Number Theoretic And Algebraic Cryptography

Or Dagmi - http://digmi.org

May 14, 2015

Lecture notes of the class of 2015. Lecturer: Dr. Zvika Brakerski

Course website: http://tiny.cc/ntalg15b

Contents

Chapter 1

Key Agreement

19/03/2015

1.1 Definition of the problem

We have two actors, Alice and Bob, and they like to communicate in a secure way. There is a channel of communication between them, but it is eavesdropped.

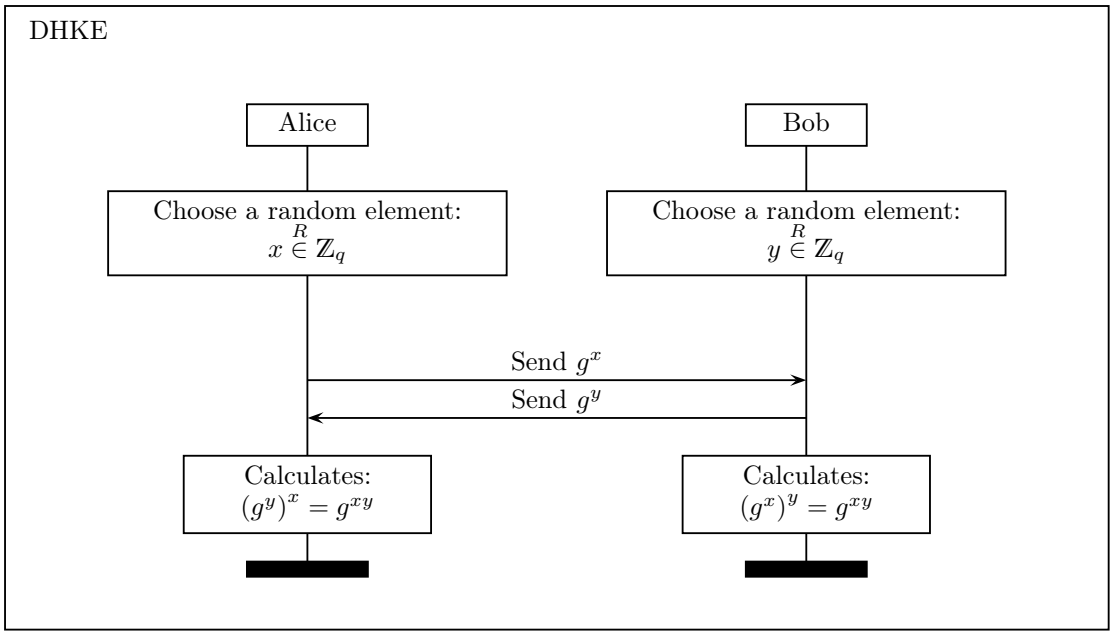
If Alice and Bob had a shared long randomness that is known only to them, they could communicate securely. But the problem is to obtain such long randomness.

So we want a way to obtain a random string using the channel when the eavesdropper cannot learn anything about the string.

1.2 Diffie-Hellman Key Exchange Protocol

We are generating a group $\mathbb{G}$ of order q with a generator g (we assume that q is prime because it is easier). The group is public and anyone knows it.

The protocol is as follows:



Note that now, Alice and Bob both obtain the same group element g^{xy}! But the eavesdropper (or the adversary) only see g^x and g^y.

We want to argue that this protocol is secure. This is not true in any case, because for example if the adversary has unbounded computational power he can find x and y (by simply check $g^1, g^2, \ldots$).

Remarks 1.2.1 Finding x while obtaining g^x is actually solving the *Discrete Log* problem.

But we still want to define security, So we need to limit the computational power of the adversary, and define how to. In order to do so, we will first define what is a "security parameter".

1.2.1 Security Parameter

Definition 1.2.2 (Security Parameter (k)) We allow the adversary to run in time poly(k). We say that the adversary "wins" if "advantage"$> \frac{1}{\text{poly}(k)}$.

Definition 1.2.3 (Negligible Function) $\varepsilon(k)$ is negligible if $\forall$polynomial p $\exists k_0$ $\forall k > k_0 \quad \varepsilon(k) < \frac{1}{p(k)}$.

Remarks 1.2.4 We usually abuse the notation and write $\varepsilon(k) = \text{negl}(k)$ (while we should have write something like $\varepsilon \in$ Negl, but no one does that).

So, we said that we need a group $\mathbb{G}$, we now want to define a family of groups for every k.

1.2.2 An example construction for the group $\mathbb{G}$

We are going to define the group $\mathbb{G}_k$ as follows:

1. Find a safe prime p of k-bits.

   Remarks 1.2.5 We say that p is prime if $q = \frac{p-1}{2}$ is also prime. There is a way to generate them, but we are not going to discuss how.

2. $\mathbb{G}_k = \text{QR}_p$ (All the quadratic residue in p, meaning all the elements that has square roots in the group).

Note that the order of $\mathbb{G}_k$ is prime (because p is safe prime and QR_p has $\frac{p-1}{2}$ the elements of $\mathbb{Z}_p$). It is conjectured that Discrete Log problem is hard in this group.

Note that Alice and Bob should be polynomial in k. But note that calculating g^x is logarithmic in the order of the group by repeated squaring. We calculate g square it and get g^2. Now we square g^2 and get g^4. We repeat the process and get $g^8, g^{16}, \ldots, g^{2^i}$. Now we decompose x into binary representation and then we can multiply the corresponding g^{2^i} and get g^x.

1.2.3 One way function

We claim that the function $\exp(g, x) \to (g, g^x)$ is one way function.

Definition 1.2.6 (One Way Function) We say that $f_k : \{0,1\}^k \to \{0,1\}^{k'}$ is a *One Way Function* (OWF) if:

1. f_k is computable in poly(k) time.
2. $\forall A$ PPT we have:
$$\Pr_{x \overset{R}{\in} \{0,1\}^k} \left[f(A(f(x), 1^k)) = f(x) \right] = \text{negl}(k)$$

So, if Discrete Log is hard for $\mathbb{G}$, then $\exp(g, x) \to (g, g^x)$ is a OWF.

1.2.4 Back to security

But is it enough? Can the adversary do other things other than simply solve Discrete Log?

The answer is that we do not know. We do not know how to prove the protocol only by Discrete Logarithm. In fact we know of groups that Discrete Log is hard, but still this protocol won't be secure.

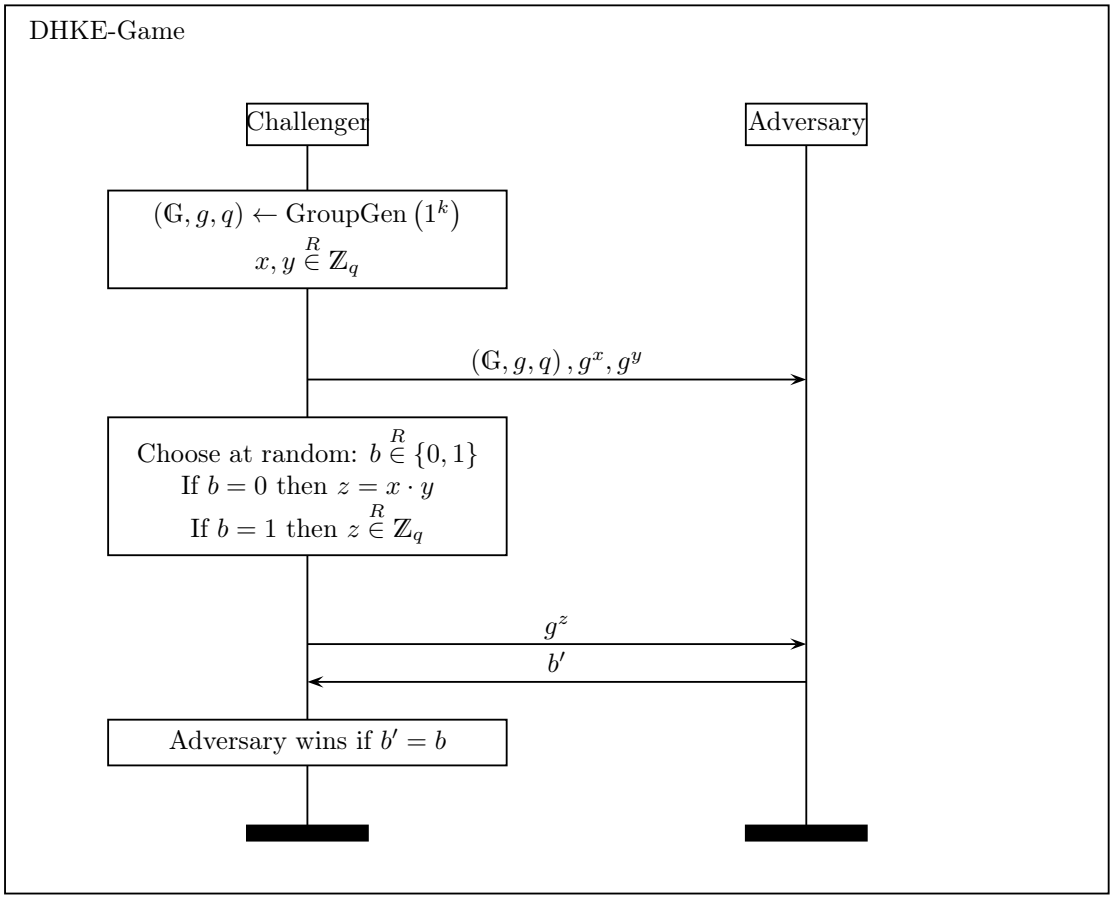
What is the view of the adversary? (what does he see)

The view of the adversary is:

1. $\mathbb{G}, g, q$.
2. g^x (for random x).
3. g^y (for random y).

And in this case, that is all. The goal of the adversary is to find g^{xy}.

We are going to define the game:



We want to evaluate the advantage of the adversary:

$$\begin{aligned} \text{DHKE} - \text{Adv}\left[A\right](k) &= \left| \Pr_{\text{Randomness of Challenger and Adversary}} \left[A\left(1^k\right) \text{ wins}\right] - 1/2 \right| \\ &= \frac{1}{2}\left|\Pr\left[A\left(1^k\right) = 1 \mid b = 0\right] - \Pr\left[A\left(1^k\right) = 1 \mid b = 1\right]\right| \end{aligned}$$

We do not know how to prove security for this protocol, but we use this protocol as a base line for other constructions.

We define an Hardness Assumption which encapsulates the hardness of this protocol.

We assume Decisional Diffie-Hellman (DDH). The assumption, with respect to GroupGen, will yield that the protocol is secure.

In order to define it formally we need some definitions:

Definition 1.2.7 (Computational Indistinguishability) $X = \{X_k\}$, $Y=\{Y_k\}$. We say that X and Y are *Computational Indistinguishable* if:

$$\forall A \text{ PPT} \quad \text{DistAdv}[A] = \frac{1}{2}\left|\Pr_{\substack{R\\ x\in X_k}}\left[A\left(1^k, x\right)=1\right] - \Pr_{\substack{R\\ y\in Y_k}}\left[A\left(1^k, y\right)=1\right]\right| = \text{negl}(k)$$

We will write $X \cong Y$.

Now we can define DDH:

Definition 1.2.8 (Decisional Diffie-Hellman (DDH)) With respect to GroupGen:

$$((\mathbb{G}, g, q), g^x, g^y, g^{xy}) \cong ((\mathbb{G}, g, q), g^x, g^y, g^u)$$

For a uniform u. Also:

$$(g, g^x, g^y, g^{xy}) \cong (g, g^x, g^y, g^u)$$

Chapter 2

Public Key Encryption

2.1 Definition

We now have two players, Encryptor and Decryptor. We want that the decryptor will have a secret key sk and that the encryptor will have a public key pk (which is also available for every one who eavesdrop the line. And we want that the encryptor will have a message m and he will send to the decryptor $\mathrm{Enc}(m)$ and that the eavesdropper won't be able to learn anything about m.

Definition 2.1.1 (Public Key Encryption (PKE)) PKE system is three algorithms:

1. $\mathrm{KeyGen}\left(1^k\right) \to (\mathrm{sk}, \mathrm{pk})$.
2. $\mathrm{Enc}_{\mathrm{pk}}\left(1^k, m\right) \to c$, where $m \in M_k$ (the message space).
3. $\mathrm{Dec}_{\mathrm{sk}}\left(1^k, c\right) \to m$.

