Notes for Lecture 5

1 Last time (Not previously!) on COS 533:

Authentification - MACs and Signature Schemes.

2 This time on COS 533: A thrilling episode

- Describe how to create digital signatures from OWF.
- Multiparty Comunication (MPC)

3 1-time Weak Signatures from OWF

Consider a OWF $F : \{0,1\}^\lambda \to \{0,1\}^{m(\lambda)}$

The following is a 1-time weak signature scheme called lamport signatures on messages on length n:

- $\text{Gen}(1^\lambda)$:
  $x_{i,b} \xleftarrow{\$} \{0,1\}^\lambda, i \in [n], b \in \{0,1\}$
  $y_{i,b} = F(x_{i,b})$
  $sk = (x_{i,b})_{\forall i \in [n], b \in \{0,1\}}$
  $pk = (y_{i,b})_{\forall i \in [n], b \in \{0,1\}}$
- Sign(sk,m) $= (x_{i,m_i})_{\forall i \in [n]}$
- $\text{Ver}(pk, m, (Z_i)_{\forall i \in [n]})$:
  Output 1 iff $F(Z_i) = y_{i,m_i} \forall i \in [n]$

Theorem 1 *If F is an OWF, (Gen,Sign,Ver) is a weakly EUF-CMA secure one time signature scheme.*

Intuition: If the adversary could sign a different message than the one message she gets to see, she must have inverted the OWF on all the points where the two messages differed.

This however, is not two time secure. A 2-time adversary could trivially query on 0^n, and 1^n, causing the challenger to reveal the whole secret key.

Note that this is only weakly secure. Let F' be another OWF, with a one bit smaller domain. If $F(x) = F'(x_{[1,\lambda-1]})$, in essence ignoring the last bit of the input, it can be easily shown that F is still a OWF. However an adversary who sees a signature on m could easily create another signature by flipping the last bit of every x_i revealed in the previous signature, which would still be evaluated the same by the OWF.

4 q-time Weakly Secure Digital Signature Scheme

Intuition: Just use q one time signatures.

This will be a stateful scheme, as it will remember how many signatures it has already signed.

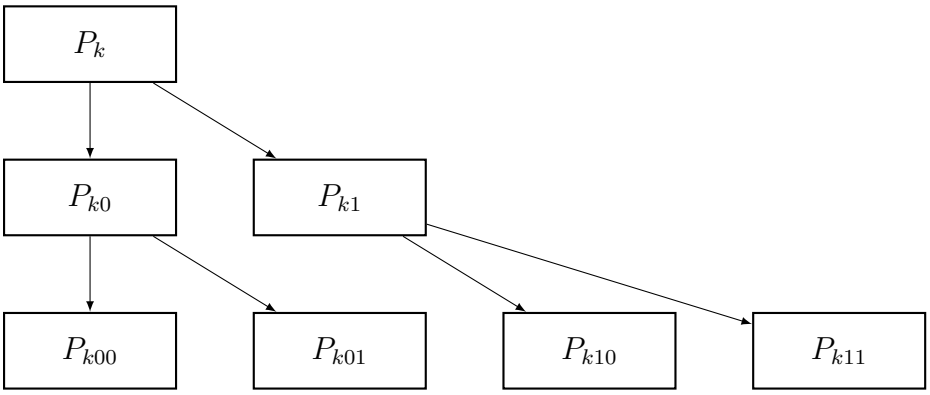
- $Gen_q(1^\lambda)$:
  Run $(sk_i, pk_i) \leftarrow Gen(1^\lambda)$ for $i = [1, q]$
  sk = $(sk_i)_{\forall i \in [1,q]}$
  pk = $(pk_i)_{\forall i \in [1,q]}$
- Sign(sk,m): for the ith message signed output $(i, sign(sk_i, m))$
- $Ver(pk, m, (i, \sigma)) = Ver(pk_i, \sigma)$

While this works, it is non-ideal. First problem is that public key size grows with q, which will limit us if we want to create a many time scheme.

5 Shrinking the public key

We will discuss a tree with only three levels, but you can imagine how to extend it to more.

Diagram:



Gen stays the same. In essence, we only want to output P_k, but then want to be able to sign with all the leaf secret keys. So we will use P_k to prove that the public keys associated with those secret keys are in fact ours.

