Msg Auth Codes (MAC), Hashing, Digital Signatures

Authentication

Alice sends Bob a message m (likely encoded but not our concern)

Authentication

Alice sends Bob a message m (likely encoded but not our concern)

Or does she?

Authentication

Alice sends Bob a message m (likely encoded but not our concern)

Or does she?

Maybe it was send by Eve!

Authentication

Alice sends Bob a message m (likely encoded but not our concern)

Or does she?

Maybe it was send by Eve!

We need Bob to be able to Authenticate it came from Alice.

Authentication

Alice sends Bob a message m (likely encoded but not our concern)

Or does she?

Maybe it was send by Eve!

We need Bob to be able to Authenticate it came from Alice.

Note: In this lecture we do not care what m is. It could be a ciphertext and perhaps should be called c. But we call it m. We are only concerned with authentication.

Authentication

Alice sends Bob a message m (likely encoded but not our concern)

Or does she?

Maybe it was send by Eve!

We need Bob to be able to Authenticate it came from Alice.

Note: In this lecture we do not care what m is. It could be a ciphertext and perhaps should be called c. But we call it m. We are only concerned with authentication.

Terminology: Security is not the right term. Non-forgeability is. We still use the term Security Parameter.

Formal Def of MAC

Def: A MAC is $\Pi = (GEN, MAC, V)$ where:

1. $GEN(1^n)$ is a uniform $k \in \{0,1\}^n$.
2. Given key k and msg m, $MAC_k(m) = t$, a tag. MAC_k is PPT.
3. $V_k(m,t) = 1$ if $MAC_k(m) = t$, 0 otherwise.

How to Use: Alice and Bob have $\Pi = (GEN, MAC, V)$

1. Alice generates k via GEN and sends it to Bob privately.
2. For Alice to send $m \in \{0,1\}^*$ to Alice computes $t = MAC_k(m)$ and sends (m,t).
3. Bob authenticates that its from Alice via by: $V_k(m,t) = 1$.

Note: We often restrict to $m \in \{0,1\}^{p(n)}$, p poly.

Example of a Message Authentication Code (MAC)

1. $k \in \{0, \ldots, p-1\}$ unif.
2. $MAC_k(m) = m + k$.
3. $V_k(m, t) = 1$ if $t = m + k$

Not Secure: If Eve has access to MAC_k or has old messages she knows $k = 7$.
Eve can Forge: If Eve has key k then she can forge.

Example of a Message Authentication Code (MAC)

1. $k \in \{0,1\}^n$ unif.
2. $MAC_k(m) = m \oplus k$.
3. $V_k(m,t) = 1$ if $t = m \oplus k$

Not Secure: If Eve has access to MAC_k or has old messages she knows k.
Eve can Forge: If Eve has key k then she can forge.

Need: A function f such that knowing f on a few values does not reveal what f is.

Example of a Message Authentication Code (MAC)

1. $k \in \{0,1\}^n$ unif.
2. $MAC_k(m) = m \oplus k$.
3. $V_k(m,t) = 1$ if $t = m \oplus k$

Not Secure: If Eve has access to MAC_k or has old messages she knows k.
Eve can Forge: If Eve has key k then she can forge.

Need: A function f such that knowing f on a few values does not reveal what f is.
We have them! Psuedo-Random Functions!

Construction of a Fixed Length MAC

Message are of length n
Let F be a PRF from $\{0,1\}^n$ to $\{0,1\}^n$.
MAC:

1. GEN: choose a uniform key $k \in \{0,1\}^n$ for F
2. $MAC_k(m)$: output $F_k(m)$
3. $V_k(m,t)$: output 1 iff $F_k(m) = t$

Theorem: Π is a non-forgeable MAC
Proof Sketch: If forgeable then F_k would not be psuedorandom.
Issue: We have not defined forgeable formally and we won't.

Drawbacks?

- This only works for *fixed-length* messages
- Since need tag t to be short, this only works for *short* messages

To get variable length we need a new Hardness Assumption.

Collision Resistant Hash Functions (CRHF)

Informal Def: A function H from $\{0,1\}^n$ to X where X is finite is Collision Resistant if it is HARD to find x, y such that $H(x) = H(y)$.

Common Hardness Assumption: There exists Collision Resistant Hash Functions.
Often keyed: H_k where k is a key. k of length n gives H on $\{0,1\}^n$.

Random Oracle Model (ROM)

Def: The Random Oracle Model is the Hardness Assumption that there exists a H such that both:

- H is a Collision Resistant Hash Functions.
- H is a Psuedorandom function.

Often keyed: H_k where k is a key.

Random Oracle Model: Warning

Compare the following Hardness Assumptions:

- Factoring is hard. Well tested. Fermat (1600's) worked on it! Easy to increase security parameter.
- RSA assumption. Worked on since 1978. But 40 years of modern math is a lot. Easy to increase security parameter.
- ROM. Hmmm. No candidate for the RO has been that well tested. The assumption H is random harder to test then Factoring is hard. Not clear how much increasing the security parameter will help.

But! There are real functions (in two slides) that are really being used that seem to satisfy ROM.

Possible Collision Resistant Hash Function

Security Parameter n
$H_k(x, y)$: k encodes (p, q, g, h) where

- p, q is an n-bit primes (who would have guessed! :-))
- g has a larger period mod $N = pq$ (omit how we can find such a g).
- h is some other random element of $\mathbb{Z}_N$.

$H : \mathbb{Z}_N \times \mathbb{Z}_N \rightarrow \mathbb{Z}_N$ is defined by

$$H(k, y) = g^k h^y \pmod{N}$$

Note: This is fixed length, but can use bigger and bigger security parameters so considered to be a function on $\{0, 1\}^n$.

