Homework 3

Adrián Enríquez Ballester

February 22, 2022

1 Collision resistant hash function?

Let $H : \mathcal{M} \to \mathcal{T}$ be a collision resistant hash function. We define $\hat{H}(x||b) := H(x)||b$ where b denotes one bit and $||$ denotes concatenation.

We are going to prove that $\hat{H}$ is also a collision resistant hash function. For that, let $\mathcal{A}_{\hat{H}}$ be an adversary for the collision resistance of $\hat{H}$. We can define an adversary $\mathcal{A}_H$ for the collision resistance of H which also plays the role of oracle for $\mathcal{A}_{\hat{H}}$:

$\mathcal{A}_H$:
  $(\mathtt{m}_1||\mathtt{b}_1,\ \mathtt{m}_2||\mathtt{b}_2) \leftarrow \mathcal{A}_{\hat{H}}()$
  $\mathtt{O}(\mathtt{m}_1,\ \mathtt{m}_2)$

It receives a pair of messages provided by $\mathcal{A}_{\hat{H}}$, removes its last bit and sends them to the oracle for the collision resistance of $\mathcal{A}_H$. Note that it is an efficient adversary, as it only removes the last bit of two messages.

Recall the collision resistance advantage of $\mathcal{A}_H$, which is the probability of it to win (i.e. the event $H(m_1) = H(m_2) \wedge m_1 \neq m_2$, where (m_1, m_2) is the pair of messages provided by $\mathcal{A}_H$ to its oracle). We are going to analyze this probability in terms of the advantage of $\mathcal{A}_{\hat{H}}$ itself.

On the one hand, if $\mathcal{A}_{\hat{H}}$ wins, then we have

$$\hat{H}(m_1||b_1) = \hat{H}(m_2||b_2)$$
$$H(m_1)||b_1 = H(m_2)||b_2$$

and $m_1||b_1 \neq m_2||b_2$. From the equality of the hashes we can extract $b_1 = b_2$ and $H(m_1) = H(m_2)$, so $m_1 \neq m_2$ and therefore $\mathcal{A}_H$ also wins.

On the other hand, if $\mathcal{A}_{\hat{H}}$ does not win, then

$$\hat{H}(m_1||b_1) \neq \hat{H}(m_2||b_2)$$
$$H(m_1)||b_1 \neq H(m_2)||b_2$$

or $m_1||b_1 = m_2||b_2$. In this case, $\mathcal{A}_H$ can still win if $H(m_1) = H(m_2)$ and $m_1 \neq m_2$, which is only possible if $b_1 \neq b_2$.

Given the above cases, the probability of $\mathcal{A}_H$ to win is as follows:

$$\begin{aligned}\mathsf{CRadv}[\mathcal{A}_H, H] &= \mathsf{CRadv}[\mathcal{A}_{\hat{H}}, \hat{H}] \cdot 1 + (1 - \mathsf{CRadv}[\mathcal{A}_{\hat{H}}, \hat{H}]) \cdot \varepsilon \\ &\geq \mathsf{CRadv}[\mathcal{A}_{\hat{H}}, \hat{H}]\end{aligned}$$

where

$$\varepsilon = \mathsf{Pr}[m_1 \neq m_2 \wedge b_1 \neq b_2 \wedge H(m_1) = H(m_2)]$$

If $\mathsf{CRadv}[\mathcal{A}_{\hat{H}}, \hat{H}]$ were not negligible, then $\mathsf{CRadv}[\mathcal{A}_H, H]$ would also be non negligible, which is not possible because H is collision resistant. This implies that $\hat{H}$ is also collision resistant.

2 Signatures vs. encryption

In some literature, digital signatures are sometimes described as an "inversion" of public key encryption schemes, where we treat a message m as a ciphertext of the encryption scheme and decrypt it using sk to produce a signature. To verify, the idea is then to encrypt and check whether cyphertext matches the message.

More concretely, let $\mathcal{E} = (\mathsf{G}_{\mathcal{E}}, \mathsf{E}, \mathsf{D})$ be a deterministic encryption scheme (i.e. algorithms E and D are deterministic). Then we can define a signature scheme $\Pi = (\mathsf{G}_{\Pi}, \mathsf{S}, \mathsf{V})$ as follows:

```
G_Π(n):                  S(m):              V(m, σ):
  (sk, pk) ← G_E(n)        σ ← D_sk(m)        c ← E_pk(σ)
  return (sk, pk)          return σ           if m = c:
                                                return 1
                                              else:
                                                return 0
```

For this exercise, we consider a weak security notion of digital signature scheme where the adversary does not get access to a signature oracle.