Sign(sk,m) on $b_o b_1$...th message outputs the following.

The signature itself and the public key associated with it: $\sigma = Sign(sk_{b_0 b_1}, m)$ and $Pk_{b_0 b_1}$.

Proof that the $pk_{b_0 b_1}$ was not created by an adversary:

$pk_{b_0 \bar{b_1}}$ (We need to output this as well, because we won't be able to sign with sk_{b_0} again, as we are using a one time scheme)

$\sigma_{b_0 b_1} = Sign(sk_{b_0}, pk_{b_0 b_1} || pk_{b_0, \bar{b_1}})$

$pk_{b_0}, pk_{\bar{b_0}}$

$\sigma_{b_1} = Sign(sk, pk_{b_0} || pk_{\bar{b_0}})$

One problem is that in our 1-time scheme, the length of the public key increase in the length of the message you have to sign. This causes a blow up in the size of the public key to $m(\lambda) * 2n$.

Solution: Use a different one time signature key that can sign any length messages with only a constant size public key.

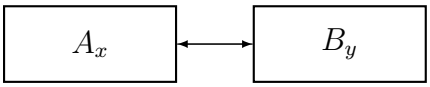
Next Steps:

- Generate the tree on the fly to reduce the size of sk.
- Make tree size exponential in λ, so we will never run out of leaves, giving us a many time scheme.
- Make stateless

6 Multi-Party Computation(MPC)

Today: 2 Party Communication (2PC)

Diagram:



Alice has a secret value x, and Bob has a secret value y. Alice wants to learn $f_A(x, y)$ and bob wants to learn $f_B(x, y)$ However, we want to make sure Bob doesn't learn x and Alice doesn't learn y. This is impossible in general - if $f_A(x, y)$ reveals the value of y, Alice must learn y. We will therefore change our requirment so that Alice only learns $f_A(x, y)$, and nothing more about y, and Bob only learns $f_B(x, y)$ and nothing more about x.

Today we will focus on a specific case of 2PC, called oblivious transfer(OT).

x = (x_0, x_1) y = b (a single bit)

$f_A(x, y) = \emptyset$

$f_B((x_0, x_1), b) = x_b$

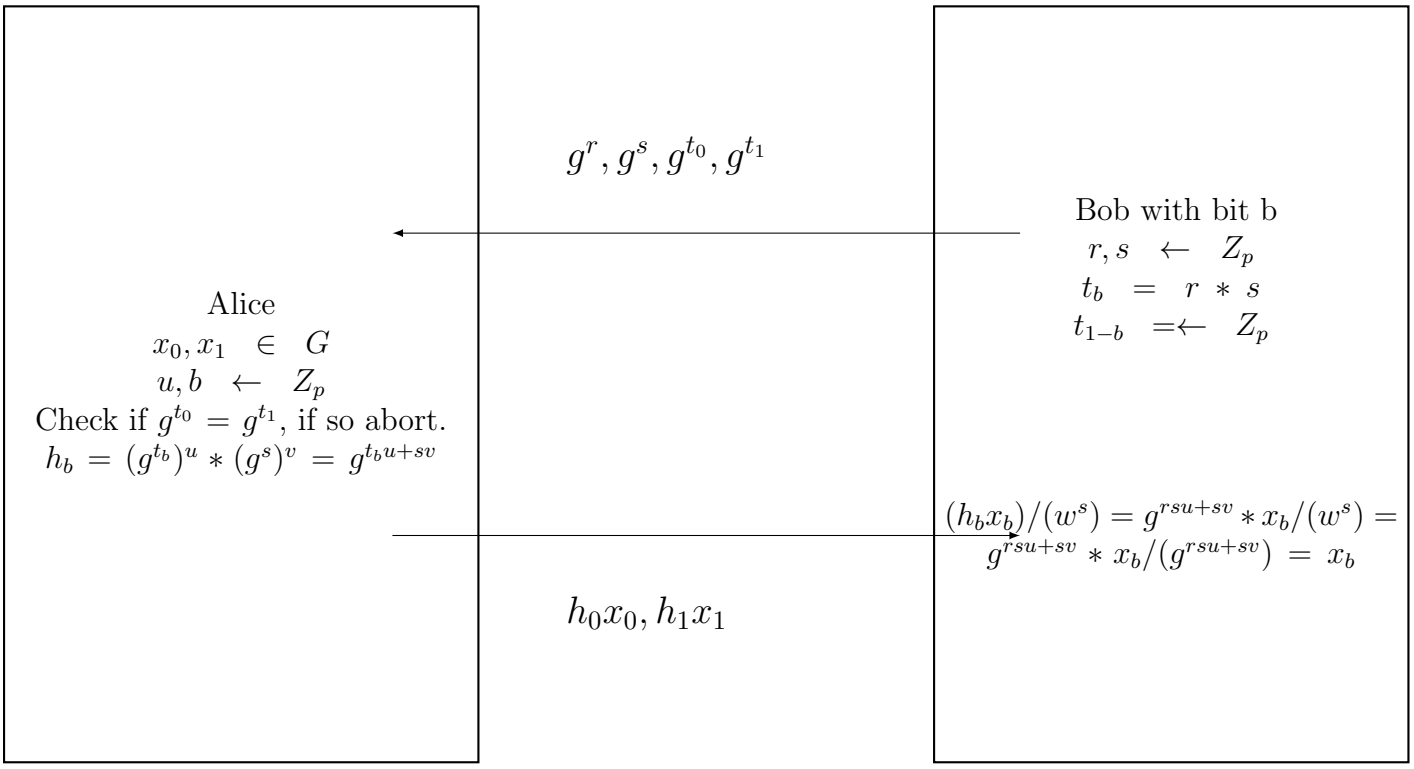
Basically, Alice wants to learn one of the messages x_0 or x_1 from Bob without revealing witch one she wants. Bob doesn't want her to learn anything about the message she didn't choose.

