CS171: Cryptography

Lecture 17

Sanjam Garg

Digital Signatures

Public-Key Analogue of MAC

- Software Update
  - Comes pre-loaded with public-key
  - No need to share separate keys with everyone

- Non-repudiation
  - ability to ensure that a party to a contract or a communication cannot deny the authenticity of their signature on a document
  - Judge can enforce (couldn’t do so with MACs)

Syntax

- $Gen(1^n)$: Outputs public key and secret key pair (pk, sk).

- $Sign_{sk}(m)$: Outputs a signature σ on the message m.

- $Vrfy_{pk}(m, \sigma)$: Outputs $0/1$.

Correctness: For all n, except for negligible choices of (pk, sk), it holds that for all m, $Vrfy_{pk}\big(m, Sign_{sk}(m)\big) = 1$.

Security?

- Attacker’s power
  - Public-key is provided to the attacker
  - ``Adaptive chosen-message attack’’
  - The attacker can request signatures on messages of its choice

- Security Goal
  - ``Existentially Unforgeable’’
  - Attacker can forge any message not already signed by the signer

Unforgeability/Security of Digital Signature

$\mathrm{Forge}_{A,\Pi}(1^n)$

1. Sample *(pk,sk)* $\leftarrow$ Gen(1^n).
2. Let (m^*, σ^*) be the output of $A^{Sign_{sk}(\cdot)}(pk)$. Let M be the list of queries A makes.
3. Output 1 if $Vrfy_{pk}(m^*, \sigma^*) = 1 \wedge m^* \notin M$ and 0 otherwise.

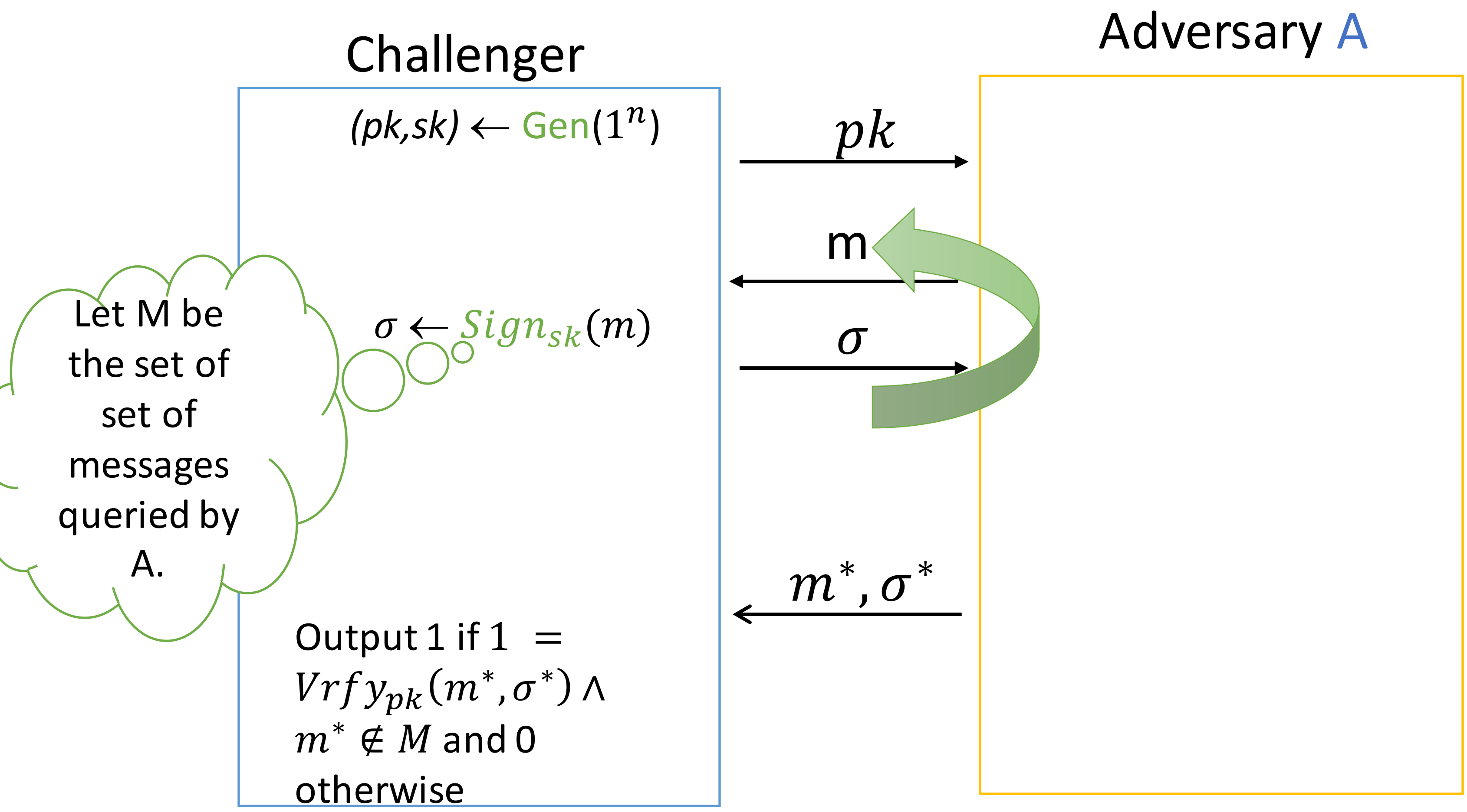
$\Pi = (Gen, Sign, Vrfy)$ is existentially unforgeable under adaptive chosen attack if

