Disclaimer. Draft note. No guarantee on completeness nor soundness. Read with caution, and shoot me an email at fsong@pdx.edu for corrections/comments (they are always welcome!)

Last time. PubKE, CCA
Today. More CCA, digital signature, full-domain-hash

CCA motivation recap, and CCA constructions. See Lecture 12 note.

1 Digital signatures

Diffie-Hellman also suggested running a trapdoor permutation reversely, i.e. $\sigma := I_{sk}(m)$, as a means of message and identity authentication. Although we know nowadays that this construction is not secure, the primitive, formalized as a *digital signature* scheme, is nonetheless extremely influential in a digital era.

Definition 1 (KL-12.1)**.** A digital signature scheme consists PPT algorithms (G, S, V) such that:

1. G: $(pk, sk) \leftarrow G(1^n)$.
2. S: on input sk and message m, outputs $\sigma \leftarrow S_{sk}(m)$.
3. V: on input pk and message-signature pair (m, σ), output $V_{pk}(m, \sigma) = \text{acc/rej}$.

Software update example. How to use a signature scheme?

- MS generates (pk, sk). Publish pk.
- Software patch m is signed under sk: $\sigma \leftarrow S_{sk}(m)$.
- User downloads (m, σ), and verifies σ with pk, $V_{pk}(m, \sigma) = b$. If $b = \text{acc}$, execute m and complete update.

Comparison to MAC. DS and MAC both protect date integrity. One drawback of DS, as in PubKE, is that it is usually less efficient than MAC. Otherwise DS are advantageous in many aspects.

- No need to share a private-key with each user. Everyone just generates its (pk_i, sk_i) pair and makes pk_i public.
- Publicly verifiable.
- Transferable.
- Non-repudiation. Once a sender signs a message, s/he cannot later deny having done so.

A common misconception. Signature is often considered the "inverse" of public-key encryption. Use decryption as signing, and encrypting as verification. This is *totally unsound*!

1.1 Defining secure digital signature

Intuitively, we would need that no adversary without knowing the secret key can produce valid signatures. The formal definition is similar to that of MAC, considering chosen-message-attacks where an adversary may ask to see signatures on messages of its choice.

FS NOTE: Draw sig-forge game

1. CH generates $(pk, sk) \leftarrow G(1^n)$.
2. $\mathscr{A}$ is given pk and access to signing oracle $S_{sk}(\cdot)$. Let $\mathscr{L} := \{m_i\}$ be the set of messages that $\mathscr{A}$ has queried. At the end $\mathscr{A}$ outputs (m^*, σ^*).
3. We say $\mathscr{A}$ succeeds if 1) $V_{pk}(m^*, \sigma^*) = \text{acc}$; and 2) $m^* \notin \mathscr{L}$. Define the output of the game $\mathsf{Sig\text{-}forge}_{\mathscr{A},\Pi}(n) = 1$ iff. $\mathscr{A}$ succeeds.

Figure 1: Signature forgery game $\mathsf{Sig\text{-}forge}_{\mathscr{A},\Pi}(n)$

Definition 2 (KL-Def. 12.2)**.** A signature scheme $\Pi = (G, S, V)$ is existentially unforgeable under an adaptive chosen-message attack (eu-cma-secure), if for all PPT $\mathscr{A}$,

$$\Pr[\mathsf{Sig\text{-}forge}_{\mathscr{A},\Pi}(n) = 1] \leq \text{negl}(n).$$

Note that implicitly, $\mathscr{A}$ also has access to the verification procedure $V_{pk}(\cdot)$. What if we give verification oracle $V_k(\cdot)$ to $\mathscr{A}$ in MAC? Does this make any difference?

2 Constructing digital signatures based on TDPs

Today we talk about signatures based on trapdoor one-way permutations.

2.1 (Insecure) direct TDP-DS

Let $\Sigma = (G, F, I)$ be a TDP. If we execute Diffie-Hellman's idea directly, we would get $\Pi = (G, S, V)$:

- $G = G$: $(pk, sk) \leftarrow G(1^n)$.
- S: on input sk and m, $\sigma = I_{sk}(m)$.
- V: on input pk and (m, σ), output "acc" iff. $F_{pk}(\sigma) = m$.