Let p be a large prime s.t q = 2p + 1 is also a prime.

$Z_q^* = \{1, ...q - 1\}$

Then there exists a subgroup $G \subset Z_q^*$ with order p. There also exists a generator g for G s.t $G = \{g^0, g^1, ...g^{p-1}\}$

(Not fully covered in lecture, but I think we want to pick this specific subgroup because it is one we are confident the DDH assumption (covered later) will be true on)



Receiver's (Bob's) Privacy:

Trivial to prove under the Decision Diffie Hellman (DDH) assumption, which states: $(g^r, g^s, g^r s)$ is indistinguishable from from (g^r, g^s, g^t), where $r, s, t \leftarrow Z_p$

In essence, DDH cane be applied on g^r, g^s and g^{t_b} to show that the Alice basically sees the generator raised to 4 random group elements.

Sender's Privacy:

Claim if $t_b = rs$ and $t_{1-b} \neq t_b$ then h_{1-b} is random in G, even condition on r,s,t_0, t_1, h_b, and w.

$h_b = g^{t_b u+sv} = w^s$, so we can ignore Bob's h_b information, as it can be recovered from the other values.

$Pr[h_{1-b} = g^z | w = g^y]$

$= Pr_{u,v}[t_{1-b}u + sv = z | ru + v = y]$

$= Pr_u[t_{1-b}u + s(y - ru) = z]$

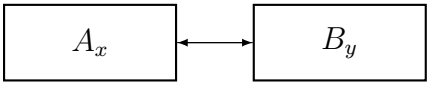
$= Pr_u[u = (z - sy)/(t_{1-b} - rs)] = 1/p$

Note that the division is well defined by our assumption that $t_{1-b} \neq rs$

We assumed that our adversaries are "Honest-but-curious". they don't deviate from the protocol, but are curious about the other value.

7 Next time on COS 533

We will cover general 2 Party Computation.



We will assume $f_A(k, y) = \emptyset$, and allow any $f_B(x, y)$

This is enough as long as the parties are honest, as they can just run the algorithm in twice.

If the parties are allowed to deviate from the protocol, it is impossible in general for both parties to learn their functions. Assume Bob sends the final message in the protocol. This means Bob must have already recovered $f_B(x, y)$, as he will get no more information. He can then choose not to send the final message and prevent A from learning her function. This is called the issue of fairness.