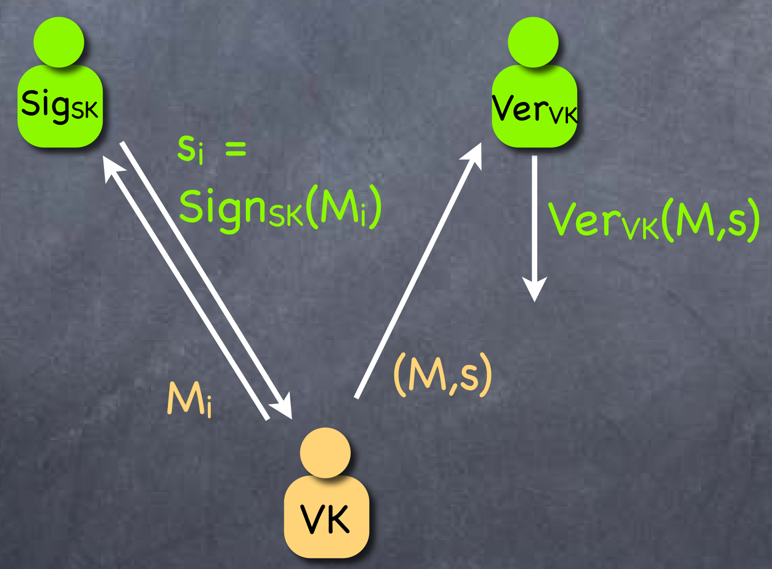
Digital Signatures (ctd.)

Lecture 17

RECALL

Digital Signatures

- Syntax: KeyGen, $Sign_{SK}$ and $Verify_{VK}$.
  Security: Same experiment as MAC's, but adversary given VK



Advantage = Pr[$Ver_{VK}(M,s)=1$ and $(M,s) \notin \{(M_i,s_i)\}$]

Weaker variant: Advantage = Pr[$Ver_{VK}(M,s)=1$ and $M \notin \{M_i\}$]

Digital Signatures

- Online verification of real life identity is difficult
- But the verification key for a digital signature can serve as your digital identity
  - OK to own multiple digital identities
  - Compromised if you lose your signing key
- Central to identity on the internet (with the help of certificate authorities), crypto currencies, etc.

"On the Internet, nobody knows you're a dog."

Signatures from OWF

- Lamport's scheme based on OWF
  - One-time and has a fixed-length message
- One-time, fixed-length message signatures (Lamport)

  Domain-Extension → arbitrary length messages (using UOWHF)

  "Certificate Tree" → many-time signatures (using PRF)
- So full-fledged digital signatures can be entirely based on OWF
- Last time: Hash-and-Sign domain extension for signatures
  - Domain extension can be done using CRHF (more efficient) or UOWHF (more secure)
- Today: "Certificate tree"

One-Time → Many-Times

- Certificate chain: $VK_1 \rightarrow (VK_2, \sigma_2) \rightarrow \ldots \rightarrow (VK_t, \sigma_t) \rightarrow (m,\sigma)$ where σ_i is a signature on VK_i that verifies w.r.t. VK_{i-1}
  - Suppose a "trustworthy" signer only signs the verification key of another "trustworthy" signer. Then, if VK_1 is known to be issued by a trustworthy signer, and all links verified, then the message is signed by a trustworthy signer.
- Certificate tree for one-time → many-times signatures
  - Idea: Each message is signed using a unique VK for that message
    - Verifier can't hold all VKs: A binary tree of VKs, with each leaf designated for a message. Parent VK signs its pair of children VKs (one-time, fixed-length sign). Verifier remembers only root VK. Signer provides a certificate chain to the leaf VK used.
    - Signer can't remember all SKs: Uses a PRF to define the tree (i.e., SK for each node), and remembers only the PRF seed

RECALL

Domain Extension of Signatures using Hash

- Domain extension using a CRHF (not weak CRHF, unlike for MAC)
  - $Sign^*_{SK,h}(M) = Sign_{SK}(h(M))$ where $h \leftarrow \mathcal{H}$ in both SK*,VK*
  - Security: Forgery gives either a hash collision or a forgery for the original (finite domain) signature
  - Formal reduction to a pair of adversaries. Hash adversary sends h it receives as part of VK
- Can use UOWHF, with fresh h every time (included in signature)
  - $Sign^*_{SK}(M) = (h, Sign_{SK}(h,h(M)))$ where $h \leftarrow \mathcal{H}$ picked by signer
  - Security: To use a signature s_i in forgery, need M such that $h(M)=h(M_i)$. But h is picked by signing algorithm after M_i is submitted. Breaks UOWHF security by finding such a collision.
    - In reduction, hash adversary guesses an i where collision occurs and sends h it received as part of signature

RECALL

More Efficient Signatures: Hash and Invert

- Using a trapdoor OWP and a "hash": **$\mathrm{Sign}(M) = f^{-1}(\ \mathrm{Hash}(M)\ )$**
  - Where (SK,VK) = (f^{-1},f), a Trapdoor OWP pair
  - Secure in the random oracle model
  - Hash can handle variable length inputs
  - "Standard schemes" like RSA-PSS are based on this

Schnorr Signature

- Public parameters: (G,g) where G is a prime-order group and g a generator, for which DLA holds, and a random oracle H
  - Or (G,g) can be picked as part of key generation
- Signing Key: $y \in Z_q$ where G is of order q. Verification Key: $Y = g^y$
- $Sign_y(M) = (x,s)$ where $x = H(M||g^r)$ and $s = r-xy$, for a random r
- $Verify_Y(M,(x,s))$: Compute $R = g^s \cdot Y^x$ and check $x = H(M||R)$
- Secure in the Random Oracle model under the Discrete Log Assumption for a group
  - Alternately, under a heuristic model for the group (called the Generic Group Model), but under standard-model assumptions on the hash function

VK as ID: An Example
Identity-Based Encryption

- In PKE, KeyGen produces a random (PK,SK) pair
- Can I have a "fancy public-key" (e.g., my name)?
  - No! Not secure if one can pick any PK and find an SK for it!
- But suppose a trusted authority for key generation
  - Then: Can it generate a valid (PK,SK) pair for any PK?
  - Identity-Based Encryption: a key-server (with a master secret-key) that can generate such pairs
    - Encryption will use the master public-key, and the receiver's "identity" (i.e., fancy public-key)
    - In PKE, sender has to retrieve PK for every party it wants to talk to (from a trusted public directory)
    - In IBE, receiver has to obtain its SK from the authority

VK as ID: An Example

Identity-Based Encryption

- Security requirement for IBE (will skip formal statement):
  - Environment/adversary decides the ID of the honest parties
  - Adversary can adaptively request SK for any number of IDs (which are not used for honest parties)
  - "Semantic security" for encryption with the ID of honest parties (i.e., with no access to decryption: CPA security)
- IBE (even CPA-secure) can easily give CCA-secure PKE!
  - IBE: Can't malleate ciphertext for one ID into one for another
  - $\text{PKEnc}_{\text{MPK}}(m) = (id, C=\text{IBEnc}_{\text{MPK}}(id; m), \text{sign}_{id}(C))$
  - Security: can't create a different encryption with same id (signature's security); can't malleate using a different id (IBE's security)

Digital Signature with its public-key used as the ID in IBE