A signature scheme $\Pi = (\mathsf{G}_{\Pi}, \mathsf{S}, \mathsf{V})$ is unforgeable under a key only attack if for any PPT algorithm $\mathcal{A}$, the probability that the experiment $\mathsf{Sig-forge}^{ka}_{\mathcal{A},\Pi}(n)$ evaluates to 1 is negligible, where

```
Sig-forge^ka_A,Π(n):
  (sk, pk) ← G_Π(n)
  (m, σ) ← A(pk)
  return V(m, σ)
```

The advantage of the adversary $\mathcal{A}$ over Π under this security notion is the aforementioned probability:

$$\mathsf{Sig-forge}^{ka}\mathsf{adv}[\mathcal{A}_\Pi, \Pi] = Pr[\mathsf{Sig-forge}^{ka}_{\mathcal{A},\Pi}(n) = 1]$$

Te following attack shows that Π is not $\mathsf{Sig-forge}^{ka}$ secure:

```
𝒜:
  pk ← O()
  σ ←$ 𝒮
  m ← E_pk(σ)
  O(m, σ)
```

This adversary generates a random σ from $\mathcal{S}$, namely the signature space of Π and also the message space of $\mathcal{E}$. It then obtains the message m that matches this signature by performing the same operation used by V to verify. This trivially leads to $\mathsf{V}(m, \sigma)$ to output 1, as $\mathsf{E}_{pk}(\sigma) = m$.

Note that this adversary is efficient because it generates a random signature and uses E, which also has to be efficient. It always succeeds in creating a valid forgery, so its advantage is

$$\mathsf{Sig-forge}^{ka}\mathsf{adv}[\mathcal{A}_\Pi, \Pi] = 1$$

and then Π is not $\mathsf{Sig-forge}^{ka}$ secure.

3 Σ-protocol for Pedersen commitments

Let $\mathcal{G}$ be a group with a prime number q of elements where the discrete logarithm problem is hard. The public parameters of the Pedersen commitments are two group elements g_1 and g_2.

To commit an element $m \in \mathbb{Z}_q$, the committer has to choose a random element $x \in \mathbb{Z}_q$ and compute the commitment as $c \coloneqq g_1^m \cdot g_2^x$.

We are going to define some Σ-protocols for proving statements within this setting and prove its completeness, special soundness and honest-verifier zero-knowledge.

3.a $\mathsf{POK}[\exists m \exists x : c = g_1^m \cdot g_2^x]$

This protocol allows a prover to announce that he has committed a message without revealing his choice, which cannot be modified once the announcement has been made:

```
Prover((𝒢, g₁, g₂, c), (m, x)):
  (u₁, u₂) ← ℤ_q^2
  a :=_q g₁^u₁ · g₂^u₂
  ch ← Verifier(a)
  r₁ :=_q u₁ + ch · m
  r₂ :=_q u₂ + ch · x
  Verifier(r₁, r₂)
```

```
Verifier((𝒢, g₁, g₂, c)):
  a ← Prover()
  ch ←$ ℤq
  (r₁, r₂) ← Prover(ch)
  if g₁^r₁ · g₂^r₂ =q a · c^ch:
    return 1
  else:
    return 0
```

The name ch has been used to denote the challenge, although the letter c is used more often, because one of the parameters already has the name c.

3.a.1 Completeness

Completeness holds because

$$\begin{aligned} g_1^{r_1} \cdot g_2^{r_2} &= g_1^{u_1 + ch \cdot m} \cdot g_2^{u_2 + ch \cdot x} \\ &= g_1^{u_1} \cdot g_1^{ch \cdot m} \cdot g_2^{u_2} \cdot g_2^{ch \cdot x} \\ &= a \cdot g_1^{ch \cdot m} \cdot g_2^{ch \cdot x} \\ &= a \cdot (g_1^m \cdot g_2^x)^{ch} \\ &= a \cdot c^{ch} \end{aligned}$$

if the protocol has been followed properly.

3.a.2 Special soundness

Given two conversations with the same announcement where the challenge is different, $(a, ch, (r_1, r_2))$ and $(a, ch', (r'_1, r'_2))$ with $ch \neq ch'$, the witness can be recovered:

$$\begin{cases} g_1^{r_1} \cdot g_2^{r_2} &= a \cdot c^{ch} \\ g_1^{r'_1} \cdot g_2^{r'_2} &= a \cdot c^{ch'} \end{cases}$$

$$\begin{aligned} g_1^{r_1 - r'_1} \cdot g_2^{r_2 - r'_2} &= c^{ch - ch'} \\ g_1^{(r_1 - r'_1) \cdot (ch - ch')^{-1}} \cdot g_2^{(r_2 - r'_2) \cdot (ch - ch')^{-1}} &= c \\ g_1^{(r_1 - r'_1) \cdot (ch - ch')^{-1}} \cdot g_2^{(r_2 - r'_2) \cdot (ch - ch')^{-1}} &= g_1^m \cdot g_2^x \end{aligned}$$

$$\begin{cases} m &= (r_1 - r'_1) \cdot (ch - ch')^{-1} \\ x &= (r_2 - r'_2) \cdot (ch - ch')^{-1} \end{cases}$$

3.a.3 Honest-verifier zero-knowledgeness

Given a challenge ch. Take r_1 and r_2 at random from $\mathbb{Z}_q$ and compute

$$a = g_1^{r_1} \cdot g_2^{r_2} \cdot c^{-ch}$$

in order to generate a simulated conversation $(a, ch, (r_1, r_2))$.