$\forall$ PPT A it holds that:

$\Pr[\mathrm{Forge}_{A,\Pi} = 1] \leq$ negl(n)

Unforgeability (Pictorially)

$$\mathrm{MacForge}_{\mathrm{A},\Pi}(1^n)$$



Security

- What about replay attacks?
  - Same as the issue for MACs
  - Leave it to the application designer

Hash and Sign (A method to sign messages of arbitrary length)

- Signature scheme $\Pi = (Gen, Sign, Vrfy)$ that can sign short messages (length n)
- Hash function $H: \{0,1\}^* \rightarrow \{0,1\}^n$

- New signature scheme $\Pi' = (Gen, Sign', Vrfy')$
  - $Sign'_{sk}(m) = Sign_{sk}\big(H(m)\big)$
  - $Vrfy'_{pk}(m, \sigma) = Vrfy_{pk}(H(m), \sigma)$

Proof of Security

- Given an attacker A breaking Π', we will construct attacker B breaking Π or an attacker C breaking H.
  - Let M be the list of queries A makes.
  - Let M' be the hashes of strings in M.

- A outputs a forgery (m^*, σ^*) such that $m^* \notin M$. Two cases arise:
  - $H(m^*) \in M'$: An attack against the CRHF property of H.
  - $H(m^*) \notin M'$: An attack against Π.

Analogue of Hash-and-Mac paradigm.

Detour: Schnorr Identification Scheme

Schnorr Signature Scheme

Schnorr Identification Scheme

- Allows a prover to convince a verifier of its identity

- Consists of four algorithms

$$Gen, P_1, P_2, I$$

Identification Scheme

$(pk, sk) \leftarrow Gen(1^n)$

Prover (sk) | **Verifier (pk)**

$(I, st) \leftarrow P_1(sk)$

$\xrightarrow{\quad I \quad}$

$\xleftarrow{\quad r \quad}$ $\quad r \leftarrow \Omega_{pk}$

$s \coloneqq P_2(sk, st, r)$

$\xrightarrow{\quad s \quad}$

$I \overset{?}{=} V(pk, r, s)$

Correctness: Honest prover can always (or almost always) convince the verifier.

Security of Identification Schemes

- Adversary (that doesn't know sk) shouldn't be able fool the verifier into accepting.
- Even if the attacker is able to passively eavesdrop on multiple honest executions of the protocol.

Security of Identification Schemes

$\mathrm{Iden}_{\mathrm{A},\Pi}(1^n)$

1. Sample *(pk,sk)* $\leftarrow$ Gen(1^n).
2. Let I^* be the output of $A^{Tran_{sk}(\cdot)}(pk)$.
3. Sample $r^* \leftarrow \Omega_{pk}$.
4. Let s^* be the output of $A^{Tran_{sk}(\cdot)}(r)$.
5. Output 1 if $V(pk, r^*, s^*) = I^*$ and 0 otherwise.

$\Pi = (Gen, P_1, P_2, V)$ is secure under a passive attack if

$\forall$ PPT A it holds that:

$\Pr[\mathrm{Iden}_{\mathrm{A},\Pi} = 1] \leq$ negl(n)

$Tran_{sk}(\cdot)$ outputs a transcript of execution between a prover and a verifier.

Construction of Schnorr Identification Scheme

- $Gen(1^n)$: Output $pk = (G, g, q, h)$ where $h = g^x$ and $sk = x$.
- $P_1(sk)$: Sample $k \leftarrow Z_q$. Output $I = g^k$ and $st = (I, k)$.
- Ω_{pk}: Output $r \leftarrow Z_q$.
- $P_2(sk, st, r)$: Output $s = r\,x + k \bmod q$
- $V(pk, r, s)$: Output $\frac{g^s}{h^r}$

$Tran_{sk}$ is useless

1. Sample $r, s \leftarrow Z_q$.
2. Set $\mathrm{I} := \frac{g^s}{h^r}$.
3. Output (I, r, s)

Note that verification passes (sk was not used)

Suffices to prove the security without access to $Tran_{sk}$

Proof without $Tran_{sk}$

- Once the attacker outputs I we could run it on two different values r_1 and r_2.
- Consider the case where adversary produces accepting s_1 and s_2 for both challenges.
- $\frac{g^{s_1}}{h^{r_1}} = I = \frac{g^{s_2}}{h^{r_2}}$
- $s_1 - r_1 x = s_2 - r_2 x$
- Solve for x!