Such that:

Correctness:

$$\mathrm{Dec}_{\mathrm{sk}}\left(\mathrm{Enc}_{\mathrm{pk}}\left(m\right)\right) = m$$

This is a definition for perfect correctness. But we can also define in in a probabilistic way:

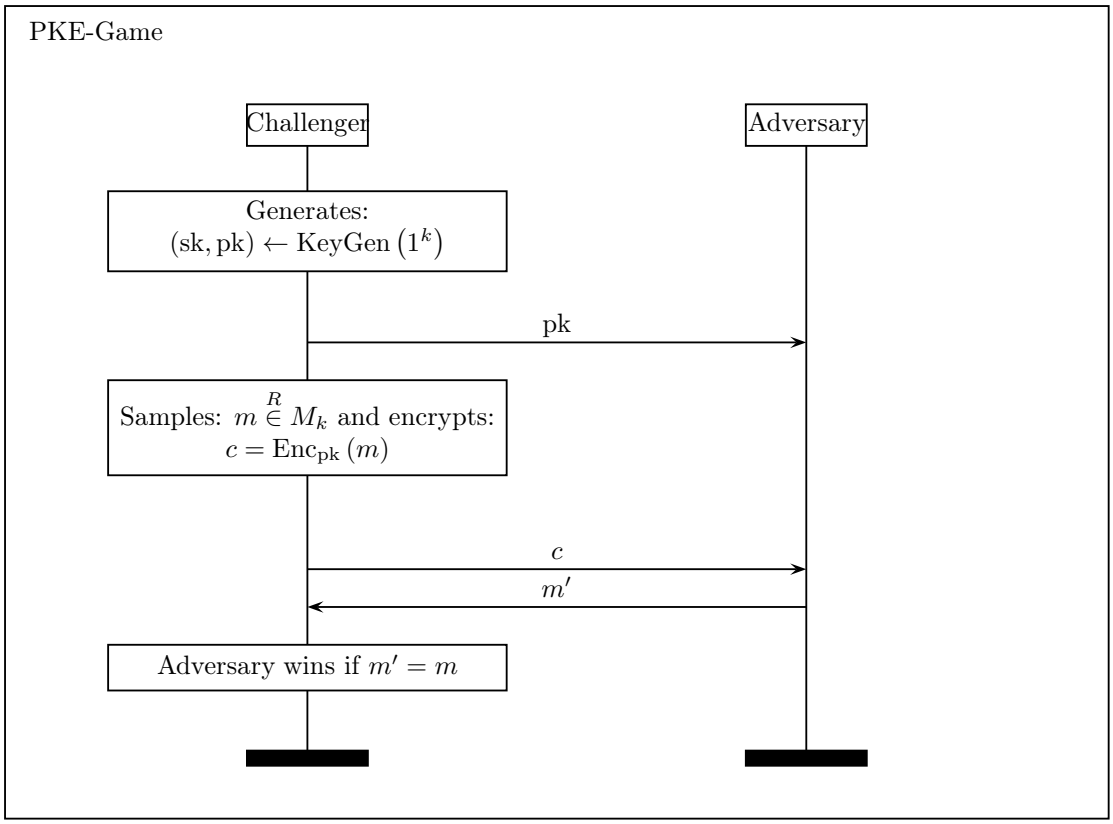
$$\forall m \quad \Pr_{\substack{(\mathrm{sk},\mathrm{pk}) \leftarrow \mathrm{KeyGen}\left(1^k\right) \\ r}} \left[\mathrm{Dec}_{\mathrm{sk}}\left(\mathrm{Enc}_{\mathrm{pk}}\left(m; r\right)\right) \neq m\right] = \mathrm{negl}\left(k\right)$$

With out loss of generation, the Dec algorithm can be deterministic but the KeyGen and Enc should be randomized.

Remarks 2.1.2 We give the KeyGen the parameter 1^k so that the algorithm will be able to run polynomial time in the security parameter. If we would have given it only k then it will only be $\log k$.

2.2 Security

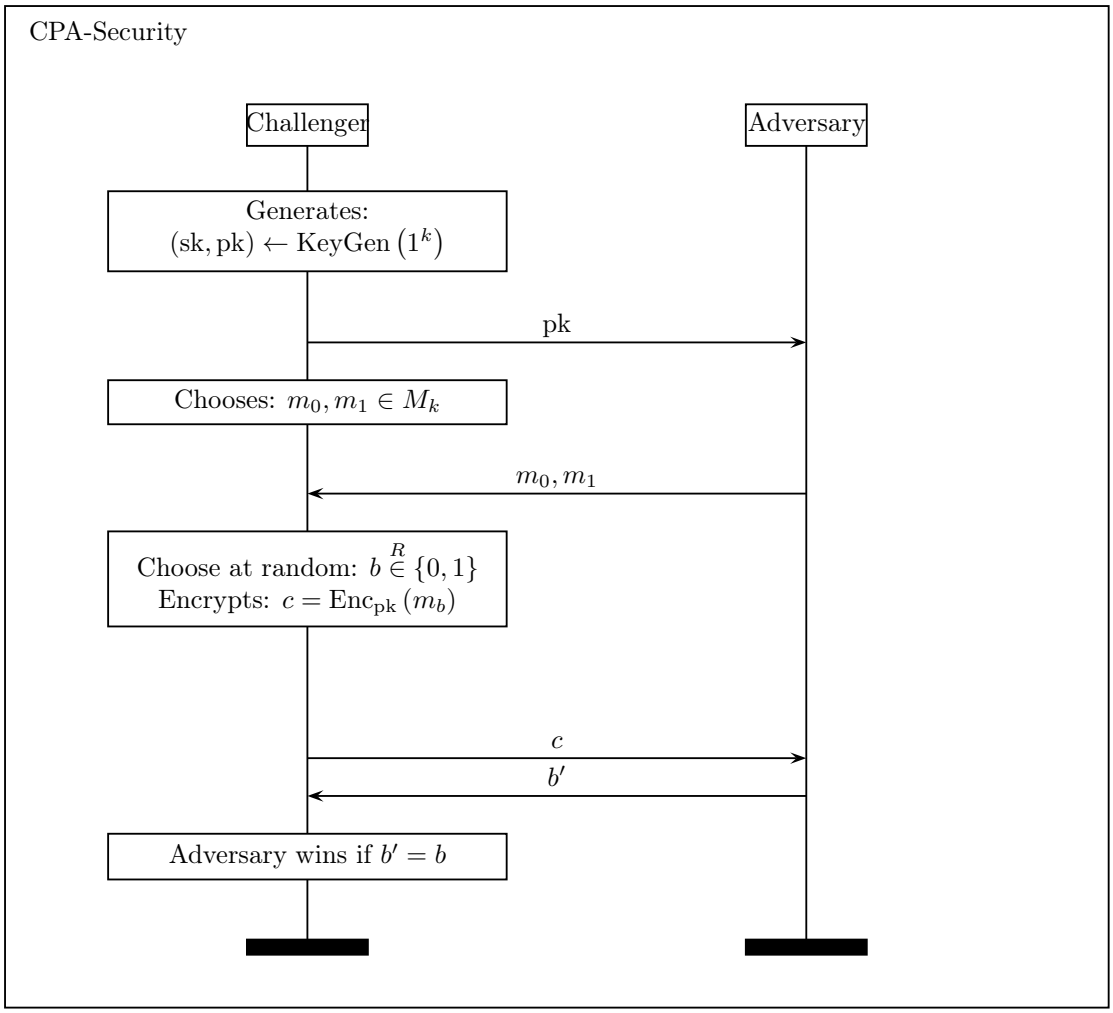
Again we are going to define a game between the challenger and the adversary:



This is a one way to define security. This definition doesn't cover everything. For example, the adversary might be able to learn something about m but not everything. If the message space is very small (for example $0, 1$) this definition is good.

2.2.1 Chosen Plaintext Attack (CPA)

A better definition is:



This notion of security is called security against Chosen Plaintext Attack.

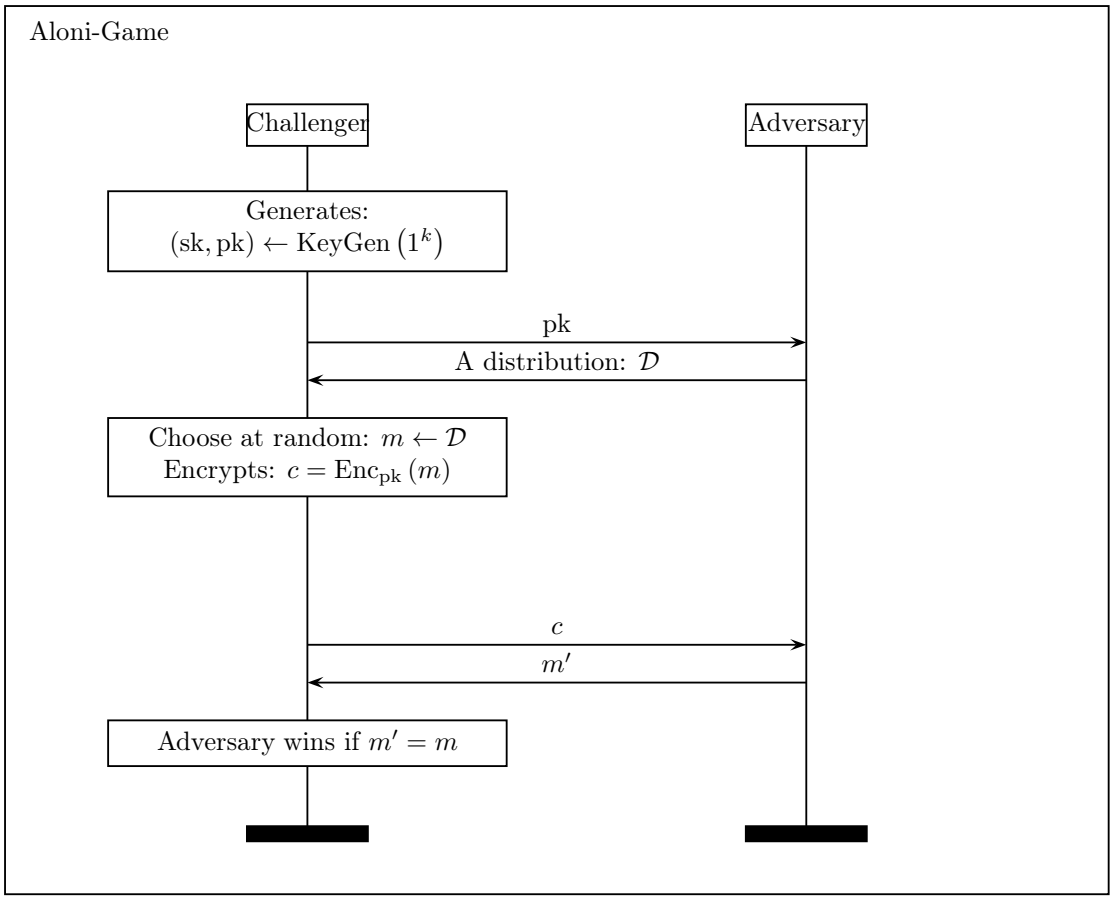
The definition for the encryption to be secure is:

$$\mathrm{CPA} - \mathrm{Adv}\left[A\right] = \left|\Pr\left[A \text{ wins}\right] - 1/2\right| = \mathrm{negl}\left(k\right)$$

But why can't the adversary can simply encrypt m_0 and m_1 and compare with c? That's why Enc need to be probabilistic algorithm. So there are many encryptions for m_0 and m_1, and the adversary should not be able to distinguish between both sets. 24/03/2015

2.2.2 Aloni Attack

Aloni asked after last lecture, can we generalize the above definition. For example:



We would define the advantage as follows:

$$\mathrm{Aloni} - \mathrm{Adv}[A] = \Pr[m = m'] - \max_x \Pr[\mathcal{D} = x]$$

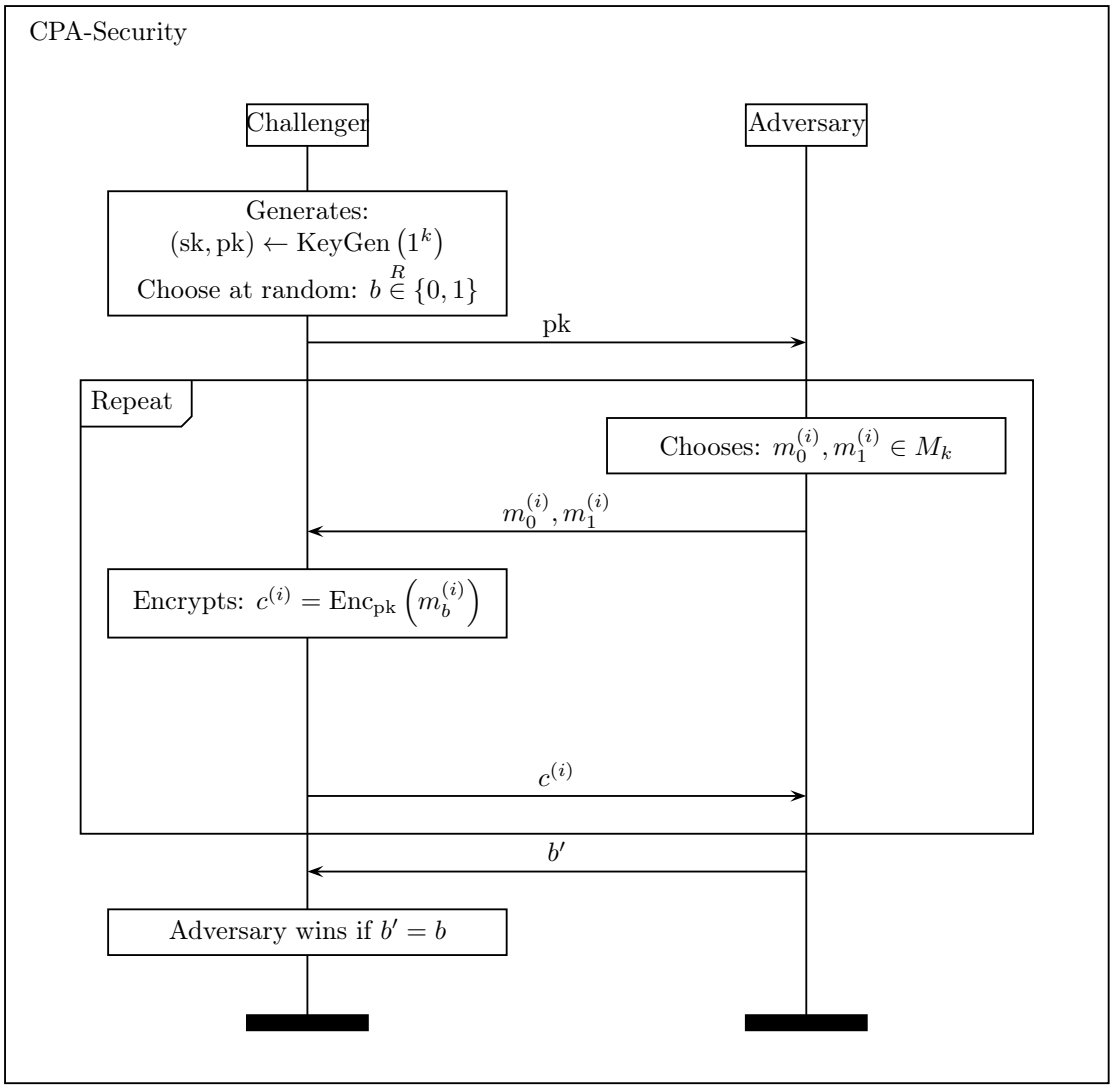
Remarks 2.2.1 It is not clear that this makes sense, It might not. Think about it at home.

