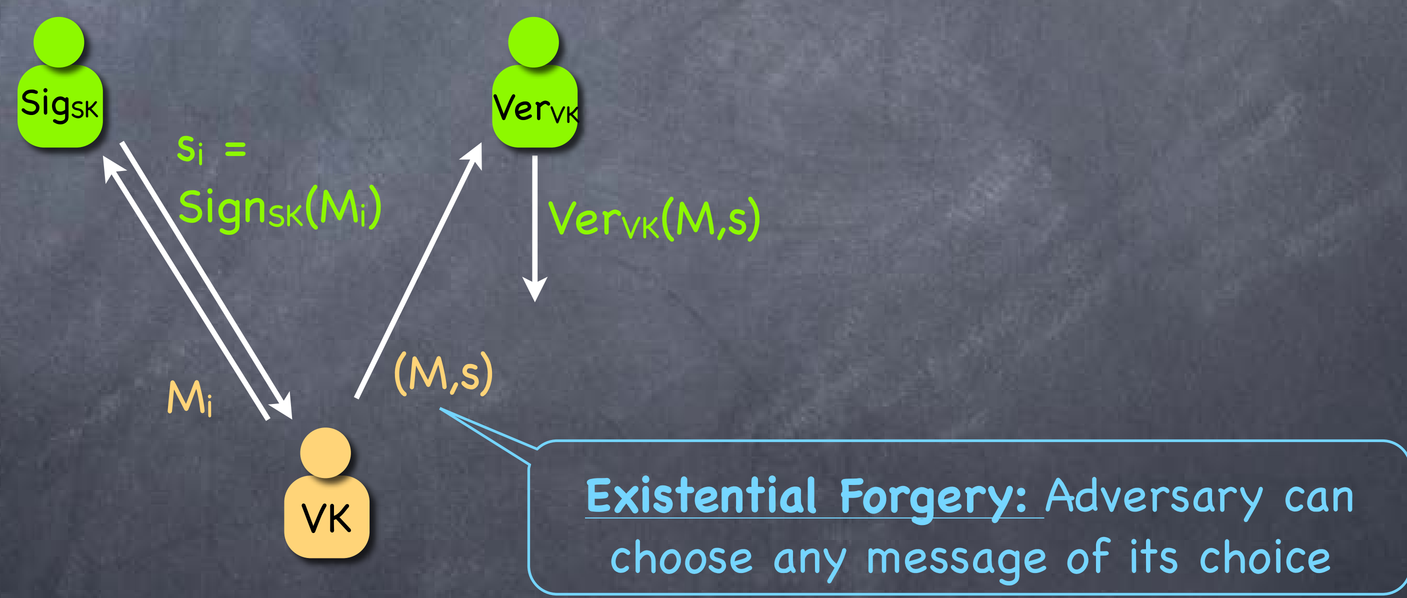
Digital Signatures

Lecture 12

Digital Signatures

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



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

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

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."

One-time Digital Signatures

Lamport's One-Time Signature

- Recall One-time MAC to sign a single n bit message
  - Shared secret key: 2n random strings (each k-bit long) $(r^i_0, r^i_1)_{i=1..n}$
  - Signature for $m_1 \ldots m_n$ be $(r^i_{m_i})_{i=1..n}$
- One-Time Digital Signature: Same signing key and signature, but VK= $(f(r^i_0), f(r^i_1))_{i=1..n}$ where f is a OWF
  - Verification applies f to signature elements and compares with VK
  - Security [Exercise]

$f(r^1_0)$	$f(r^2_0)$	$f(r^3_0)$
$f(r^1_1)$	$f(r^2_1)$	$f(r^3_1)$

r^1_0	r^2_0	r^3_0
r^1_1	r^2_1	r^3_1

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, in principle, full-fledged digital signatures can be entirely based on OWF
- Coming up:
  - Hash-and-Sign domain extension for signatures
    - Domain extension using CRHF (UOWHF suffices, but less efficient)
  - "Certificate tree"

