CSE 5852: Lecture 19

November 7, 2016

1 Last Class

Last class, we showed that public-key encryption "automatically" is secure for multiple messages even if they are adaptively chosen by the attacker. We also introduced a couple of ways to use public-key encryption:

1. Hybrid encryption: the public-key encryption is used to encrypt a symmetric key and the actual content for this message is protected using the symmetric encryption scheme.
2. Key encapsulation: the public-key encryption is used to encrypt a symmetric key and the participants talk using this key for the rest of their conversation.

However, neither of these methods remove the attacker in the middle problem. The attacker could (blindly) change the key transmitting using the public key encryption and then we don't know how any of our methods are going to work. (This includes the MAC, we don't know what it does if the two parties have different keys.)

What we really need is a method for an individual to send a message where its clear that the message came from them.

2 Public Key Signatures

We'll start by informally defining public-key signatures, work to get a construction and then come back to a formal definition. Like a public-key cryptosystem we want a triple of algorithms.

1. $(vk, sk) \leftarrow \mathsf{Gen}(1^n)$: this is the generate algorithm. It produces two keys a signing key or sk (the notational collision with secret key is unfortunate but this key does need to be secret). The second key is a public verification key.
2. $\sigma \leftarrow \mathsf{Sign}(sk, m)$: this is the signing algorithm. It takes in a message and the signing key to produce a signature σ. The idea is that only someone is possession of sk should be able to produce this signature.
3. $1/0 \leftarrow \mathsf{Vfy}(vk, m, \sigma)$: this is the verify algorithm. It takes in a message, signature and the verification key to check if this signature was produced using the signing key corresponding to this verification key.

2.1 Signatures and Encryption are not the same

Its somewhat unfortunate that signatures and encryption share a lot of the same notation. The existence of a pair of keys, a triple of algorithms. Its tempting to think we can build a signing algorithm out of an encryption algorithm. (Some people think you can use an encryption algorithm as a signing algorithm and swap the roles of the two keys. This is a very bad idea and doesn't work.)

As an example, signatures do not extend to multiple messages for free since the secret signing key is involved. An outside party has no ability to create valid signatures.

3 Constructing Signatures

To construct signatures we will take our last detour into number theory. Over the last month of the class we've work in the groups $\mathbb{Z}_p$ and $\mathbb{Z}_p^*$. We assumed that the discrete log problem was difficult to solve in $\mathbb{Z}_p^*$.

However, throughout the class we have worked with a prime modulus. A prime modulus had several advantages:

1. There was nothing private about p we only needed to know it was prime. It could be generated by anyone.

2. The only item excluded from the multiplicative group was 0. This is because for all $x \in \{1, ..., p-1\}$, $\gcd(x, p) = 1$.

3.1 Composite Modular Arithmetic

Let's go back to consider modular arithmetic for a number N that is not prime.

As before we can define the group $\mathbb{Z}_N$ which is a group under addition modulo N.

Let's think about how to create a group with respect to multiplication $\bmod N$. The main thing that we want to ensure is that group elements have an inverse. We'll show the following proposition

Proposition 1. *Let b, N be integers, with $b \geq 1, N > 1$. Then b is invertible* $\bmod N$ *if and only if* $\gcd(b, N) = 1$.

Proof. Assume b is invertible $\bmod N$, denote by $c = b^{-1}$. Since $bc \equiv 1 \mod N$ this implies that $bc - 1 = \gamma N$ for some $\gamma \in \mathbb{Z}$. Equivalently $bc - \gamma N = 1$. The gcd of two numbers is smallest positive integer that can be expressed as $Xb + YN = \gcd(b, N)$. Since 1 is the smallest positive integer it must be the $\gcd(b, N)$.

Now assume that $\gcd(b, N) = 1$. This means there exists integers X, Y such that $Xb + YN = 1$. Take the mod of both sides of this equation yields $Xb \equiv 1 \mod N$. We have that $X \mod N$ is the inverse of b. □

Lets now consider the set of elements b where $\gcd(b, N) = 1$. We'll call this group $\mathbb{Z}_N^*$. We want to show this is a group with respect to multiplication.

