CS6111: Foundations of Cryptography Oct 31, 2018

Homework 4

Instructor: Shweta Agrawal *Due: Nov 9*

Notation. We let || denote the concatenation operator.

Problem 1: Zero Knowledge (9 pts)

Summarize the zero knowledge protocols for graph isomorphism, 3 coloring and Rudrata/Hamiltonian cycle that we saw in class. Argue correctness and soundness of the protocols formally. Define zero knowledge. Intuitively, why do you expect the protocols to be zero knowledge (formal proof is not expected).

Problem 2: Commitment and Encryption (5 pts)

The hiding and binding properties of commitments are reminiscent of the privacy and correctness properties of an encryption scheme. Discuss the differences between a commitment and an encryption scheme.

Problem 3: Identification Protocol (6 pts)

In an identification protocol, the prover P proves its identity to verifier V by demonstrating knowledge of a secret known to be associated with P without revealing the secret itself to V. A classic identification protocol works as follows.

First we fix a cyclic group G of prime order q. Let g be a generator for G. Next we choose a random $x \in \mathbb{Z}_q$ and set the prover's secret key $\mathsf{SK} = x$ and the verifier's key $\mathsf{PK} = g^x$. Now, the protocol works as follows:

1. The prover chooses a random $r \in \mathbb{Z}_q$ and sends $R = g^r$ to the verifier.
2. The verifier chooses a random challenge $c \in \{1, \ldots, B\}$ (for some fixed integer B) and sends c to the prover.
3. The prover computes $z = x \cdot c + r \in \mathbb{Z}_q$ and sends z to the verifier.
4. The verifier outputs "yes" if $g^z = \mathsf{PK}^c \cdot R$.

Thus, the verifier is convinced that it is speaking to someone who has knowledge of the discrete log x. We will study the correctness and security of the above protocol.

- **(2 pts)** Show that an honest prover who follows the protocol will always be accepted by the verifier (i.e. the verifier will say "yes").
- **(4 pts)** Suppose steps 1 and 2 of the protocol were swapped. Show that the resulting protocol is insecure. That is, an attacker who knows PK but not the secret x can play the role of the prover and cause the verifier to accept.

Problem 4: Online-Offline signatures (4 pts)

In the practice problems, we saw the definition of online-offline signatures. The idea of designing off-line/on-line signatures is to split the (expensive) signing process into two components. The off-line component will prepare some information σ_1 even before the message to be signed is known. This component could be a little slow since it is done off-line. The on-line component is performed after the message m arrives.

Let us see another construction. Assume $(\mathsf{Gen}^1, \mathsf{Sign}^1, \mathsf{Verify}^1)$ is a secure *one time* signature scheme and $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ be a regular signature scheme (existentially unforgeable under chosen message attack). In the offline phase, pick one time keys $(sk^1, vk^1) \leftarrow \mathsf{Gen}^1(1^k)$ and sign vk^1 using the regular signing algorithm $\sigma_1 = \mathsf{Sign}_{sk}(vk^1)$. In the online phase, one time sign the message $\sigma_2 = {\mathsf{Sign}^1}_{sk^1}(m)$. The overall signature is $\sigma = (\sigma_1, \sigma_2, vk^1)$.

Is the resulting scheme secure? Prove or provide an attack.

Problem 5: A Different Kind of Signature

The digital signature schemes we have studied so far all offer a property called **non-repudiability**. Informally, this property states that, since it is computationally difficult to forge someone's signature without knowing their private key, someone can only convincingly disavow a signature if they are willing to admit that their key has been compromised. This is often a desirable property, since signatures are no good for many applications if they can be easily revoked. However, there are some settings in which Alice may wish to authenticate her messages to Bob without committing herself to the contents in such a permanent way.

For such applications, we may wish to use a **designated verifier signature scheme**. In such a scheme, Alice uses Bob's public key to sign her messages in such a way that Bob could have forged the signatures himself using his private key. Upon receiving such a signature Bob will know that it came from Alice, because she is the only other party capable of producing the signature. However, any third party cannot be sure that Bob was not the signer instead, thus preserving Alice's deniability.

Formally, a DVS scheme is given by the following four efficiently-computable functions:

- $KeyGen(1^k) \rightarrow (pk, sk)$ generates a public key and secret key given a security parameter.
- $Sign(sk_{signer}, pk_{signer}, pk_{verifier}, m) \rightarrow \sigma$ produces a signature given a message, the signer's private key, and the public keys of both the signer and the verifier.
- $FakeSign(sk_{verifier}, pk_{verifier}, pk_{signer}, m) \rightarrow \sigma$ produces a signature given a message, the *verifier's* private key, and the public keys of both the signer and the verifier.
- $Verify(sk_{verifier}, pk_{verifier}, pk_{signer}, m, \sigma) \rightarrow b$ outputs 1 for accept and 0 for reject given a message and signature, the verifier's private key, and the public keys of both the signer and the verifier.