From Identification Scheme to Signatures (without proof)

- $Gen(1^n)$: Generate $(pk, sk) \leftarrow Gen_{id}(1^n)$ and setup a hash function H: $\{0,1\}^* \rightarrow \Omega_{pk}$

- $Sign_{sk}(m)$: Generate $(I, st) \leftarrow P_1(sk), r := H(I, m), s := P_2(sk, st, r)$ and $\sigma = (I, s)$.

- Vrf$y_{pk}(m, \sigma)$: Output 1 if $V(pk, H(I, m), s) = I$ and 0 otherwise

Though we will not prove. It can be proved if H behaves like a random function

PKI Infrastructure

- How does get Alice get Bob’s public key?

- Use digital signatures for secure public key distribution
- Let’s say that everyone knowns the public key of a trusted party (aka ``the CA’’)
    - Certification Authority
    - Public Key: PK_{CA}
    - Secret Key: SK_{CA}

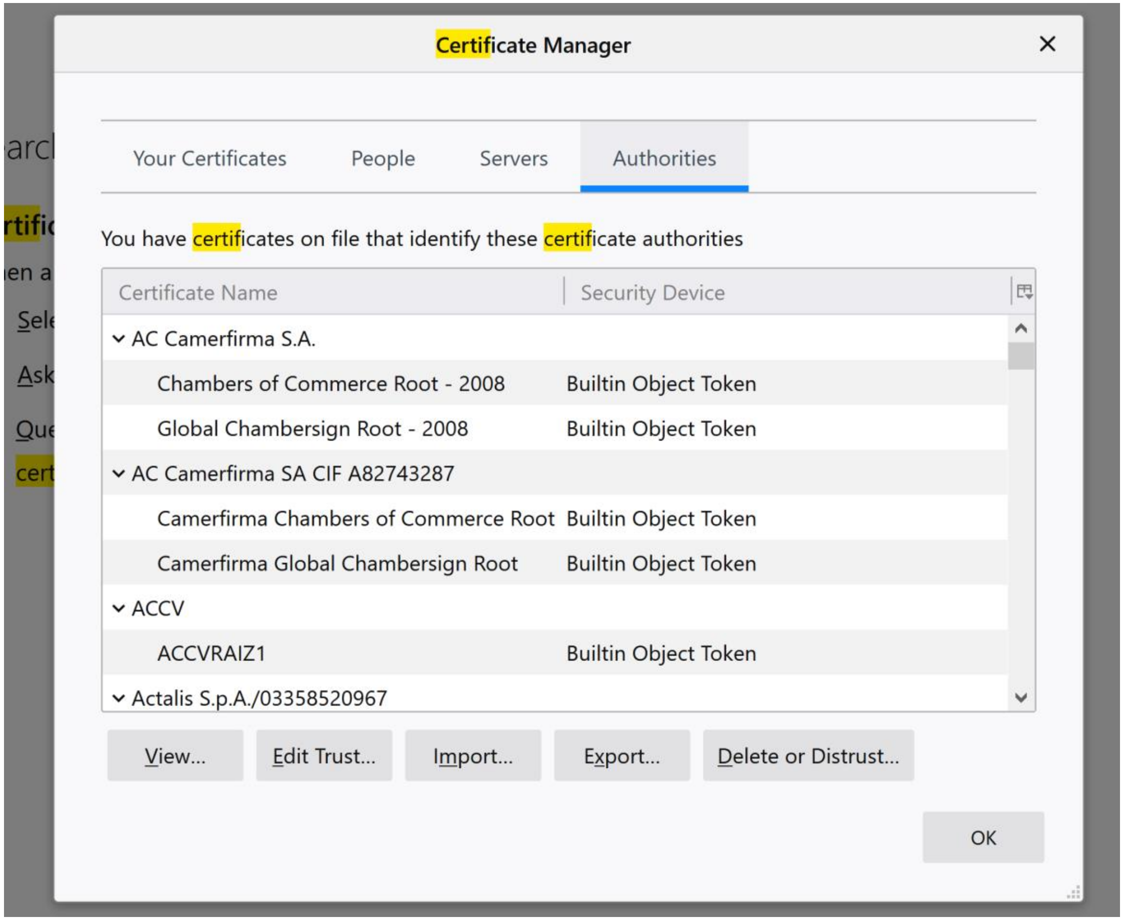
What does Bob do?

- Bob goes to the CA with his public key PK_{Bob}
- $cert_{CA \to Bob} = Sign_{SK_{CA}}(Bob || PK_{Bob})$

- Given this certificate and CA's public key, Alice can verify that PK_{Bob} is indeed Bob's public key

- How does Alice get CA's key in the first place?

How does Alice get CA's key in the first place?



Digital Signatures can be realized based on any one-way function.

We won’t see this construction.

Thank You!