Alin Tomescu, CSE408
Tuesday, March 22nd, Lecture #15

Public-key signatures

Motivation: Since MACs use the secret key that everyone shares, you can't assign identity by "signing" something with a MAC. You can only protect integrity using a MAC. In public-key cryptography, it is sometimes useful to be able to sign a message and have the receiver verify that the message was indeed signed by you and it was not tampered with. MACs are for symmetric encryption schemes, public key signatures are for public-key encryption schemes.

Alice: has her secret key sk

- $s = Sig_{sk}(m)$

Bob: has a public key pk

- $Vrfy_{pk}(m, s) = \begin{cases} valid \\ invalid \end{cases}$

Definition: A signature scheme is (t, q, ε)-secure against existential forgery if $\forall A$ running in time $\leq t$, making $\leq q$ queries, where m was not a query to $Sig_{sk}(\cdot)$, we have:

$$Adv\ A = \Pr\left[A^{Sig_{sk}(\cdot)}(pk) = (m, t) \text{ st. } Vrfy_{pk}(m, t) = OK\right] \leq \varepsilon$$

What might an attacker try to do in an attempt to forge a message signature?

- they might look at the messages being signed
- Eve can obtain the signatures on messages of her choice
- Eve has access to an oracle that signs any message
- Eve wins if she's able to sign a message that was never queried to the oracle

Signature schemes

Textbook RSA

Consider the following scheme based on RSA:

$$Sig_{(n,d)}(m) = m^d \bmod n$$

$$Vrfy_{(n,e)}(m, s) = \begin{cases} \text{valid}, & if\ s^e \bmod n = m \\ \text{invalid}, & if\ s^e \bmod n \neq m \end{cases}$$

It turns out this scheme is not secure. It turns out that you can pick any signature s and create a message m that has that signature, which is a bad thing.

Attack: Pick any $s \in Z_n^*$, set $m = s^e$, output the forged message-signature pair (m, s)

- m might not be the message the attacker wants to forge

TOWP scheme

Let f be a TOWP. Consider the following scheme based on f:

$$Sig_{sk}(m) = f_{sk}^{-1}(m)$$

$$Vrfy_{pk}(m,s) = \begin{cases} \text{valid}, & if\ f_{pk}(s) = m \\ \text{invalid}, & if\ f_{pk}(s) \neq m \end{cases}$$

This scheme is also insecure. As before, you can pick any signature s and create a message m that has that signature, which is a bad thing.

Attack: Pick any s, set $m = f_{pk}(s)$, output (m, s)

Hashing

To fix this "signature" problem, people introduced **hashing**.

Essentially the problem with the previous schemes was that given a signature s, a message m with that signature could've been easily forged. Why? Because in the $Vrfy$ function, if you fix s, the message m can be easily computed by just computing some TOWP in the forward direction, which is easy.

Popular hashing algorithms:

- MD5 – broken
- SHA1 – broken
- SHA256 – so far so god

Definition: A hash function takes an arbitrary length string and maps it to a fixed length string

$$H: \{0,1\}^* \rightarrow \{0,1\}^l$$

Hashed RSA

Consider this revised RSA-based signing scheme:

$$Sig_{(n,d)}(m) = H(m)^d \bmod n$$

$$Vrfy_{(n,e)}(m,s) = \begin{cases} \text{valid}, & if\ s^e \bmod n = H(m)? \\ \text{invalid}, & if\ s^e \bmod n \neq H(m)? \end{cases}$$

Attack:

- pick any $s \in Z_n^*$
- set $h = s^e$
- find a message m s.t. $H(m) = h$
    - this step now becomes computationally unfeasible or hard
- output (m, s)

The random oracle model

Random oracle model will define how we treat hash functions in our security model.

Note: In a hash function there's no secret.

The **random oracle model** assumes the existence of a public, randomly-chosen function H that can be evaluated only by "querying" an oracle – which can be thought of as a magic box that returns $H(x)$ on input x.