Note that encryption scheme that is secure against Aloni's is more powerful than the CPA, because we can choose a distribution that with probability 0.5 choose between m_0 and m_1. Meaning that if we have adversary that can break Aloni's game, he can break also normal CPA.

Meaning that we would rather have an encryption scheme that will be secure in the term Aloni introduced than the normal CPA.

2.2.3 MMCPA (Multiple message CPA)

Now we want to consider the following game:



We define the advantage:

$$\begin{aligned} \mathrm{MMCPA-Adv}\left[A\right](k) &= \left|\Pr\left[b' = b\right] - 1/2\right| \\ &= \frac{1}{2}\left|\Pr\left[b' = 1 \mid b = 0\right] - \Pr\left[b' = 1 \mid b = 1\right]\right| \end{aligned}$$

And we of course say that encryption scheme is secure if any PPT A has an $\mathrm{MMCPA-Adb}\left[A\right](k) = \mathrm{negl}\left(k\right)$.

Theorem 2.2.2

If E is $\mathrm{CPA-Secure} \iff$ *E is* $\mathrm{MMCPA-Secure}$.

Proof: It is trivial that if E is $\mathrm{MMCPA-Secure}$ than it is also $\mathrm{CPA-Secure}$. We want to show the other direction.

Let E be a $\mathrm{CPA-Secure}$ scheme. And let A be a PPT. Denote $\varepsilon\left(k\right) = \mathrm{MMCPA-Adv}\left[A\right](k)$. Our goal is to prove that $\varepsilon\left(k\right) = \mathrm{negl}\left(k\right)$.

Denote $t\left(k\right)$ as the running time of A, and because A is a PPT then $t\left(k\right) = \mathrm{poly}\left(k\right)$.

We are going to define Hybrids.

Hybrid H_0: The challenger is the same as in MMCPA with $b = 0$.

$$\Pr_{H_0}\left[b' = 1\right] = \Pr_{\mathrm{MMCPA}}\left[b' = 1 \mid b = 0\right]$$

Hybrid H_j:

$$c^{(i)} = \begin{cases} \mathrm{Enc}_{\mathrm{pk}}\left(m_1^{(i)}\right) & i \leq j \\ \mathrm{Enc}_{\mathrm{pk}}\left(m_0^{(i)}\right) & i > j \end{cases}$$

Now, note that:

$$\Pr_{H_{t(k)}}\left[b' = 1\right] = \Pr_{\mathrm{MMCPA}}\left[b' \mid b = 1\right]$$

We want to define an adversary B_j^A (the superscript A means that B can run A), algorithms which will be adversaries to the original CPA game.

So, B plays agains CPA challenger:

1. Receive pk and forward it to A.

2. For $i = 0$ to $t\,(k)$:

   (a) Receive $\left(m_0^{(i)}, m_1^{(i)}\right)$ from A

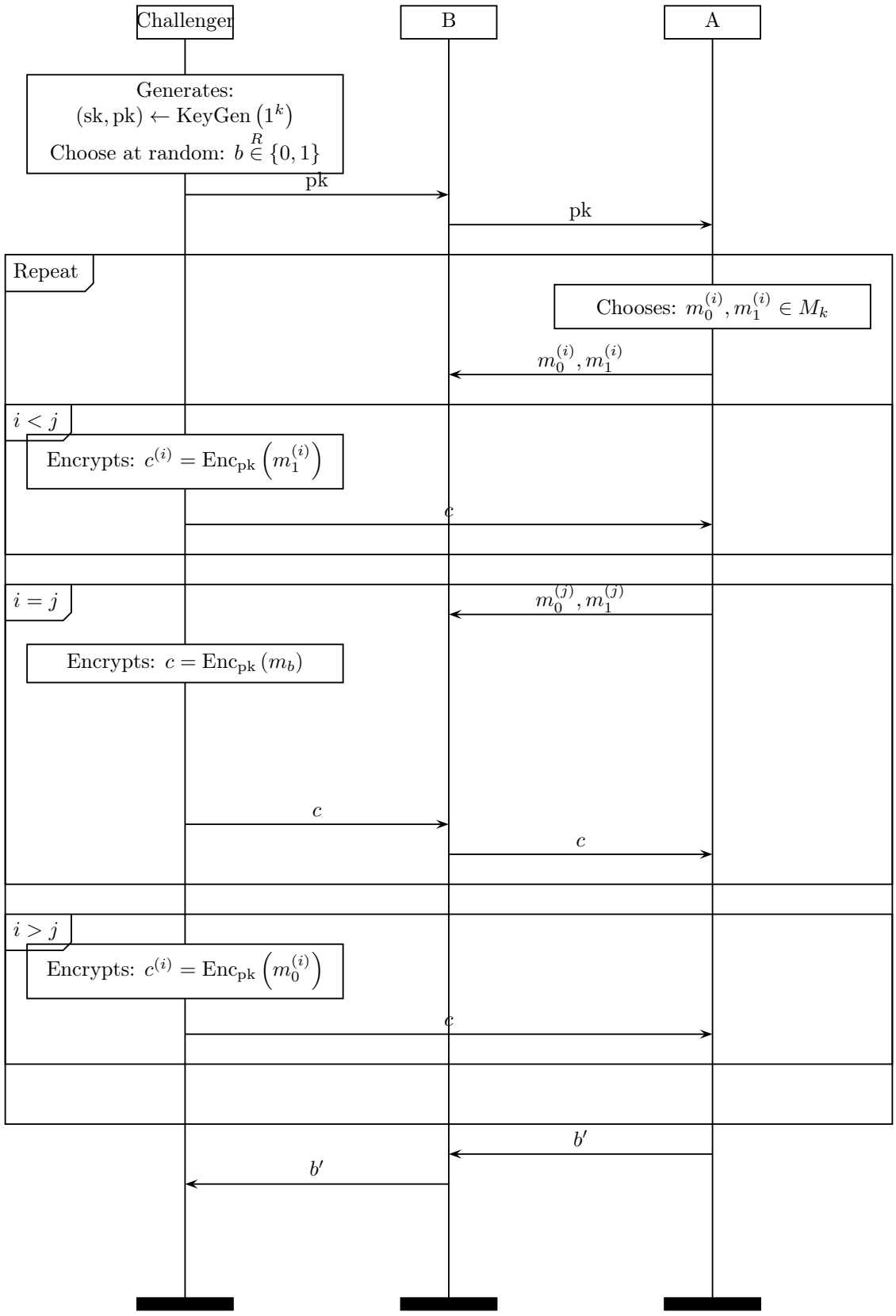
   (b) If $i < j$ send A: $c^{(i)} = \mathrm{Enc}_{\mathrm{pk}}\left(m_1^{(i)}\right)$.

   (c) If $i = j$ send the challenger $\left(m_0^{(j)}, m_1^{(j)}\right)$ and send the response to A

   (d) If $i > j$ send A: $c^{(i)} = \mathrm{Enc}_{\mathrm{pk}}\left(m_0^{(i)}\right)$.

3. Receive b' from A. return $\beta' = b'$.

CPA-Security



Note that the challenger's choice of b differ between H_j and H_{j-1}. Meaning that if we can distinguish between two consecutive hybrids, we can break the CPA security. Because:

$$\begin{aligned}\text{CPA}-\text{Adv}\left[B_j\right](k) &= \frac{1}{2}\left|\Pr\left[\beta'=1 \mid \beta=0\right]-\Pr\left[\beta'=1 \mid \beta=1\right]\right| \\ &= \frac{1}{2}\left|\Pr_{H_{j-1}}\left[b'=1\right]-\Pr_{H_j}\left[b'=1\right]\right|\end{aligned}$$

So we have:

$$\begin{aligned}\text{MMCPA}-\text{Adv}\left[A\right](k) &= \frac{1}{2}\left|\Pr_{H_{t(k)}}\left[b'=1\right]-\Pr_{H_0}\left[b'=1\right]\right| \\ &\leq \sum_{j=1}^{t(k)}\text{CPA}-\text{Adv}\left[B_j\right] \\ &\leq t\left(k\right)\cdot\text{CPA}-\text{Adv}\left[B_{j^*}\right]\end{aligned}$$

Where we used the triangle inequality in the second step, and the last, we defined B_{j^*} as the best adversary against CPA. ■

Remarks 2.2.3 Note that another option is to choose j at random. And consider the adversary B_{rand}. Note that:

$$\begin{aligned}\text{CPA}-\text{Adv}\left[B_{\text{rand}}\right](k) &= \frac{1}{2}\left|\Pr\left[\beta'=1 \mid \beta=0\right]-\Pr\left[\beta'=1 \mid \beta=1\right]\right| \\ &= \frac{1}{2}\left|\frac{1}{t}\sum\Pr_{H_j}\left[b'=1\right]-\frac{1}{t}\sum\Pr_{H_{j-1}}\left[b'=1\right]\right| \\ &= \frac{1}{2t}\left|\sum\Pr_{H_j}\left[b'=1\right]-\Pr_{H_{j-1}}\left[b'=1\right]\right| \\ &= \frac{1}{2t}\text{MMCPA}-\text{Adv}\left[A\right]\end{aligned}$$

Remarks 2.2.4 The idea of the B_{j^*} is used in a non-uniform variant of the definition, where the adversary is also allowed to have an advice string that depends only on the security parameter (telling him which is the best hybrid to attack for example).

We showed that for every adversary A for the MMCPA there exists adversary $B\left(1^k,j\right)$ for CPA s.t. $\forall j,k$ we have:

$$\Pr\left[B\left(1^k,j\right)=1 \mid \beta=1\right]-\Pr\left[B\left(1^k,j\right)=1 \mid \beta=0\right]= \Pr_{H_j}\left[A\left(1^k\right)=1\right]-\Pr_{H_{j-1}}\left[A\left(1^k\right)=1\right]$$

From that we derived that for all k we have:

$$\sum_{j=1}^{t(k)}\left(\Pr\left[B\left(1^k,j\right)\text{ wins (1M)CPA}\right]-1/2\right)=\Pr\left[A\text{ wins MMCPA}\right]-1/2 \qquad (\star)$$

We now want to bound the sum. For that, we showed two solutions:

1. The non-uniform solution:

   $$\forall k\ \exists j^*\left(k\right)\quad \forall j\ \left|\Pr\left[B\left(1^k,j\right)\text{ wins}\right]-1/2\right|\leq\left|\Pr\left[B\left(1^k,j^*\right)\text{ wins}\right]-1/2\right|$$

   We define: $B^*\left(1^k\right)=B\left(1^k,j^*\left(k\right)\right)$. Thus:

   $$\left|\Pr\left[A\text{ wins MMCPA}\right]-1/2\right|\leq t\left(k\right)\left|\Pr\left[B^*\left(1^k\right)\text{ wins CPA}\right]-1/2\right|$$

   And now, the right hand side, is only a function of k instead of function of j and k.

Remarks 2.2.5 We are allowing the adversary to have an advice (the j^*) that depends only on the security parameter).

2. The random solution:
   We define $B^{**}\left(1^k\right)$ as follows: $j \overset{R}{\in} [t(k)]$ and run $B\left(1^k, j^{**}\right)$. As an exercise prove that it works here. It doesn't always work. The reason that it work here is because that the equation $(\star)$ is true without absolute value.

Chapter 3

Groups

3.1 Definition

Definition 3.1.1 (Group) A group is a set of elements and an operation: $\mathbb{G}=(S,\cdot)$ where: $\cdot : S \times S \to S$. And it maintains:

1. If $g,h \in \mathbb{G} \to g \cdot h \in \mathbb{G}$.
2. $g_1,g_2,g_3 \in \mathbb{G} \to g_1 \cdot (g_2 \cdot g_3) = (g_1 \cdot g_2) \cdot g_3$.
3. $\exists 1 \in \mathbb{G}$ s.t. $\forall g \in \mathbb{G}$: $g \cdot 1 = 1 \cdot g = g$.
4. $\forall g \in \mathbb{G}\ \exists h \in \mathbb{G} \quad g \cdot h = h \cdot g = 1$

Definition 3.1.2 (Commutative Group) We say that the group is commutative if: $\forall g,h\ g \cdot h = h \cdot g$.

Definition 3.1.3 (Cyclic Group) We say that a group is cyclic if it is generated by a single element g (called a generator) and denote it by $\langle g \rangle$. Thus the group is:

$$g^0 = 1, g^1, g^2, g^3, \ldots, g^q = g^0 = 1$$

And we say that q is the order of $\langle g \rangle$ (also the order of the element g).

Remarks 3.1.4 We are only talking about finite groups.