More Collision Resistant Hash Functions

The following are really used! The definitions are ugly (like Trivium).

Hash Sch	Year Const	Numb bits	Year Broken
MD4	1990	128	1995
MD5	1992	128	1998
SHA1	1994	160	2005*
SHA-256	2005	256	Not Yet!

*SHA1 – collision found, but not quite broken.

Construction of a $\geq n$-length MAC

Message are of length $\geq n$
Let F_k be a PRF from $\{0,1\}^n$ to $\{0,1\}^n$.
Let H_k be a CRHF from $\{0,1\}^*$ to $\{0,1\}^n$.
Both keys are in $\{0,1\}^n$.
MAC:

1. GEN: choose a uniform key $k \in \{0,1\}^n$ for F and H
2. $MAC_k(m)$: output $F_k(H_k(m))$
3. $V_k(m,t)$: output 1 iff $F_k(H_k(m)) = t$

Theorem: Π is a non-forgeable MAC
Proof Sketch: If forgeable then F_k would not be psuedorandom OR H_k would not be CRHF.
Issue: We have not defined forgeable formally and we won't.

Drawbacks?

Alice: Bob, you signed a document saying you owe me $100,000

Bob: I didn't! And even if I did you can't prove it!

(Why do criminals on TV shows always say You can't prove I'm guilty? They should just lie and say I didn't do it.)

Need for the signature to be public!

Digital Signatures

Digital signatures

1. MAC uses private Key
2. MAC is good if Alice and Bob's only enemy is Eve.
3. MAC is bad if Bob says I didn't send that

Need a public key version of MAC that witnesses can verify.

Compare MAC's to Dig. Sig

- *Public verifiability*
  - Dig Sig: Anyone can verify a signature
  - MAC: Only a holder of the key can verify a MAC tag.
- *Transferable*
  - Dig Sig: Can forward a signature to someone else . . .
- Dig Sig: *Non-repudiation* Bob can't deny he signed!

Signature schemes

- A *signature scheme* is defined by three PPT algorithms (GEN, SIGN, V):
  - GEN: takes as input 1^n, outputs $sk, pk \in \{0,1\}^n$ (Secret Key, Public Key).
  - SIGN: takes as input a private key sk and a message $m \in \{0,1\}^*$; outputs a signature σ

$$\sigma \leftarrow SIGN_{sk}(m)$$

  - V: takes a public key pk, message m, and signature σ as input; outputs 1 or 0

$$\forall m, pk, sk[V_{pk}(m, SIGN_{sk}(m)) = 1]$$

First Attempt at a Signature Scheme

1. Alice GEN generates primes p, q of length n. p, q is private, $N = pq$ is public. Let $R = (p-1)(q-1)$. e, d such that $ed \equiv 1 \pmod{R}$. (N, e) public, (p, q, d) private. Alice has d, nobody else does.
2. For Alice to sign message m, Alice sends $\sigma = m^d \pmod{N}$.
3. For Bob to verify Alice computes σ^e

$$\sigma^d \equiv (m^d)^e \equiv m^{de} \equiv m^{de \pmod{R}} \equiv m \pmod{N}.$$

Looks Secure But Its Not

There are attacks on it that work.

Omitted.

But what to do?

Just a small adjustment.

Second Attempt at a Signature Scheme

Assume the Random Oracle Model. Assume Let H be a Random Oracle. It is public.

1. Alice GEN generates primes p, q of length n. p, q is private, $N = pq$ is public. Let $R = (p-1)(q-1)$. e, d such that $ed \equiv 1 \pmod{R}$. e public, d private. Alice has d, nobody else does.
2. For Alice to sign message m, Alice sends $\sigma = H(m)^d \pmod{N}$.
3. To verify Bob computes σ^e:

$$\sigma^e \equiv (H(m)^d)^e \equiv H(m)^{de} \equiv H(m)^{de \pmod{R}} \equiv H(m) \pmod{N}.$$

Secure?

Second Attempt at a Signature Scheme

Assume the Random Oracle Model. Assume Let H be a Random Oracle. It is public.

1. Alice GEN generates primes p, q of length n. p, q is private, $N = pq$ is public. Let $R = (p-1)(q-1)$. e, d such that $ed \equiv 1 \pmod{R}$. e public, d private. Alice has d, nobody else does.
2. For Alice to sign message m, Alice sends $\sigma = H(m)^d \pmod{N}$.
3. To verify Bob computes σ^e:

$$\sigma^e \equiv (H(m)^d)^e \equiv H(m)^{de} \equiv H(m)^{de \pmod{R}} \equiv H(m) \pmod{N}.$$

Secure?
Theorem: If a message can be forged then H is not a Random Oracle.

Second Attempt at a Signature Scheme

Assume the Random Oracle Model. Assume Let H be a Random Oracle. It is public.

1. Alice GEN generates primes p, q of length n. p, q is private, $N = pq$ is public. Let $R = (p-1)(q-1)$. e, d such that $ed \equiv 1 \pmod{R}$. e public, d private. Alice has d, nobody else does.
2. For Alice to sign message m, Alice sends $\sigma = H(m)^d \pmod{N}$.
3. To verify Bob computes σ^e:

$$\sigma^e \equiv (H(m)^d)^e \equiv H(m)^{de} \equiv H(m)^{de \pmod{R}} \equiv H(m) \pmod{N}.$$

Secure?
Theorem: If a message can be forged then H is not a Random Oracle.
Secure!