Unfortunately, this is NOT always secure. In fact, it is often insecure. For instance if we use the RSA-TDP, then we have the "textbook" RSA signature:

- G: run $(N, e, d) \leftarrow \mathsf{GRSA}(1^n)$, and output $pk = (N, e)$, $sk = (N, d)$.
- S: $S_{sk}(m) = [m^d \bmod N]$.
- V: $V_{pk}(m, \sigma)$ outputs "acc" iff. $[\sigma^e \bmod] = m$.

This is completely insecure. Pick an arbitrary σ and compute $m = [\sigma^e \bmod N]$. This is possible since e is public. (m, σ) is a valid forgery! One may say the message obtained this way may not make any sense. Nonetheless, under chosen-message-attacks, an adversary can forge on any message! Read [KL: page 446]. **FS NOTE**: How to get a signature of $m_1 \cdot m_2$ from $\sigma_1 = S_{sk}(m_1)$ and $\sigma_2 = S_{sk}(m_2)$?

2.2 TDP with Full domain Hash

An efficient construction of a secure signature based on a TDP is called Full-Domain-Hash (FDH) proposed in [BR93], which employs the random oracle. Let $\Sigma = (G, F, I)$ be a trapdoor permutation on n bits, $\mathcal{O} : \{0,1\}^* \to \{0,1\}^n$ be a random oracle.

Construct $\Pi = (KG, S, V)$ **FS NOTE**: Draw FDH game

- $KG = G$: $(pk, sk) \leftarrow G(1^n)$.
- S: compute $y := \mathcal{O}(m)$, and output $\sigma := I_{sk}(y)$.
- V: compute $y := F_{pk}(\sigma)$ and accept iff. $y = \mathcal{O}(m)$.

Figure 2: Full-Domain-Hash

Theorem 3 (KL-Thm. 12.7)**.** Π *is a secure signature, assuming* Σ *is a trapdoor one-way permutation and* $\mathcal{O}$ *is modeled as a RO.*

Proof idea. Intuitively, because $\mathcal{O}$ is a random oracle, to forge on a new message that an adversary would need to invert a random element y. This is nonetheless difficult since F is one-way.

FS NOTE: draw reduction diagram

We use $\mathcal{A}$ to break one-wayness of TDP. Our reduction algorithm B receives pk, y with a random y of which we want to find the inverse. We would run $\mathcal{A}$, but we will need to simulate a random oracle and a signing oracle to answer two types of queries from $\mathcal{A}$: RO queries and signing queries. Assume the (extremely) simple case that $\mathcal{A}$ does not query the signing oracle at all (i.e., no-message-attack). We may just answer RO queries on-the-fly. But this does not seem helpful to solve the inverting problem.

One crucial observation is that to make a forgery (m^*, σ^*) such that $\mathcal{O}(m^*) = F_{pk}(\sigma^*)$, $\mathcal{A}$ must have queried $\mathcal{O}$ on m^*, because otherwise $\mathcal{O}(m^*)$ would be uniformly random and unlikely to equal $F_{pk}(m^*)$. Then we could **embed** y as the response $\mathcal{O}(m^*)$. This trick, a baby version of "*programming*" a RO, is totally legitimate because y is uniformly random, i.e. same view from $\mathcal{A}$'s perspective. But now we've found the preimage of y, which is σ^*! Of course we do not know in advance which RO query m_i will $\mathcal{A}$ choose as m^* to forge on, so we choose one at random which will be correct with probability $1/q_{\mathcal{O}}$.

For general $\mathcal{A}$, we need a way to answer signing queries. But it's not clear how this is possible since we do not know a signing key sk. Note again that to sign m, we need to find σ such that $F_{pk}(\sigma) = \mathcal{O}(m)$. The main trick that RO allows us is to just pick a random σ and program $\mathcal{O}(m) = F_{pk}(\sigma)$. This maintains the uniform distribution $\mathcal{O}$'s output, as far as $\mathcal{A}$ is concerned. □

RSA-OAEP. Instantiating TDP-FDH with RSA-TDP, we get RSA-FDH which is eu-cma-secure. A variant of it has been standardized in RSA PKCS #1 v2.1.

References

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the First ACM conference on Computer and Communications Security*, pages 62–73. ACM, 1993.