Domain Extension of Signatures using Hash

- Domain extension using a CRHF (not weak CRHF, unlike for MAC)
  - $\text{Sign}^*_{SK,h}(M) = \text{Sign}_{SK}(h(M))$ where $h \leftarrow \mathcal{H}$ included in both SK*,VK*
  - Security: Forgery gives either a hash collision or a forgery for the original (finite domain) signature
  - Formal reduction: Given adversary A for Sign*, define
    - Event_1: A outputs (M,σ) s.t. $h(M)=h(M_i)$, $M_i \neq M$, where A had asked for signature on M_i.
      Event_2: A's forgery not on such an M.
    - Advantage $\leq \Pr[\text{Event}_1 \text{ or } \text{Event}_2] \leq \Pr[\text{Event}_1] + \Pr[\text{Event}_2]$
    - CRHF adversary: given h, sample (SK,VK), let VK*=(VK,h), and run A; answer signing queries of A using (SK,h). If A outputs (M,σ) s.t. $\exists i\ h(M)=h(M_i)$, $M_i \neq M$, then output (M,M_i). Advantage = $\Pr[\text{Event}_1]$
    - Signature adversary: given VK, pick h, let VK*=(VK,h), and run A; answer signing queries of A using h and Sign oracle. If A outputs forgery (M,σ), output $(h(M),\sigma)$. Advantage = $\Pr[\text{Event}_2]$

One-Time $\rightarrow$ 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}, and σ is a signature on m w.r.t. VK_t
  - 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 $\rightarrow$ 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

Signatures from OWF
Summary

- One-time, fixed-length message signatures (Lamport)
  - Domain-Extension → arbitrary length messages (using UOWHF)
  - "Certificate Tree" → many-time signatures, fixed length (using PRF)
  - Domain-Extension → arbitrary length messages
- UOWHF and PRF can be based on OWF, and so, in principle, full-fledged digital signatures can be entirely based on OWF
- Not very efficient: Say hashes are O(k) bits long. Then, a signature contains O(k) VKs of Lamport signature, each of which, to allow signing O(k) bit messages, is $O(k^2)$ bits long. Overall $O(k^3)$ bits long.
- Coming up: More efficient schemes

Hash and Invert

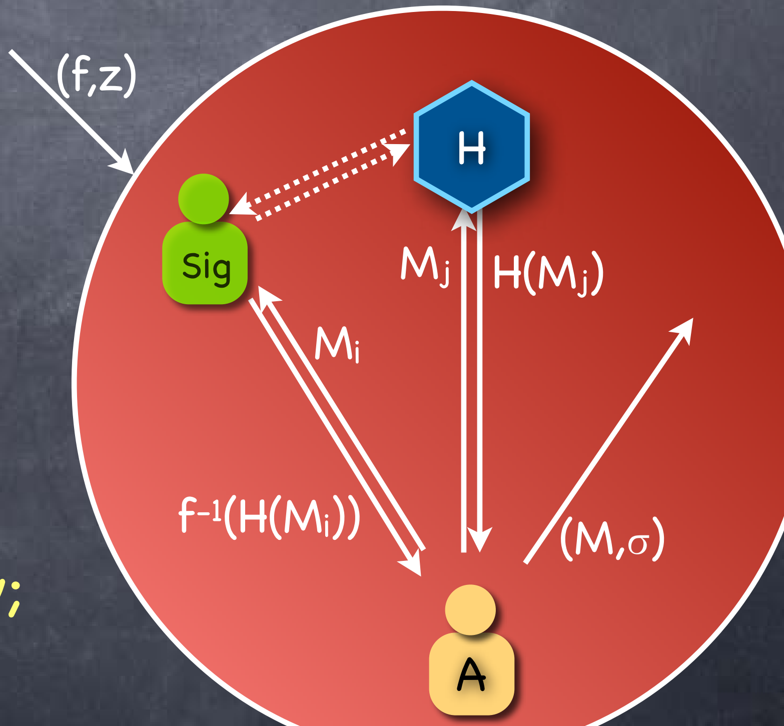
- Diffie-Hellman suggestion (heuristic): Sign(M) = f^{-1}(M) where (SK,VK) = (f^{-1},f), a Trapdoor OWP pair. Verify(M,σ) = 1 iff f(σ)=M.
  - Attack: pick σ, let M=f(σ) (Existential forgery)
- Fix, using a "hash": **Sign(M) = f^{-1}(Hash(M))**
  - Secure in the **random oracle** model
  - Hash can handle variable length inputs
  - RSA-PSS in RSA Standard PKCS#1 is based on this