This model provides a **formal methodology** that can be used to design and validate cryptographic schemes via the following two step approach:

1. First, a scheme is design and proven secure under the random oracle model. We assume the world contains a random oracle, and construct and analyze the scheme based on this assumption.
2. Since a random oracle is not available in the real world, to construct the scheme we "instantiate" the random oracle with a cryptographic function H (like SHA-1).

The hope is that the **cryptographic hash** is "sufficiently good" at emulating the random oracle, so that the security proof in the first step will carry over to the real world instantiation of the scheme.

- there is **no theoretical justification** for this presumptuous hope
- there exist schemes proven secure in the random oracle model which become insecure when instantiated in the real world (apparently some contrived scheme will fail no matter how the random oracle is instantiated too)
- it is **not clear** what it means for a **hash function to be "good" at emulating a random oracle**
    - it is **not clear** whether this is an achievable goal either
- a **proof of security** in the random oracle model should be **viewed as providing evidence** that **the scheme has no inherent design flaws**
    - it should not be taken as a rigorous proof that any real-world instantiation of the scheme is secure

Full domain hash

We assume H is a random function. When we model our attacker we give him access to **a signature oracle** and a **hashing oracle** (a random oracle). He doesn't need access to a verify oracle, he has all the info he needs.

Theorem: If f is TOWP and H is a random oracle, then the signature scheme below is $(t - O(q), q_S, q_H, q_H\varepsilon)$-secure.

$$Sig_{(n,d)}(m) = f_{sk}^{-1}\big(H(m)\big)$$

$$Vrfy_{(n,e)}(m, s) = \begin{cases} \text{valid}, & if\ f_{pk}(t) = H(m) \\ \text{invalid}, & if\ f_{pk}(t) \neq H(m) \end{cases}$$

Note: It is important that if $f: S \to S$ then $H: \{0,1\}^* \to S$, and it is important that H exhibits **full domain hashing** (just in case f is easily invertible on some part of S).

Proof:

Let A be our adversary. If A never queries H, then the H function will choose its output with uniform probability from S, and so we have:

$$\Pr\big[f_{pk}(t) = H(m)\big] = \frac{1}{|S|}$$

So, in this case, all the attacker can do is pick a tag t and guess an m such that $f_{pk}(t) = H(m)$.

Therefore, to increase its chances A must query H on a lot of messages, especially on the message m that A is attempting to forge, so that he can beat the $1/|S|$ bound.

Assume A queries H on m, if A wins then it found an m where the tag t is $f_{sk}^{-}\big(H(m)\big)$.

Alin Tomescu, CSE408
Tuesday, March 22nd, Lecture #15

Let's build a bigger attacker B to break f given the pk and the image y of an unknown element x through f (so $f(x) = y$). B runs A giving it access to the modified oracles H' and Sig'.

A outputs (m, t) such that $f_{pk}(t) = H(m)$.

To build H', pick $j \in \{1 \dots q_h\}$ randomly and let:

$$H'(m_i) = \begin{cases} random, i \neq j \\ y, i = j \end{cases}$$

Claim: H' is a random function.

We don't know sk, so we can't compute f_{sk}^{-1} but we can compute f_{pk}

Pick $s_1, \dots s_q$ (where $q = q_s + q_h$) as the signatures our Sig' oracle will be returning.

Let $h_i = f_{pk}(s_i)$. If the s_i are chose uniformly randomly and independently and f is a TOWP then the distribution of the h_i's will be uniformly randomly distributed and independent.

$$H'(m_i) = \begin{cases} h_i, if\ i \neq j \\ y, if\ i = j \end{cases}$$

$$Sig'(m) = \begin{cases} s_i, if\ m \neq m_j \\ FAIL\ if\ m = m_j \end{cases}$$

A's environment will be defined by the H' and Sig' oracles.

To successfully invert y

- A must output (m_j, t)
- (m_j, t) must be a valid forgery
- A did not query $Sig(m_j)$ (implied by the above since otherwise the Sig' oracle would have "failed" the experiment)
- the inverse of y would be $x = t$

$$Adv\ B = \frac{Adv\ A}{q_H + q_S}$$

$$Adv\ A = (q_H + q_S) AdvB \leq (q_H + q_S)\varepsilon$$