Remarks 3.1.5 Note that $g^x = g^{x \pmod q}$. Because, take a look at: $x = y + qz$. We can write:

$$g^x = g^{y+qz} = g^y \left(g^q\right)^z = g^y \left(1\right)^z = g^y$$

3.2 What can we calculate easily?

Note that if we have: g^x, g^y we can easily calculate g^{x+y} by simply multiplying them.

Now lets say that we have h and we want to calculate h^{-1}, we can simply do that by raising it to the exponent $q-1$, h^{q-1} by the binary decomposition method we've seen.

Now, assume we have: g^x, g^y, a, b and we want to calculate: g^{ax+by}, by calculating $\left(g^x\right)^a, \left(g^y\right)^b$ and taking the product.

We can also consider $g^{\underline{x}}, \underline{a}$ (where $\underline{x}, \underline{a}$ are vectors) and we can calculate: $g^{(\underline{x},\underline{a})}$ from the same reason of the last.

We can also consider the case: $\left(g^M\right)_{i,j} = g^{M_{i,j}}$ where g^M is an $n \times m$ matrix. And then given two matrices: L which is $l \times n$ and R which is $m \times r$ and calculate: $g^{L \cdot M \cdot R}$ which is an $l \times r$ matrix.

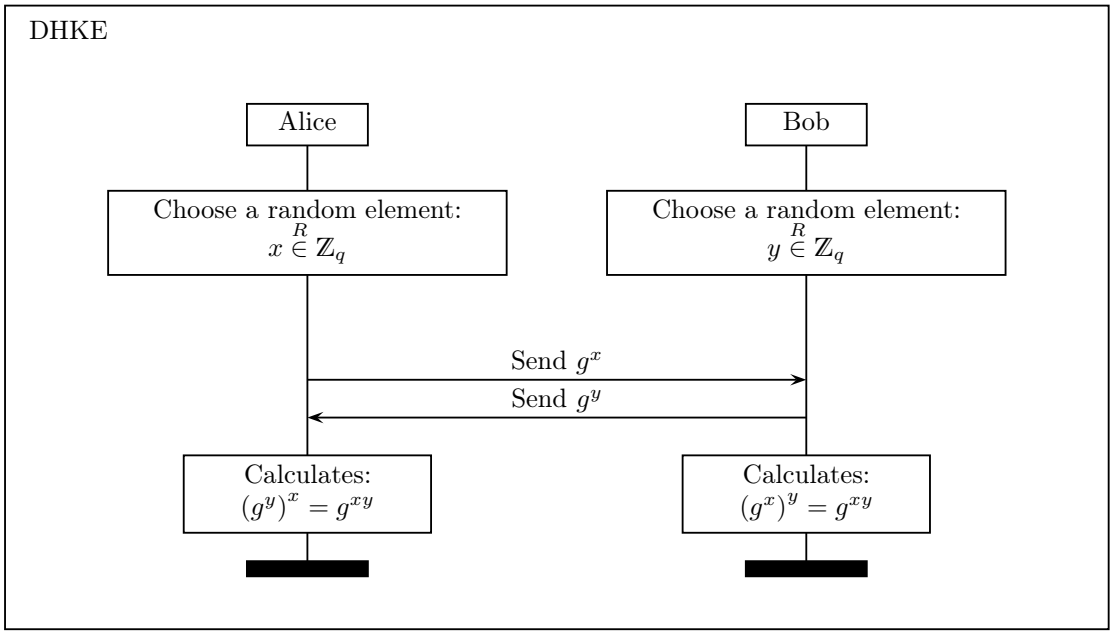
3.3 Back to Key Agreement

Recall that in the Diffie-Hellman key agreement we had algorithm: $\text{GroupGen}\left(1^k\right) \to (\mathbb{G}, g, q)$ which returned a group $\mathbb{G}$, a generator and the order q.

Definition 3.3.1 (The DDH Assumption (for GroupGen)) We generate a group $(\mathbb{G}, g, q) = \text{GroupGen}\left(1^k\right)$ and sample: $x, y, u \overset{R}{\in} \mathbb{Z}_q$ then:

$$(g, g^x, g^y, g^{xy}) \cong (g, g^x, g^y, g^u)$$

Recall that we had the key agreement protocol:



Now alice and bob can use g^{xy} as their key. The DDH assumption means that they could ignore the g^{xy} and simply agree on g^u in some miraculous way (telepathy) and the adversary won't be able to distinguish between the two cases. That gives us the security for the protocol.

Chapter 4

ElGamal Encryption

4.1 Definition

The ElGamal encryption was published around 8 years after Diffie-Hellman published their paper.

Definition 4.1.1 (ElGamal Encryptin) :

- KeyGen $\left(1^k\right)$:
  $(\mathbb{G}, g, q) = \text{GroupGen}\left(1^k\right)$
  Generate: $s \overset{R}{\in} \mathbb{Z}_q$
  And set sk $= s$ and pk $= (g^s, (\mathbb{G}, g, q))$ (we are going to take the definition of the group $(\mathbb{G}, g, q)$ as implicit, and not going to mention them anymore).

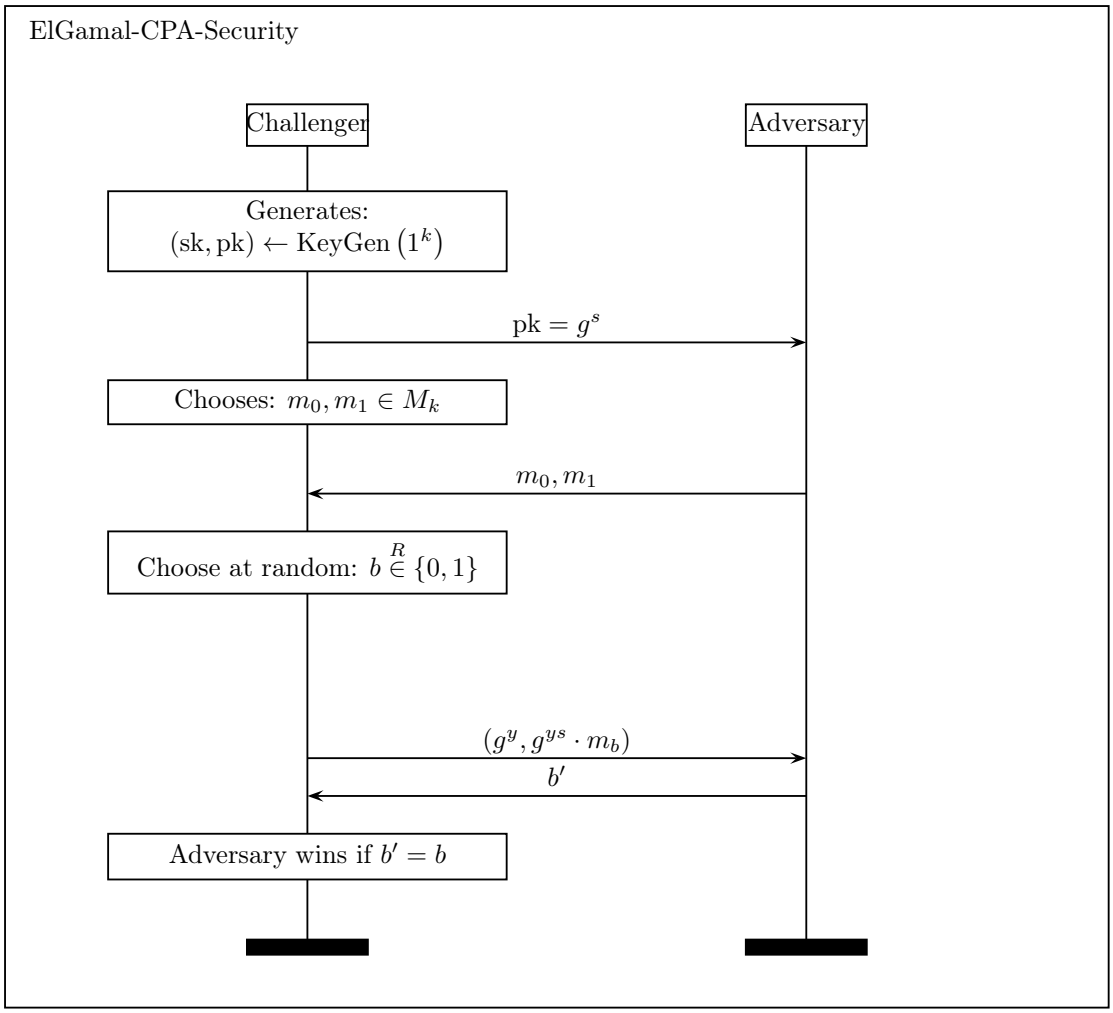
- $\mathrm{Enc}_{\mathrm{pk}}(m)$:
  Parse $\mathrm{pk} = g^s$. Generate $y \overset{R}{\in} \mathbb{Z}_q$ and then: $(g^y, (g^s)^y \cdot m)$.

  Remarks 4.1.2 The message space of this scheme is: $M = \mathbb{G}$.
  But that is a little bit weird. Because we want to encrypt for example the message "Hello World!", so we can reduce the message space to be $M' = \{1, g\}$ (meaning g^0, g^1), so we can encrypt 0 and 1.
  But if we encrypt bit by bit, we are choosing different y for every bit! Meaning, we run the entire Enc algorithm for every bit.

- $\mathrm{Dec}_{sk}(c)$:
  We have:$c = (g^y, g^w)$ and the secret key $\mathrm{sk} = s$ so we can calculate:
  $$g^w \cdot (g^y)^{-s} = g^{w-sy}$$

Correctness: Remember that correctness means that $\mathrm{Dec}_{\mathrm{sk}}(\mathrm{Enc}_{\mathrm{pk}}(m)) = m$.
Note that if we properly encrypt a message then: $c = (g^y, g^w) = (g^y, g^{sy}m)$. So if we decrypt we get: $g^{w-ys} = g^{sy} \cdot m \cdot g^{-ys} = m$.

Security: We want to show CPA-security. We've seen that it is enough to consider the one message CPA-game. We will prove security by reduction to the hardness of the DDH assumption.
Let A be a PPT that tries to break CPA-security for the encryption scheme with advantage $\varepsilon(k)$.



In order to prove security, we are going to use again hybrids. We are defining:
$$H_i - \mathrm{Adv}[A](k) = \Pr_{H_i}[A \text{ wins}] - 1/2$$

Hybrid H_0: Challenger acts the same as the CPA seen above.
Note that:

$$\text{CPA} - \text{Adv}\left[A\right](k) = \left|H_0 - \text{Adv}\left[A\right](k)\right|$$

Hybrid H_1: Instead of $c = (g^s, g^{sy} m_b)$ send $c = (g^y, g^u m_b)$ where $u \overset{R}{\in} R$.

Claim 4.1.3

There exists an adversary B s.t. $\text{DDH} - \text{Adv}\left[B\right](k) \geq \frac{1}{2}\left|H_1 - \text{Adv}\left[A\right](k) - H_0 - \text{Adv}\left[A\right](k)\right|$.

Proof: Define $B^A\left(1^k, g, g^x, g^y, g^z\right)$ as follows:

1. Running A on $\text{pk} = g^x$, A sends $m_0, m_1 \in \mathbb{G}$.
2. Choose $b \overset{R}{\in} \{0,1\}$ and compute: $c = (g^y, g^z m_b)$.
3. Send c to A, obtain b' from A.
4. If $b = b'$ then output 1 otherwise, output 0.

$$\begin{aligned}
\text{DDH} - \text{Adv}\left[B\right](j) &= \frac{1}{2}\left|\Pr\left[B\left(1^k, g^x, g^y, g^{xy}\right) = 1\right] - \Pr\left[B\left(1^k, g, g^x, g^y, g^u\right) = 1\right]\right| \\
&= \frac{1}{2}\left|\Pr_{H_0}\left[A \text{ wins}\right] - \Pr_{H_1}\left[A \text{ wins}\right]\right|
\end{aligned}$$

■

Hybrid H_2: Instead of $c = (g^y, g^u m_b)$ use: $c = (g^y, g^u)$.

Claim 4.1.4

$\left|H_2 - \text{Adv}\left[A\right](k) - H_1 - \text{Adv}\left[A\right](k)\right| \leq 0$.

This claims follows from:

Claim 4.1.5

$H_2 - \text{Adv}\left[A\right](k) = 0$.

Proof: This is because the bit b is not even present in c, so the adversary can only guess it with probability $1/2$. ■

What we have is the following claim:

Claim 4.1.6

For every PPT adversary A, exists a PPT B such that: $\text{CPA} - \text{Adv}\left[A\right](k) \leq 2\text{DDH} - \text{Adv}\left[B\right](k)$. *Thus, if DDH holds ⇒ ElGamal is CPA-Secure.*

Chapter 5

Matrices

5.1 DDH in matrix form

Note that DDH means: $\begin{pmatrix} g & g^x \\ g^y & g^{xy} \end{pmatrix} \cong \begin{pmatrix} g & g^x \\ g^y & g^u \end{pmatrix}$.

Or in different notation:

$$g^{\begin{pmatrix} 1 & x \\ y & xy \end{pmatrix}} \cong g^{\begin{pmatrix} 1 & x \\ y & u \end{pmatrix}}$$

Note that the rand of the first matrix $\begin{pmatrix} 1 & x \\ y & xy \end{pmatrix}$ is 1. While the second one is 2 (except rarely, probability $1/q$).

Claim 5.1.1

Assuming DDH, If $A \overset{R}{\in} \mathrm{Rk}_1\left(\mathbb{Z}_q^{2\times 2}\right)$; $B \overset{R}{\in} \mathrm{Rk}_2\left(\mathbb{Z}_q^{2\times 2}\right)$ *then* $g^A \cong g^B$.