Proving Security in the RO Model

- To prove: If Trapdoor OWP secure, then Sign(M) = f^{-1}(Hash(M)) is a secure digital signature, when Hash is modelled as a random oracle
  - Hope: Since adversary can't invert Hash, needs to compute f^{-1}
  - Problem: Signing oracle gives adversary access to the f^{-1} oracle. But then, trapdoor OWP gives no guarantees!
  - But adversary only sees $(x,f^{-1}(x))$ where x = Hash(M) is random. This can be arranged by picking $f^{-1}(x)$ first and fixing Hash(M) afterwards!
- Modeling as an RO: RO randomly initialized to a random function H from $\{0,1\}^*$ to $\{0,1\}^k$
  - Signer and verifier (and forger) get oracle access to H(.)
  - All probabilities also over the initialization of the RO

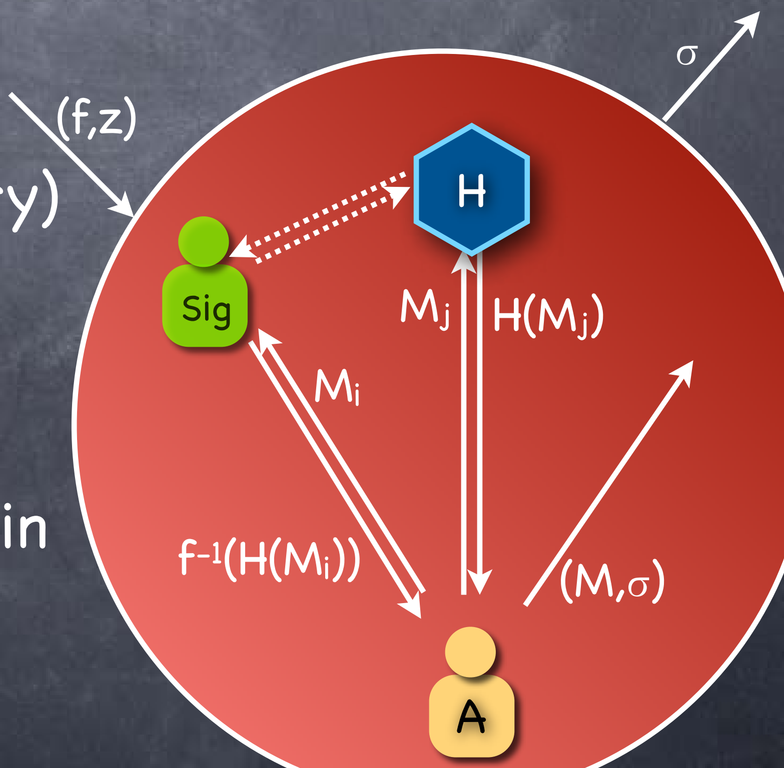
Proving Security in ROM

- Reduction: If A forges signature (where Sign(M) = $f^{-1}(H(M))$ with (f,f^{-1}) from Trapdoor OWP and H an RO), then A* that can break Trapdoor OWP (i.e., given just f, and a random challenge z, can find $f^{-1}(z)$ w.n.n.p). A*(f,z) runs A internally.
  - A expects f, access to the RO and a signing oracle f^{-1}(Hash(.)) and outputs (M,σ) as forgery
  - A* can implement RO: a random response to each new query!
  - A* gets f, but doesn't have f^{-1} to sign
    - But x = H(M) is a random value that A* can pick!
    - A* picks H(M) as x=f(y) for random y; then Sign(M) = $f^{-1}(x)$ = y



Proving Security in ROM

- A* s.t. if A forges signature, then A* can break Trapdoor OWP
  - A* implements H and Sign: For each new M queried to H (including by Sign), A* sets H(M)=f(y) for random y; Sign(M) = y
  - But A* should force A to invert z
    - Easy if forgery on a fresh M (set H(M)=z at the end). But needn't be so.
    - For a random (new) query M (say t^{th}) A* sets H(M)=z
      - Here queries include the "last query" to H, i.e., the one for verifying the forgery (which may or may not be a new query)
  - Given a bound q on the number of queries that A makes to Sign or H, with probability ≥ 1/q, A* would have set H(M)=z, where M is the message in the forgery
    - In that case forgery ⇒ $\sigma = f^{-1}(z)$



Summary

- Digital signatures can be based on OWF + UWOHF + PRF
  - In turn based on OWF (or more efficiently on OWP)
- More efficiently, can be based on number-theoretic/algebraic assumptions (e.g., Cramer-Shoup signatures based on Strong RSA and CRHF)
- In practice, based on number-theoretic/algebraic assumptions in the random oracle model
  - RSA-PSS, of the form f^{-1}(Hash(M)), where f a Trapdoor OWP
  - DSA and variants, based on Schnorr signature (next time)
- Next up: Zero-Knowledge proofs