{ is public key of S})$$

and give the signature to S. $cert_{T\to S}$ is the certificate of public key of S issued by T.

Now, let S wants to communicate with a party R who already know pk_T. Then, S can send $(pk_S, cert_{T\to S})$ to R, who can verify that the signature $cert_{T\to S}$ is valid on the message "pk_S is public key of S" w.r.t. pk_T. If the verification is successful than R knows that T has signed the indicated message, and if he trusts T then he can accept pk_S as the public key of S. If an active adversary interferes with the transmission of $(pk_S, cert_{T\to S})$, he would not be able to generate a valid certificate linking S to any other public key $pk_S^{'}$ unless T had previously signed some other certificate linking S with $pk_S^{'}$, that again is not much like an attack. All of this in based on assumption that T is honest and pk_T is not compromised.

In order for R to learn pk_T; T to be sure that pk_S is public key of S; and R trusting T, a public-key infrastructure (**PKI**) is required that enables the widespread distribution of public keys. Some of the PKI's are discussed.

3.1 A single certificate authority

The simplest PKI assumes a single certificate authority (CA) who is completely trusted by everybody and who issues certificates for everyone's public key. CA is typically a company whose business it is to certify public keys, a government agency, or perhaps a department within an organization (only be used by people within the organization). A common way for a CA to distribute its public key in practice is to bundle this public key with some other software. For example, this occurs today in many popular web browsers as, a CAs public key is provided together with the browser, and the browser is programmed to automatically verify certificates as they arrive.

3.2 Multiple certificate authority

Outside a single organization it is unlikely for everyone to trust the same CA. Also, the CA is a single point of failure for the entire system. If the CA is corrupt, or can be bribed the legitimacy of issued certificates may be called into question. It is also inconvenient for all parties who want certificates to have to contact this CA.

This issue can be taken care using multiple CA's. A party who wants to obtain a certificate on his public key can choose which CA(s) it wants to issue a certificate, and a party who is presented with a certificate, or even multiple certificates issued by different CAs, can choose which CAs certificates to trusts. These CA's validates each others keys regularly after a particular period.

3.3 Transport Layer Security(TLS) protocol

TLS protocol that is used extensively to secure communication over the web. TLS is the protocol used by the browser any time we connect to a website using https rather than http. TLS allows a client (e.g., a web browser) and a server (e.g., a website) to agree on a set of shared keys and then to use those keys to encrypt and authenticate their subsequent communication. It consists of two parts: a handshake protocol that performs authenticated key exchange to establish the shared keys, and a record-layer protocol that uses those shared keys to encrypt/authenticate the parties communication. Although TLS allows for clients to authenticate to servers, it is primarily used only for authentication of servers to clients because only servers typically have certificates.

3.3.1 Handshake protocol

Let the client C has a set of CA's public keys $\{pk_1, pk_2, \ldots, pk_n\}$, the server S has a pair (pk_S, sk_S) along with a certificate $cert_{i \to S}$ issued by one of the CAs whose public key C knows. To communicate with S, parties takes the following steps,

1. C sends a message to S specifying the information about the versions of protocols supported by client, the ciphersuites supported by the client like which hash functions or block ciphers the client allows, and a uniform nonce N_C.

2. S replies with selecting the latest version of the protocol it supports and an appropriate ciphersuite. It also sends its public key pk_S, its certificate $cert_{i \to S}$, and its own uniform value N_S.

3. C verifies the certificate. If successful, it runs $(c, pmk) \leftarrow Encaps_{pk_S}(1^n)$ to obtain a ciphertext and a pre-master key pmk. This pmk is then used to derive a master key mk using a key-derivation function applied to pmk, N_C, and N_S. Then, C applies a pseudorandom generator to mk to derive four keys k_C, $k_C^{'}$, k_S, $k_S^{'}$. After that, C computes $\tau_C \leftarrow Mac_{mk}(transcript)$, where transcript denotes all messages exchanged between C and S so far. Then, this τ_C is send to S.

4. S computes $pmk = Decaps_{sk_S}(c)$, and derives mk and k_C, $k_C^{'}$, k_S, $k_S^{'}$ from it just like the client did. If $Vrfy(transcript, \tau_C) \neq 1$, then S aborts. Otherwise, it sets $\tau_S \leftarrow Mac_{mk}(transcript^{'})$, where $transcript^{'}$ denotes all messages exchanged between C and S so far, including last message sent by C. Then, τ_S is send to C.

5. If $Vrfy_{mk}(transcript^{'}, \tau_S) \neq 1$, client aborts.

At the end of successful execution C and S shares four symmetric keys k_C, $k_C^{'}$, k_S, $k_S^{'}$. The intuition behind security of the handshake protocol is that since C verifies the certificate, it knows that only the legitimate server S can learn pmk and hence mk. Thus, if the protocol terminates successfully, C knows that it shares keys with the legitimate S.

3.3.2 Record-layer Protocol

After successfully getting the keys k_C, $k_C^{'}$, k_S, $k_S^{'}$, these keys are used for all further communication. C uses k_C and $k_C^{'}$ for encryption and authentication, and S uses k_S and $k_S^{'}$ for encryption and authentication. Also, sequence numbers are used to prevent replay attacks.