The probability of a simulated conversation to happen is $1/q^2$ due to the randomness of r_1 and r_2 over $\mathbb{Z}_q$. It is the same probability that for an honest conversation with a fixed challenge ch, due to the randomness of u_1 and u_2 over $\mathbb{Z}_q$.

3.b $\mathsf{POK}[\exists m \exists x : c = g_1^m \cdot g_2^x \wedge c' = g_3^x]$

This protocol combines the previous one with a new statement where g_3 is also a public element of $\mathcal{G}$:

```
Prover((𝒢, g₁, g₂, g₃, c, c’), (m, x)):
  (u₁, u₂, u₃) ← ℤq³
  a₁ ≔q g₁^u₁ · g₂^u₂
  a₂ ≔ g₃^u₃
  ch ← Verifier(a₁, a₂)
  r₁ ≔q u₁ + ch · m
  r₂ ≔q u₂ + ch · x
  r₃ ≔q u₃ + ch · x
  Verifier(r₁, r₂, r₃)

Verifier((𝒢, g₁, g₂, g₃, c, c’)):
  (a₁, a₂) ← Prover()
  ch ←$ ℤq
  (r₁, r₂, r₃) ← Prover(ch)
  if g₁^r₁ · g₂^r₂ =q a₁ · c^ch and g₃^r₃ =q a₂ · c’^ch:
    return 1
  else:
    return 0
```

3.b.1 Completeness

Completeness holds because

$$\begin{aligned}
g_1^{r_1} \cdot g_2^{r_2} &= g_1^{u_1 + ch \cdot m} \cdot g_2^{u_2 + ch \cdot x} \\
&= g_1^{u_1} \cdot g_1^{ch \cdot m} \cdot g_2^{u_2} \cdot g_2^{ch \cdot x} \\
&= a_1 \cdot g_1^{ch \cdot m} \cdot g_2^{ch \cdot x} \\
&= a_1 \cdot (g_1^m \cdot g_2^x)^{ch} \\
&= a_1 \cdot c^{ch}
\end{aligned}$$

and

$$
\begin{aligned}
g_3^{r_3} &= g_3^{u_3 + ch \cdot x} \\
&= g_3^{u_3} \cdot g_3^{ch \cdot x} \\
&= a_2 \cdot c'^{ch}
\end{aligned}
$$

if the protocol has been followed properly.

3.b.2 Special soundness

Given two conversations with the same announcement where the challenge is different, $((a_1, a_2), ch, (r_1, r_2, r_3))$ and $((a_1, a_2), ch', (r_1', r_2', r_3'))$ with $ch \neq ch'$, the witness can be recovered:

$$
\begin{cases}
g_1^{r_1} \cdot g_2^{r_2} & = a_1 \cdot c^{ch} \\
g_1^{r_1'} \cdot g_2^{r_2'} & = a_1 \cdot c^{ch'}
\end{cases}
$$

$$
\begin{aligned}
g_1^{r_1 - r_1'} \cdot g_2^{r_2 - r_2'} &= c^{ch - ch'} \\
g_1^{(r_1 - r_1') \cdot (ch - ch')^{-1}} \cdot g_2^{(r_2 - r_2') \cdot (ch - ch')^{-1}} &= c \\
g_1^{(r_1 - r_1') \cdot (ch - ch')^{-1}} \cdot g_2^{(r_2 - r_2') \cdot (ch - ch')^{-1}} &= g_1^{m} \cdot g_2^{x}
\end{aligned}
$$

$$
\begin{cases}
m & = (r_1 - r_1') \cdot (ch - ch')^{-1} \\
x & = (r_2 - r_2') \cdot (ch - ch')^{-1}
\end{cases}
$$

But, in this case, we also have an alternative way to recover x:

$$
\begin{cases}
g_3^{r_3} & = a_2 \cdot c'^{ch} \\
g_3^{r_3'} & = a_2 \cdot c'^{ch'}
\end{cases}
$$

$$
\begin{aligned}
g_3^{r_3 - r_3'} &= c'^{ch - ch'} \\
g_3^{(r_3 - r_3') \cdot (ch - ch')^{-1}} &= c' \\
g_3^{(r_3 - r_3') \cdot (ch - ch')^{-1}} &= g_3^{x}
\end{aligned}
$$

$$
x = (r_3 - r_3') \cdot (ch - ch')^{-1}
$$

3.b.3 Honest-verifier zero-knowledgeness

Given a challenge ch. Take r_1, r_2 and r_3 at random from $\mathbb{Z}_q$ and compute

$$\begin{cases} a_1 & = g_1^{r_1} \cdot g_2^{r_2} \cdot c^{-ch} \\ a_2 & = g_3^{r_3} \cdot c'^{-ch} \end{cases}$$

in order to generate a simulated conversation $((a_1, a_2), ch, (r_1, r_2, r_3))$.

The probability of a simulated conversation to happen is $1/q^3$ due to the randomness of r_1, r_2 and r_3 over $\mathbb{Z}_q$. It is the same probability that for an honest conversation with a fixed challenge ch, due to the randomness of u_1, u_2 and u_3 over $\mathbb{Z}_q$.