31/03/2015

Remarks 5.1.2 The notation $\mathrm{Rk}_i\left(\mathbb{Z}_q^{2\times 2}\right)$ means a 2×2 matrix of rank i over a the ring $\mathbb{Z}_q$.
Note that the DDH matrices are of a specific form, why is it enough that we cannot distinguish between them means that we cannot distinguish between rank 1 and rank 2 matrices?

Claim 5.1.3

Let $A \in \mathbb{Z}_1^{n\times m}$; $L \overset{R}{\in} \mathrm{Rk}_n\left(\mathbb{Z}_q^{n\times n}\right)$; $R \overset{R}{\in} \mathrm{Rk}_m\left(\mathbb{Z}^{m\times m}\right)$. *Then:* $B = L\cdot A\cdot R$ *is uniform in* $\mathrm{Rk}_s\left(\mathbb{Z}_q^{n\times m}\right)$ *with* $s = \mathrm{Rk}(A)$.

Let D be a the distribution of matrices $\begin{pmatrix} 1 & x \\ y & xy \end{pmatrix}$ and let U be the distribution $\begin{pmatrix} 1 & x \\ y & u \end{pmatrix}$. We are getting g^X where X is from either D or U.
We will generate $L, R \overset{R}{\in} \mathrm{Rk}_2\left(\mathbb{Z}_q^{2\times 2}\right)$ and calculate: $g^{L\cdot X\cdot R}$ then we have that: $g^{L\cdot D\cdot R}$ is uniform in Rk_1 but $g^{L\cdot U\cdot R}$ is (almost) uniform in Rk_2.

Claim 5.1.4

Let $A \overset{R}{\in} \mathrm{Rk}_{d_1}\left(\mathbb{Z}_q^{n\times m}\right)$, $B \overset{R}{\in} \mathrm{Rk}_{d_2}\left(\mathbb{Z}_q^{n\times m}\right)$ *with* $1 \le d_1, d_2 \le \min(n, m)$ *then* $g^A \cong g^B$ *assuming DDH.*

Remarks 5.1.5 Note that the case of $d_1 = 1$ and $d_2 = 2$ is a direct result form the last claim just choose $L = \mathrm{Rk}_2\left(\mathbb{Z}_q^{2\times n}\right)$ and $R = \mathrm{Rk}_2\left(\mathbb{Z}_q^{m\times 2}\right)$. And the same proof will work.

Claim 5.1.6

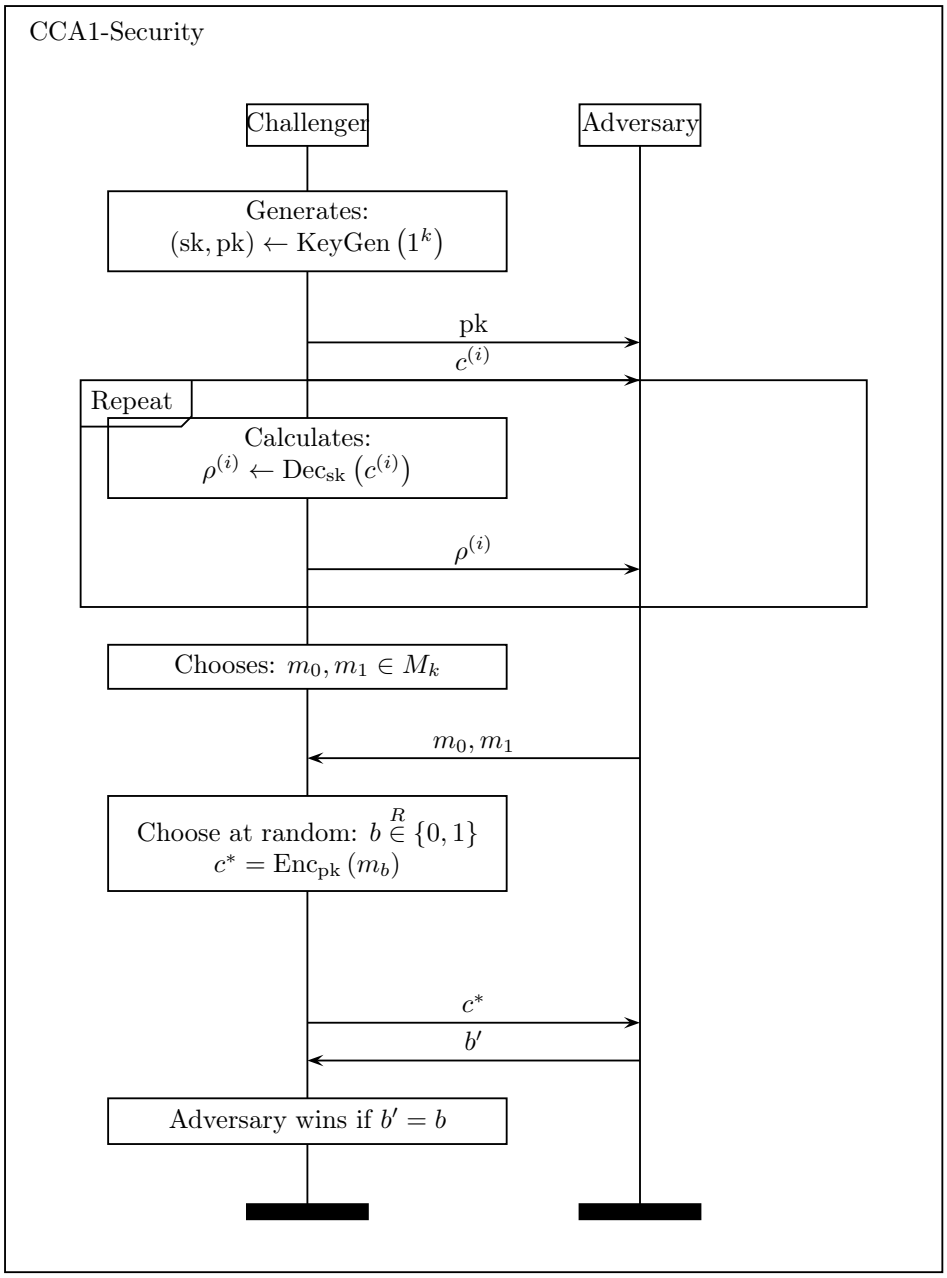
Let $A \in \mathbb{Z}_q^{n\times m} \Rightarrow A$ *is full rank w.p.* $\ge 1 - \frac{1}{(q-1)}$.

Chapter 6

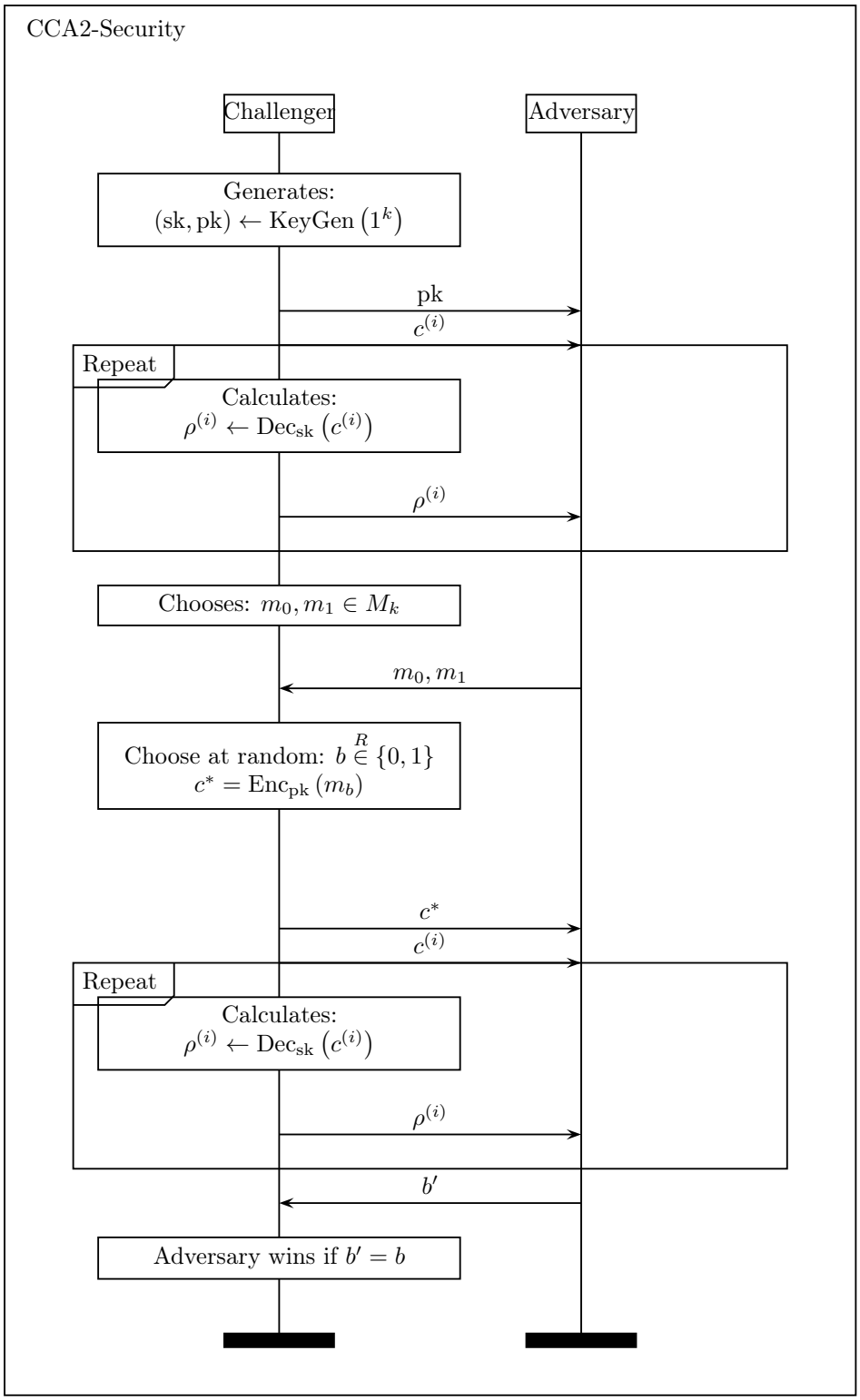
CCA Security

6.1 Introduction

Chosen Ciphertext Attach(CCA) is allowing the adversary to interact with the decryption algorithm, not only the encryption (like in CPA). We allow the adversary to encrypt messages he selects and run the decryption on them. It doesn't seem like a great help, but maybe if the adversary creates illegal messages he can learn from them something.



There is also a second notion, called CCA2 where we allowed to query the decryption also after receiving c^* (without allowing to query c^*).



✎ **Example 6.1.1 ()** ElGamal is not CCA2 secure.
Recall that $\mathrm{sk} = s$; $\mathrm{pk} = g^s$. $\mathrm{Enc}\,(m) = (g^r, g^{r \cdot s} \cdot m)$.
Assume that the adversary got from the challenger: $c^* = (g^r, g^w)$. The adversary can calculate: $c' = (c^r, g^w \cdot g)$ in the post-challenge queries phase. And then the challenger will answer because $c' \neq c^*$ and we will get:

$$g^w \cdot g \cdot g^{-rs} = g^{w-rs} \cdot g = m_b \cdot g$$

What we did here, called malleability, we got one message and transformed it to another message. This seems like a bad feature, but it is actually very useful in homomorphic encryption for example.

6.2 Cramer-Shoup (lite) - CCA1

The scheme works as follows:

- KeyGen $\left(1^k\right)$: Choose $0 \neq \underline{a} \overset{R}{\in} \mathbb{Z}_q^2$; $\underline{s}, \underline{t} \overset{R}{\in} \mathbb{Z}_q^2$. And calculate:

$$\underline{y}^T = \underline{a}^T \left[\underline{s} \| \underline{t}\right]$$

  The secret key is: $\mathrm{sk} = [\underline{s}\|\underline{t}]$ and the public key is: $pk = \left(g^{\underline{a}^T}, g^{\underline{y}^T}\right)$.

- $\mathrm{Enc}_{\mathrm{pk}}(m)$: Choose $r \overset{R}{\in} \mathbb{Z}_q^m$ and calculate:

$$c = \left(g^{r\underline{a}^T}, g^{r \cdot \underline{y}^T}\left[m\|1\right]\right)$$

  Remarks 6.2.1 It is not a matrix product, between $g^{r\cdot\underline{y}^T}$ and $[m\|1]$. It is an element wise product. $g^A \cdot g^B = g^{A+B}$.

- $\mathrm{Dec}_{\mathrm{sk}}\left(g^{\underline{b}^T}, g^{\underline{z}^T}\right)$: We calculate:$g^{\underline{z}^T - \underline{b}^T[\underline{s}\|\underline{t}]}$ and check whether it looks like $[m\|1]$. If so, output m otherwise output $\perp$.

Note that if we are trying to decrypt legal cipher text, then what we learn on the decryption is $\underline{a}^T \cdot \underline{t}$ and not t completely.

What is the test that is done in the decryption? We are checking whether: $g^{z_2 - \underline{b}^T \cdot \underline{t}} = g^0$. Meaning $\iff$ $z_2 = \underline{b}^T \cdot \underline{t}$ (we know $\underline{a}^T \cdot \underline{t} = y_2$).