In order for a DVS scheme to be correct, the following must hold:

- $\Pr[\sigma \leftarrow Sign(sk_s, pk_s, pk_v, m); b \leftarrow Verify(sk_v, pk_v, pk_s, m, \sigma) : b = 1] = 1$ for all key pairs (pk_s, sk_s) and (pk_v, sk_v) in the domain of $KeyGen$ and for all m in the message domain. In other words, signatures produced by the signer must always verify.
- $\Pr[\sigma \leftarrow FakeSign(sk_v, pk_v, pk_s, m); b \leftarrow Verify(sk_v, pk_v, pk_s, m, \sigma) : b = 1] = 1$ for all key pairs (pk_s, sk_s) and (pk_v, sk_v) in the domain of $KeyGen$ and for all m in the message domain. In other words, signatures produced by the verifier must always verify.

The two key security properties of a DVS scheme are as follows:

- Unforgeability: For all PPT adversaries $\mathcal{A}$ there exists some negligible ν such that

$$\begin{aligned}
&\Pr[(pk_s, sk_s), (pk_v, pk_s) \leftarrow KeyGen(1^k);\\
&\qquad (m^*, \sigma^*, Q_{Sign}, Q_{FakeSign}) \leftarrow\\
&\qquad\qquad \mathcal{A}^{\mathcal{O}Sign(sk_s,pk_s,pk_v,\cdot),\mathcal{O}FakeSign(sk_v,pk_v,pk_s,\cdot),\mathcal{O}Verify(sk_v,pk_v,pk_s,\cdot,\cdot)}(pk_s, pk_v);\\
&\qquad b \leftarrow Verify(sk_v, pk_v, pk_s, m^*, \sigma^*) : m^* \notin Q \wedge b = 1]\\
&\le \nu(k)
\end{aligned}$$

  where Q_{Sign} is the list of $\mathcal{O}Sign$ queries made by $\mathcal{A}$ and $Q_{FakeSign}$ is the list of $\mathcal{O}FakeSign$ queries made by $\mathcal{A}$. In other words, no PPT adversary can find even an existential forgery with non-negligible probability, even when given the public keys of both parties and oracle access to the signing and verification functions.

- Non-transferability: For all PPT adversaries $\mathcal{A}$ there exists some negligible ν such that for all messages m and all signer and verifier key pairs (pk_s, sk_s) and (pk_v, sk_v)

$$\begin{aligned}
&\Pr[(pk_s, sk_s), (pk_v, pk_s) \leftarrow KeyGen(1^k);\\
&\qquad \sigma_0 \leftarrow Sign(sk_s, pk_s, pk_v, m); \sigma_1 \leftarrow FakeSign(sk_v, pk_v, pk_s, m);\\
&\qquad b \leftarrow \{0,1\}; b' \leftarrow \mathcal{A}(pk_s, sk_s, pk_v, sk_v, \sigma_b) : b' = b] \le 1/2 + \nu(k)
\end{aligned}$$

  In other words, a signature computed by the signer is indistinguishable from a signature computed by the verifier.

Suppose Alice and Bob have agreed on a large k-bit prime q ahead of time, and some generator g for $\mathbb{Z}_q^*$. In addition, let $F_k : \mathbb{Z}_q^* \to \mathbb{Z}_q^*$ be a family of PRFs with keys in $\mathbb{Z}_q^*$. Then the following scheme is a DVS:

- $KeyGen$: Choose x at random from $\mathbb{Z}_q$ and output g^x as the public key and x as the private key.
- $Sign$: Choose r at random from $\mathbb{Z}_q{}^*$, and compute $k = pk_v^{sk_s}$ and $m^* = m \cdot r \bmod q$. Output $\sigma = (r, F_k(m^*))$.
- $FakeSign$: Choose r at random from $\mathbb{Z}_q{}^*$, and compute $k = pk_s^{sk_v}$ and $m^* = m \cdot r \bmod q$. Output $\sigma = (r, F_k(m^*))$.
- $Verify$: Compute $k = pk_s^{sk_v}$ and $m^* = m \cdot r \bmod q$. Output 1 if $\sigma = (r, F_k(m^*))$, 0 otherwise.

a. Prove that this scheme is correct.

b. Prove that this scheme is perfectly non-transferable. In other words, prove that the probability of the signer outputting a particular signature is *exactly* the same as the probability of the verifier outputting that signature, for all possible messages and key pairs.

c. Prove that this scheme is computationally unforgeable using a hybrid argument:

   (a) Consider a modified game G_1 in which $Sign$, $FakeSign$, and $Verify$ use some fixed random element of $\mathbb{Z}_q^*$ in place of the computed value k. Prove using a reduction that no PPT adversary can distinguish between this game and the original forging game G_0 if DDH assumption holds.

   (b) Consider a modified game G_2 in which $Sign$, $FakeSign$, and $Verify$ use a truly random function in place of the PRF family F. Prove using a reduction that no PPT adversary can distinguish between this game and the previous game G_1 if F is a family of PRFs.

   (c) Conclude by the hybrid argument that since G_0 and G_2 are indistinguishable games. How often can an adversary expect to succeed when playing G_2? What does this tell you about the same adversary's success probability when playing G_0?