The properties we need to show are:

1. Identity $b \equiv 1 \mod N$ is the identity.

2. Inverse this exists by the way we have defined the group.
3. Commutativity follows from the property on the integers.
4. Associativity follows from the property on the integers.
5. Closure: Consider some a, b (which both have inverses $a^{-1}b^{-1}$. Then the value ab has an inverse $b^{-1}a^{-1}$ so $\gcd(ab, N) = 1$.

A natural question to ask is how many elements are in $\mathbb{Z}_N^*$. (Recall that $p-1$ elements were in $\mathbb{Z}_p^*$.) The thing we need to count is how many elements are relatively prime to N. Consider $N = p \cdot q$ where p, q are prime. Then we need to exclude all the multiples of p and the multiples of q.

Let's consider $N = 15$.

a	$\gcd(a, 15)$
1	1
2	1
3	3
4	1
5	5
6	3
7	1
8	1
9	3
10	5
11	1
12	3
13	1
14	1
15	15

In this case we're left with 8 values in $\mathbb{Z}_{15}^*$. In general we need to exclude q multiples of p and p multiples of q and add one back for double removal of pq. This means in total we remove $q + p - 1$ elements out of the total pq elements. This means the size of $|\mathbb{Z}_N^*| = (p-1)(q-1)$. We'll use this number frequently enough that we use the symbol ϕ to denote it. That is $\phi(N) \overset{def}{=} |\mathbb{Z}_N^*|$ and in the case where $N = p \cdot q$, $\phi(N) = (p-1)(q-1)$. Also note that we have $\phi(p) = p - 1$ for any prime p.

On the homework we showed that $a^{p-1} \equiv p$. This is actually true for any modulus (substituting ϕ).

Claim 1. *For any N and $a \in \mathbb{Z}_N^*$, $a^{\phi(N)} \equiv 1 \mod N$.*

Proof. First recall that $\phi(N)$ is the size of $\mathbb{Z}_N^*$. Consider the set of all elements $a_1, ..., a_{\phi(N)}$. Similar to the proof of Fermat's little theorem note that $aa_1 \not\equiv aa_2 \mod N$ because a has an inverse (this would imply that $a_1 \equiv a_2 \mod N$). This means that

$$a_1 \cdot a_2 \cdots a_{\phi(N)} \equiv (aa_1) \cdots (aa_{\phi(N)} \mod N.$$

Multiplying by the inverses of a_i yields that $1 \equiv a^{\phi(N)} \mod N$. □

We'll now introduce the function that we'll use for our (eventual signature scheme).

Theorem 1. *Fix some N. Fix some e such that $\gcd(e, \phi(N)) = 1$. Then define $f_e : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$ by $f_e(x) = x^e \mod N$. f_e is a permutation. Furthermore for $d = e^{-1} \mod \phi(N)$ then f_d is the inverse of f_e.*

4 The hardness of factoring

In order to use the above fact we need to define a computationally hard problem. The factoring problem is this: given N find integers $a, b > 1$ such that $N = ab$. We can solve this problem by performing division by all numbers $2, ..., \lceil \sqrt{N} \rceil$. This algorithm takes time approximately $O(\sqrt{N})$. There is no currently known algorithm that can solve this problem in time polynomial in the length of N. Note that our algorithm was polynomial in the size of N not in its logarithm.

However when we say that there is no polynomial time algorithm for this we mean algorithm that works in the worst case. Consider the following experiment:

weak factoring n:

1. Choose two uniform n-bit integers x_1, x_2.
2. Compute $N = x_1 \cdot x_2$.
3. $x'_1, x'_2 \leftarrow \mathcal{A}(N)$.
4. Output 1 if $x'_1 x'_2 = N$ and $x'_1, x'_2 > 1$.

Now notice that with good probability N will be even so we can't hope that this problem is hard on average. In fact the probability that N is divisible by a small factor is very high.

The numbers that are hardest to factor are those that have only large factors. As before we'll assume we have methods to generate large primes. We'll define a new factoring experiment based on this.

factoring n:

1. Choose two n-bit prime integers p, q using a special algorithm.
2. Compute $N = p \cdot q$.
3. $p, q \leftarrow \mathcal{A}(N)$.
4. Output 1 if $pq = N$ and $p, q > 1$.

Note that it is not sufficient to just choose random n bits primes. As this problem has been studied more mathematicians have become better at factoring certain classes of problems so we have to restrict our attention to certain classes of primes.

References