Note that if $\underline{b}^T \neq r\underline{a}^T$, then: $\Pr_{\underline{t}}\left[z_2 = \underline{b}^T \cdot \underline{t}\right] = 1/q$. Note that even if we know y_2 we cannot get all the information regarding $\underline{t}$. And the same thing holds for $\underline{s}$.

Claim 6.2.2

CS-Lite is CCA1 secure under DDH.

Proof: We define:

$$\Pr_{H_i}\left[A \text{ wins}\right] - \frac{1}{2} = H_i - \mathrm{Adv}\left[A\right]$$

Hybrid 0: Normal CCA1 security game.

$$\left|H_0 - \mathrm{Adv}\left[A\right]\right| = \mathrm{CCA1} - \mathrm{Adv}\left[A\right]$$

$$\begin{aligned} c^* &= \left(g^{\underline{a}^T \cdot r}, g^{\underline{y}^T \cdot r}\left[m\|1\right]\right) \\ &= \left(g^{\underline{a}^T \cdot r}, g^{\underline{a}^T[\underline{s}\|\underline{t}] \cdot r}\left[m\|1\right]\right) \left(\underline{b^*} = r\underline{a}\right) \\ &= \left(g^{\underline{b^*}^T}, g^{\underline{b^*}^T[\underline{s}\|\underline{t}]}\left[m\|1\right]\right) \end{aligned}$$

Note that all that needed in order to simulate the hybrid H_0 is: $g^{\underline{a}^T}$, $[\underline{s}\|\underline{t}]$, $g^{\underline{b^*}^T}$.

Hybrid 1: Choose: $\begin{bmatrix} \underline{a}^T \\ \underline{b^*}^T \end{bmatrix} \overset{R}{\in} \mathrm{Rk}_2\left(\mathbb{Z}_q^{2\times 2}\right)$.

Claim 6.2.3

$\left|H_1 - \mathrm{Adv}\left[A\right] - H_0 - \mathrm{Adv}\left[A\right]\right| < 2\mathrm{Rank} - \mathrm{Adv}\left[B\right]$.

Note that: we have $g^{\begin{bmatrix} \underline{a}^T \\ \underline{b^*}^T \end{bmatrix}}$ and we want to distinguish between rank 1 and full rank, and we know that it is hard to distinguish between them in the exponent. Meaning that there is something that the adversary cannot do, but given the hardness of DDH, we can do that to him.

Hybrid 2: In the decryption queries, we are getting $\left(g^{\underline{b}^T}, g^{\underline{z}^T}\right)$. We check if: $\underline{b}^T = r\underline{a}^T$ for some r (we assume here that the challenger is not computational efficient, so if we want to prove something, it has to be statistical argument). If no output $\perp$ if yes, proceed as before.

In fact we are limiting the adversary to valid ciphertexts.

Claim 6.2.4

$|H_2\text{−Adv}\,[A] - H_1\text{−Adv}\,[A]| \leq \frac{t}{q}$ *(where t is the number of queries made by the adversary).*

Given $\underline{a}^T$, $\underline{y}^T$ and we get $\left(\underline{b}^T, \underline{z}^T\right)$, we ask $\underline{b}^T \cdot \underline{t} = z_2$ and:

$$\Pr_{\{\underline{t}:\underline{a}^T \cdot \underline{t} = y_2\}} \left[\underline{b}^T \cdot \underline{t} = z_2\right] = 1/q$$

Claim 6.2.5

$H_2\text{−Adv}\,[A] = 0$.

■ 02/04/2015

Chapter 7

Leftover Hash Lemma

7.1 Introduction

Assume we have a random variable X with arbitrary distribution function. We want to apply some function on X and get another random variable which is uniform distributed.

Definition 7.1.1 (Min-Entropy) The **Min-Entropy** of a random variable X is:

$$H_\infty(X) = -\log\left(\max_x \Pr[X = x]\right)$$

High Min-Entropy means that we have high randomness in the random variable that we can extract.
We also want to define a 2-Universal hash functions:

Definition 7.1.2 (2-Universal) Consider a distribution $\mathcal{H} \subseteq (\mathcal{S} \to \mathcal{T})$ (functions from $\mathcal{S} \to \mathcal{T}$). $\mathcal{H}$ is 2-universal if:

$$\forall x, y \in \mathcal{S},\ x \neq y \quad \Pr_{h \overset{R}{\in} \mathcal{H}}[h(x) = h(y)] = 1/|\mathcal{T}|$$

Now we can define the Leftover Hash Lemma:
Lemma 7.1.3 (*Leftover Hash Lemma (LHL)*)
Let $\varepsilon > 0$, $\mathcal{H}$ is 2-universal, X is a random variable supported on $\mathcal{S}$ s.t. $H_\infty(X) \geq \log|T| + 2\log(1/\varepsilon)$, then: for every adversary A (even computationally unbounded) it maintains:

$$\left| \Pr_{\substack{h \overset{R}{\in} \mathcal{H} \\ x \sim X}}[A(h, h(x)) = 1] - \Pr_{\substack{h \overset{R}{\in} \mathcal{H} \\ u \overset{R}{\in} T}}[A(h, u) = 1] \right| \leq \varepsilon$$

Remarks 7.1.4 A can know the distribution of X, it doesn't matter.

The left over hash lemma means that a 2-universal is a good randomness extractor.
This is all information theoretic, we haven't talk about any computational efficiency nor asymptotic analysis.

7.2 Construction of a 2-universal

How can we get such family of 2-universal functions?

Let's define a family of hash functions, this is an important example.
For all $\underline{a} \in \mathbb{Z}_q^n$ define $h_{\underline{a}} : \{0,1\}^n \to \mathbb{Z}_q$ as follows:

$$h_{\underline{a}}(x) = \langle \underline{a}, \underline{x} \rangle$$

We define $\mathcal{H}$ to be uniform over $h_{\underline{a}}$ (meaning that we are picking $\underline{a} \in \mathbb{Z}_q^n$ uniformly, and returning $h_{\underline{a}}$).
Now, let $x \neq y \in \{0,1\}^n$, we have:

$$\Pr_{\underline{a} \overset{R}{\in} \mathbb{Z}_q^n} \left[\langle \underline{a}, \underline{x} \rangle = \langle \underline{a}, \underline{y} \rangle \right] = \Pr_{\underline{a} \overset{R}{\in} \mathbb{Z}_q} \left[\langle \underline{a}, \underline{x} - \underline{y} \rangle = 0 \right]$$

We know that $x \neq y$ there is at least one coordinate, denoted by i^*, where: $x_{i^*} \neq y_{i^*}$. Hence:

$$= \Pr \left[\underbrace{a_{i^*}(x_{i^*} - y_{i^*})}_{\text{uniform}} + \sum_{i \neq i^*} a_i (x_i - y_i) = 0 \right] = 1/q$$

Thus, if we have a random variable X s.t.: $H_\infty(X) \geq \log q + 2\log(1/\varepsilon)$ (and if $1/\varepsilon$ is negligible) then:

$$(\underline{a}, \langle \underline{a}, \underline{x} \rangle) \equiv (\underline{a}, u)$$

7.3 Leftover hash lemma with side information

What if someone gives us some information about X, can we still apply the leftover hash lemma? The answer is yes:

Lemma 7.3.1 (*LHL with side information*)

Let $\mathcal{H}$ be 2-universal, and let X s.t. $H_\infty(X) \geq \log|T| + 2\log(1/\varepsilon) + \lambda$ then: for every $L(x;r) \to \{0,1\}^\lambda$ for every adversary A (even computationally unbounded) it maintains:

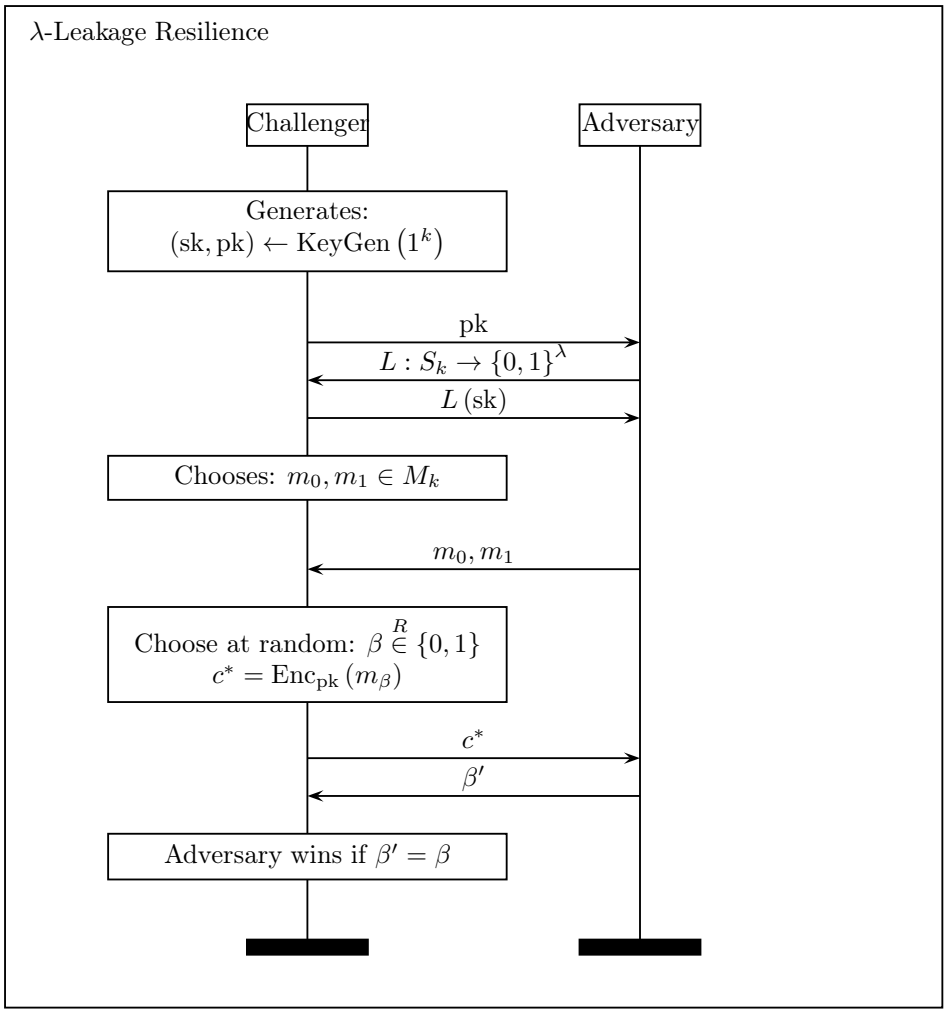
$$|\Pr[A(h, h(x), r, L(x;r)) = 1] - \Pr[A(h, u, r, L(x;r) = 1)]| \leq \varepsilon$$

Chapter 8

Leakage Resilient Encryption

8.1 Definition

We are now allowing the adversary to leak λ bits calculated on the secret key. We want to define λ-Leakage Resilience:



We do assume that L is polynomial time.

8.2 Existence of λ-Leakage-Resilient

Lemma 8.2.1

$\forall \lambda\left(k\right) = \mathrm{poly}\left(k\right)$ there exists a scheme E_λ s.t. E_λ is λ-Leakage-Resilient under DDH.

Proof: We define the scheme as follows:

- $\mathrm{KeyGen}\left(1^k\right)$: Choose: $\underline{s} \overset{R}{\in} \{0,1\}^n$ with $n = [2\log q + 2k + \lambda\left(k\right)]$. Choose $g^{\underline{a}} \in \mathbb{G}^n$, and set: $g^y = g^{\langle \underline{a}, \underline{s} \rangle}$. Set $\mathrm{sk} = \underline{s}$ and $\mathrm{pk} = (g^{\underline{a}}, g^y)$.
- $\mathrm{Enc}_{\mathrm{pk}}\left(m\right)$: The message space is $m \in \mathbb{G}$. We have: $\mathrm{pk} = (g^{\underline{a}}, g^y)$. We choose $r \overset{R}{\in} \mathbb{Z}_q$ and return: $c = (g^{r\underline{a}}, g^{r \cdot y} \cdot m)$.
- $\mathrm{Dec}_{\mathrm{sk}}\left(g^{\underline{b}}, g^z\right)$: We have: $\mathrm{sk} = \underline{s}$ so we calculate:

$$g^z \cdot g^{-\langle \underline{b}, \underline{s} \rangle}$$

Note that a cipher text in this scheme looks like:

$$\begin{aligned} c &= (g^{r\underline{a}}, g^{r \cdot y} \cdot m) \\ &= \left(g^{r\underline{a}}, g^{\langle r\underline{a}, \underline{s} \rangle} \cdot m\right) \\ &= \left(g^{\underline{b}}, g^{\langle \underline{b}, \underline{s} \rangle} \cdot m\right) \end{aligned}$$

For $\underline{b} = r\underline{a}$. Note that for any value of $\underline{b}$, the cypher text will still be decrypt-able even if $\underline{b} \neq r\underline{a}$.
Let A be a λ-Leakage Adversary, and we are going to define hybrids:

Hybrid H_0: The standard leakage game:

$$\text{Leakage} - \text{Adv}\left[A\right] = \left|\Pr_{H_0}\left[A \text{ wins}\right] - 1/2\right|$$

Hybrid H_1: We now set:

$$c^* = \left(g^{\underline{b}}, g^{\langle \underline{b}, \underline{s} \rangle} m_\beta\right)$$

For $g^{\underline{b}} \overset{R}{\in} \mathbb{G}^n$.
Note that if the adversary can distinguish between H_1 and H_0 he can break DDH, as he can distinguish between $(g^{\underline{a}}, g^{r\underline{a}})$ and $\left(g^{\underline{a}}, g^{\underline{b}}\right)$ for uniform $\underline{b}$.
Meaning that there exists an adversary B s.t.:

$$\text{Rank} - \text{Adv}\left[B\right] = \left|\Pr_{H_0}\left[A \text{ wins}\right] - \Pr_{H_1}\left[A_{\text{wins}}\right]\right|$$

How does $B\left(1^k, g^M\right)$ work?

1. Parse g^M as $\left[g^{\underline{a}} \| g^{\underline{b}}\right]$.
2. Sample $\underline{s} \in \{0,1\}^n$.
3. Run A on $\text{pk} = \left(g^{\underline{a}}, g^{\langle \underline{a}, \underline{s} \rangle}\right)$.
4. A sends L, sends back $L\left(\underline{s}\right)$.
5. Get m_0, m_1, sample β and compute $c^* = \left(g^{\underline{b}}, g^{\langle \underline{b}, \underline{s} \rangle} m_\beta\right)$.
6. Get β' and return 1 if $\beta' = \beta$.

Note that the probability that B return 1 in the low rank case is $\Pr_{H_0}\left[A_{\text{wins}}\right]$, while in the full rank case it is (close to): $\Pr_{H_1}\left[A \text{ wins}\right]$. So if A can win H_0 it can also win H_1 (otherwise it distinguish between low and full rank).
But note that $g^{\langle \underline{b}, \underline{s} \rangle}$, according to the LHL, is looks uniform! How can we use it? We are going to define yet another hybrid:

Hybrid H_2: Now, we change: $c^* = \left(g^{\underline{b}}, g^u \cdot m_\beta\right)$ for uniform $u \overset{R}{\in} \mathbb{Z}_q$. Note that in H_2 the probability of winning is exactly $1/2$. It remains to show that:

$$\left|\Pr_{H_1}\left[A \text{ wins}\right] - \Pr_{H_2}\left[A \text{ wins}\right]\right| \leq 2^{-k}$$

Which of course is negligible in the security parameter. Note that from the LHL, for every adversary C (even computationally unbounded):

$$\left|\Pr\left[C\left(\underline{b}, \left\langle \underline{b}, \underline{s} \right\rangle, \underline{a}, \tilde{L}\left(\underline{s}; \underline{a}\right)\right) = 1\right] - \Pr\left[C\left(\underline{b}, u, \underline{a}, \tilde{L}\left(\underline{s}; \underline{a}\right)\right) = 1\right]\right| \leq 2^{-k}$$

With $\tilde{L}\left(\underline{s}; \underline{a}\right) = \left(\left\langle \underline{a}, \underline{s} \right\rangle, L\left(\underline{s}\right)\right)$.
We want to show an C s.t. it gives exactly $\Pr_{H_1}\left[A \text{ wins}\right]$ and $\Pr_{H_0}\left[A \text{ wins}\right]$. We define $C\left(\underline{b}, z, \underline{a}, y, \ell\right)$ as follows:

1. Simulate A to get m_0, m_1, sample β.
2. Send $c^* = \left(g^{\underline{b}}, g^z \cdot m_\beta\right)$.
3. Get β' and output 1 if $\beta = \beta'$

■ 14/04/2015

Chapter 9

Digital Signature Scheme

9.1 The Scenario

We again have the entities Alice and Bob, but now, we don't care about the secrecy of the content of the message, but we want to verify that the message that Alice sent to Bob was really written by Alice and not by a malicious entity.

A digital signature scheme is a cryptographic primitive that allows us to solve this problem.

Alice generate a signature keys, sk (Alice's secret key) and vk (the verification key). Alice will send bob m and a signature σ that is generated using the secret key, and Bob will be able to verify it using the verification key.

Definition 9.1.1 (Digital Signature Scheme) A Digital Signature Scheme is a tuple of three algorithms:

1. $\mathrm{Gen}\left(1^k\right) \to (\mathrm{sk}, \mathrm{vk})$.
2. $\mathrm{Sign}_{\mathrm{sk}}\left(1^k, m\right) \to \sigma$ (Message space $\{0,1\}^*$)
3. $\mathrm{Ver}_{\mathrm{vk}}\left(1^k, m, \sigma\right) \to 0/1$

Correctness:

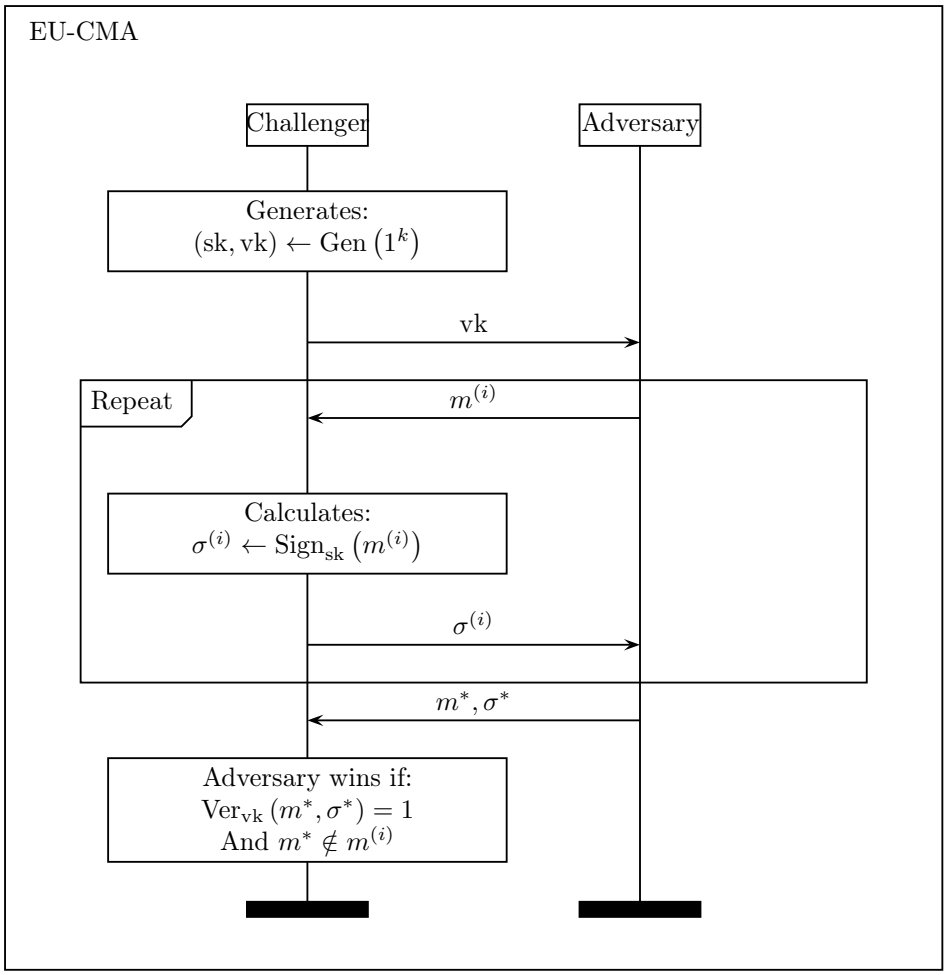
$$\forall m \quad \Pr_{(\mathrm{sk},\mathrm{vk})\leftarrow \mathrm{Gen}(1^k)}\left[\mathrm{Ver}_{\mathrm{vk}}\left(m, \mathrm{Sign}_{\mathrm{sk}}\left(m\right)\right) \neq 1\right] = \mathrm{negl}\left(k\right)$$

Remarks 9.1.2 Unlike encryption, we might have Sign and Ver being deterministic algorithms.

So, in our game, Alice is going to generate sk and vk and publish vk on her website for example. And then Alice will send a message to bob and sign it (meaning that she will actually send the tuple (m, σ) where σ is the output of $\mathrm{Sign}_{\mathrm{sk}}\left(1^k, m\right)$). Then Bob will verify the message using $\mathrm{Ver}_{\mathrm{vk}}\left(1^k, m, \sigma\right)$ where vk is the same verification Alice published in advanced.

9.2 Security Game

We are going to define security as follows:



We define the advantage:

$$\text{EU} - \text{CMA} - \text{Adv}\left[A\right] = \Pr\left[A \text{ wins}\right]$$

It turns out that we can construct a signature scheme from only OWF (unlike public key encryption).

9.3 Bilinear Maps

Definition 9.3.1 A bilinear map is a function $e : \mathbb{G} \times \mathbb{G} \to \mathbb{H}$ ($\mathbb{G}$ is a group of order q and $\mathbb{H}$ is a group) with the properties:

1. Bilinear:
$$\forall x, y \in \mathbb{Z}_q \quad e\left(g^x, g^y\right) = e\left(g, g^{xy}\right) = e\left(g, g\right)^{xy}$$
2. Non-degenerate:
$$e\left(g, g\right) \neq 1_{\mathbb{H}}$$

We are interested in groups that have bilinear maps, but we still want groups that Discrete Log is still hard on them.

So what properties can we expect from $\mathbb{G}$?

- DL is hard (Note that if DL is easy on $\mathbb{H}$ then you can also solve DL on $\mathbb{G}$ by simply looking at $e\left(g, g^x\right) = e\left(g, g\right)^x$). We can, and want to have that property.
- DDH - Is $(g, g^x, g^y, g^{xy}) \cong (g, g^x, g^y, g^u)$? Note that we cannot expect to have DDH on such groups as we can check:
$$e\left(g^x, g^y\right) = e\left(g, g^{xy}\right)$$

But note that with noticeable probability:

$$e\left(g^x, g^y\right) = e\left(g, g\right)^{xy} \neq e\left(g, g\right)^u = e\left(g, g^u\right)$$

- CDH - Given (g, g^x, g^y) can we come up with g^{xy}? We want groups such that CDH will be hard. Meaning that we will be able to calculate the inverse of e (described in the next bullet).
- Inverse: $e\left(g, g\right)^x \to g^x$. We want groups such that the inverse is hard. So CDH won't break on $\mathbb{G}$.
- BDDH (Bilinear DDH): Is $\left(g, g^x, g^y, g^z, e\left(g, g\right)^{xyz}\right) \cong \left(g, g^x, g^y, g^z, e\left(g, g\right)^u\right)$
- Rank: $g^A \cong g^B$: $A \in \mathrm{Rk}_2\left(\mathbb{Z}_q^{n \times m}\right)$, $B \in \mathrm{Rk}_{>2}\left(\mathbb{Z}_q^{n \times m}\right)$.

What is most likely to be hard by decreasing order:

1. DL on target group.
2. Inverse.
3. CDH (Note that if you solve CDH you can solve BDDH).
4. BDDH/Rank.

It is not known which one is harder BDDH or RANK.

9.4 Construction of Digital Signature Scheme from Bilinear Maps

A very high level idea:

- The message is going to be g^μ
- Keys: $\mathrm{sk} = s,\ \mathrm{vk} = g^s$.
- Sign: $g^\mu \to g^{\mu s}$.
- Verification: $e\left(g^\mu, g^s\right) = e\left(g, g\right)^{\mu s} = e\left(g, g^{\mu s}\right)$

Note that in this way, if the adversary actually know μ than he can choose μ^*, and calculate g^{μ^*} and then raise the $\mathrm{vk} = g^s$ to μ^* and get $g^{s\mu^*}$. How can we solve this? Well we are going to cheat a little bit, and use random oracles.

Definition 9.4.1 (Random Oracle) A random function $\mathcal{O} : \{0,1\}^* \to \mathbb{G}$, that everyone in the world has access to. For the sake of formality, we need to have the security parameter, so actually the function is of the form: $\{0,1\}^* \times \mathbb{N} \to \bigcup_k \mathbb{G}_k$.

The construction:

- $\mathrm{Gen}^{\mathcal{O}}\left(1^k\right)$: $s \overset{R}{\in} \mathbb{Z}_q$. $\mathrm{sk} = s$; $\mathrm{vk} = g^s$.
- $\mathrm{Sign}^{\mathcal{O}}_{\mathrm{sk}}\left(m\right)$: $g^\mu = \mathcal{O}\left(m\right)$. Output: $\sigma = g^{\mu \cdot s}$ (Note that we do not have μ at any time, only g^μ).
- $\mathrm{Ver}^{\mathcal{O}}_{\mathrm{vk}}\left(m, \sigma\right)$: $\mathrm{vk} = g^s$, $\sigma = g^\eta$, $g^\mu = \mathcal{O}\left(m\right)$. Check:

$$e\left(g^\mu, g^s\right) \overset{?}{=} e\left(g, g^\eta\right)$$

Why is it secure? Now, the adversary can also query the random oracle. But the only thing he can do with it is sending queries and receiving the response. But we can simulate the random oracle in the challenger, just output random values for every new query and store the solution in a table. If we query the same message multiple times, returned the stored value.

So, let us prove security:

Let A be an adversary agains out scheme that runs in time $t\left(k\right)$. We will of course use hybrids!

Hybrid H_0: Generate s, g^s and send $\text{vk} = g^s$ to A. In order to handle A queries to $\mathcal{O}$, the challenger will maintain a list $\mathcal{L}$, whenever A queries $\mathcal{O}(x)$ the challenger will decide as follows:

- If $x \in \mathcal{L}$, reply same as before.
- Otherwise, $\mu_x \overset{R}{\in} \mathbb{Z}_q$ and add (x, μ_x, g^{μ_x}) to the list $\mathcal{L}$. Return g^{μ_x}.

Note that:

$$\Pr_{H_0}[A \text{ wins}] = \text{EU} - \text{CMA} - \text{Adv}[A]$$

Hybrid H_1: Change response to sign queries: $m^{(i)}$: find $\left(m^{(i)}, \mu_i, g^{\mu_i}\right)$ in the list, return $\sigma = (g^s)^{\mu_i}$.

$$\Pr_{H_1}[A \text{ wins}] = \Pr_{H_0} = [A \text{ wins}]$$

Note that now, the challenger doesn't need to know s, he can use g^s and raise it to μ_i.
We need to know what is the oracle query corresponding to μ^* but we don't really know which one is it.

Hybrid H_2: The challenger generates g^s and g^{μ^*} (the challenger doesn't know μ^* nor s). The challenger samples $i^* \overset{R}{\in} [t]$, given the ith oracle query x_i :

1. If $(x_i, \mu, g^{\mu}) \in \mathcal{L} \rightarrow$ As before.
2. If $i \neq i^* \rightarrow$ As before.
3. If $i = i^*$ and x_i is "fresh" return g^{μ^*}, insert to the list $\left(x_i, -, g^{\mu^*}\right)$ (we don't know the μ^*)

Note that now, we cannot sign for i^* because we don't know the signing key nor μ^*. So if the adversary submitted x_i for signing then he caught us. And we abort.

Claim 9.4.2

$\Pr_{H_2}\left[A \text{ wins} \wedge \mathcal{O}(m^*) = g^{\mu^*}\right] \geq 1/t \cdot \Pr_{H_1}[A \text{ wins}]$.

Note that it seems like a problem, because when he gets an abort, he learns something about i^*. But when the adversary ask for signing then he wouldn't forge that value according to the game, so he couldn't lie and say that he wanted to forge that value. **Proof:** Note that:

$$\begin{aligned}
\Pr_{H_1}[A \text{ wins}] &= \Pr_{H_1}[A \text{ wins} \wedge m^* \text{ was not submitted for signing}] \\
&= \sum_{i=1}^{t} \Pr_{H_1}\left[A \text{ wins} \wedge m^* \notin \left\{m^{(i)}\right\} \wedge \mathcal{O}(m^*) \text{ was the } i\text{th oracle query}\right] \\
&= \sum_{i=1}^{t} \frac{1}{t} \Pr_{H_2}\left[A \text{ wins} \wedge \mathcal{O}(m^*) = g^{\mu^*} \wedge \mathcal{O}(m^*) \text{ was the } i^*\text{th query}\right] \\
&= \frac{1}{t} \Pr_{H_2}\left[A \text{ wins} \wedge \mathcal{O}(m^*) = g^{\mu^*}\right]
\end{aligned}$$

■

But note, that if we can win in hybrid H_2 we can also solve the Computational Diffie Hellman problem: by letting $B^A\left(g, g^s, g^{\mu^*}\right)$ plays the challenger of H_2 with A and output σ^*.

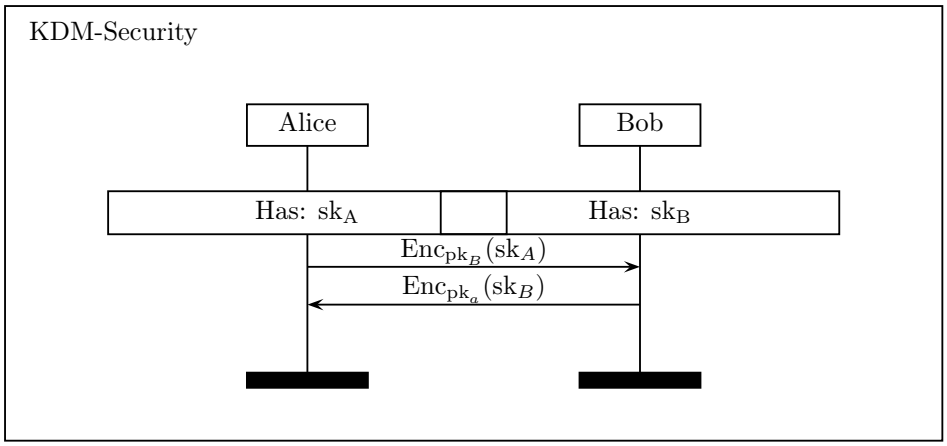
Chapter 10

KDM-Security

10.1 Introduction

We know have the scenario that Alice has a secret key sk_A associated with a public key pk_A and bob has sk_B associated with pk_B.

But they want to share the secret keys with each other:



It seems like it suppose to be secure, if the scheme is secure why would it be able to get information from encryption of the secret key?

A more typical idea is that alice has an hard-drive and it is encrypted with a public key, but also stores the secret key, encrypted.

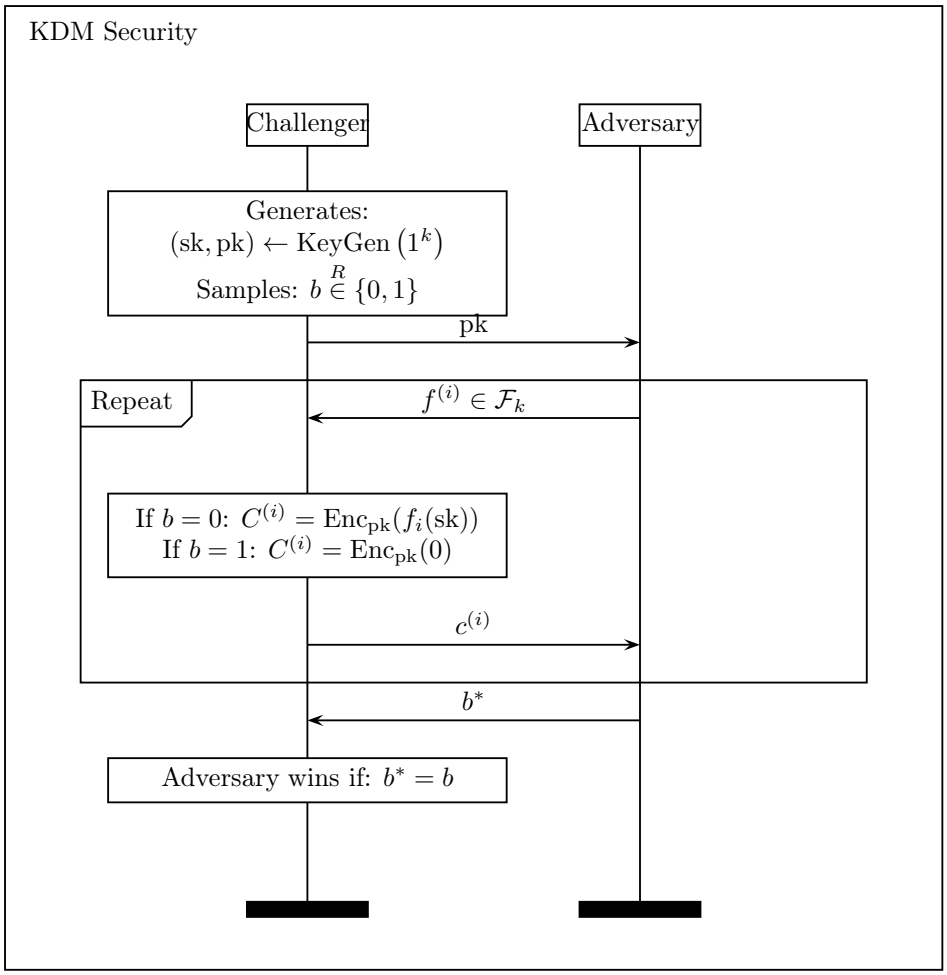
But apparently, CPA security does not imply KDM-Security immediately.

10.2 Definition

Let $E = (\mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Dec})$ be an encryption scheme. Let $\mathcal{S}_k$ be the sk space of E and $\mathcal{M}_k$ the message space.

Let $\mathcal{F} = \{\mathcal{F}_k\}_k$ where: $\mathcal{F}_k \subset \mathcal{S}_k \to \mathcal{M}_k$ (functions on theg secret key).

We say that an encryption is $(\mathcal{F}, 1)$-KDM secure if:



Remarks 10.2.1 We denote it $(\mathcal{F}, 1)$-KDM because there is only one encryption scheme, and in a more general case (for example the Alice Bob scenario from above), there might be more than one encryption scheme in use.

10.3 Construction

10.3.1 ElGamal

Let's start with ElGamal, Recall that we have the $\mathrm{sk} = s$ and $\mathrm{pk} = g^s$ and: $\mathrm{Enc}\left(g^\mu\right) = \left(g^r,\ g^{rs} \cdot g^\mu\right)$.
Now assume that we have the fuctnio class:

$$\mathcal{F} = \left\{ f_{g^a, g^b} = g^{as+b} \right\}_{g^a, g^b}$$

We can prove that ElGamal is secure with respect to $\mathcal{F}$. We won't give a proof but note that:

$$\begin{aligned} \mathrm{Enc}\left(f_{g^a,g^b}(s)\right) &= \left(g^r, g^{rs} g^{as+b}\right) \\ &= \left(g^r, g^{(r+a)\cdot s} \cdot g^b\right) \end{aligned}$$

Denote $r' = r + a$ and we get:

$$\mathrm{Enc}\left(f_{g^a,g^b}(s)\right) = \left(g^{r'-a}, g^{r's} g^b\right)$$

So, encryption of a linear function of the secret key, seems as above.
Note that now we have:

$$\left(g, g^s, g^a, g^b, \left(g^{r'-a}, g^{r's} \cdot g^b\right)\right)$$

And from DDH:

$$\left(g, g^s, g^a, g^b, \left(g^{r'-a}, g^{r's} \cdot g^b\right)\right) \cong \left(g, g^s, g^a, g^b, \left(g^{r'-a}, g^r \cdot g^b\right)\right)$$

So ElGamal is secure against $\mathcal{F}$-KDM attack.

10.3.2 Construction

The construction is going to be similar to the leakage resilient:

- KeyGen $\left(1^k\right)$:
  $\underline{s} \overset{R}{\in} \{0,1\}^n$; $n = \log q + 2k$
  $g^{\underline{a}} \overset{R}{\in} \mathbb{G}^n$
  $g^y = g^{\langle \underline{a}, \underline{s} \rangle}$.
  Set sk $= \underline{s}$ and pk $= (g^{\underline{a}}, g^y)$.
- $\mathrm{Enc}_{\mathrm{pk}}(m)$:
  $m = g^\mu$ and pk $= (g^{\underline{a}}, g^y)$. We sample $r \overset{R}{\in} \mathbb{Z}_q$ and calculate:
  $$c = (g^{r\underline{a}}, g^{r \cdot y} g^\mu)$$
- $\mathrm{Dec}_{\mathrm{sk}}(c)$:
  We have sk $= \underline{s}$ and $c = \left(g^{\underline{b}}, g^z\right)$. We calculate: $g^z \cdot g^{-\langle \underline{b}, \underline{s} \rangle}$.

The function class that we are going to talk about is:

$$\mathcal{F} = \left\{ f_{g^{\underline{\alpha}}, g^\beta}(\underline{s}) = g^{\langle \underline{a}, \underline{s} \rangle + \beta} \right\}_{g^{\underline{\alpha}}, g^\beta}$$

We want to prove that this scheme is secure against $\mathcal{F}$-KDM attacks.
Let A be $(\mathcal{F}, 1)$-KDM adversary against the scheme, A runs in time t.
We define the following hybrids:

Hybrid H_0: The $(\mathcal{F}, 1)$-KDM game.
$g^{\underline{\alpha}^{(i)}}, g^{\beta^{(i)}}$. Assume for now that $b = 0$ then:

$$\begin{aligned} c^{(i)} &= \left(g^{r^{(i)} \cdot \underline{a}}, g^{r^{(i)} \cdot y} \cdot g^{\left\langle \underline{\alpha}^{(i)}, \underline{s} \right\rangle} g^{\beta^{(i)}}\right) \\ &= \left(g^{r^{(i)} \cdot \underline{a}}, g^{r^{(i)} \cdot \langle \underline{a}, \underline{s} \rangle} \cdot g^{\left\langle \underline{\alpha}^{(i)}, \underline{s} \right\rangle} g^{\beta^{(i)}}\right) \end{aligned}$$

Denote: $\underline{b}^{(i)} = r^{(i)} \underline{a}$ and we have:

$$c^{(i)} = \left(g^{\underline{b}^{(i)}}, g^{\left\langle \underline{b}^{(i)}, \underline{s} \right\rangle} \cdot g^{\left\langle \underline{\alpha}^{(i)}, \underline{s} \right\rangle} \cdot g^{\beta^{(i)}}\right)$$

Hybrid H_1: Now, we choose: $g^{\underline{b}^{(i)}} \in \mathbb{G}^n$. In hybrid H_0 we had: $g^{\begin{bmatrix} \underline{a}^T \\ r^1 \underline{a}^T \\ \vdots \\ r^t \underline{a}^T \end{bmatrix}}$ and now we have $g^{\begin{bmatrix} \underline{a}^T \\ \underline{b}^1 \\ \vdots \\ \underline{b}^t \end{bmatrix}}$, note that the first one is a matrix of rank 1, while the second is with high probability full rank. And we know that:

$$g^{\begin{bmatrix} \underline{a}^T \\ r^1 \underline{a}^T \\ \vdots \\ r^t \underline{a}^T \end{bmatrix}} \cong g^{\begin{bmatrix} \underline{a}^T \\ \underline{b}^1 \\ \vdots \\ \underline{b}^t \end{bmatrix}}$$

Hybrid H_2: We have:

$$\begin{aligned} c^{(i)} &= \left(g^{\underline{b}^{(i)}}, g^{\langle \underline{b}^{(i)} + \underline{\alpha}^{(i)}, \underline{s} \rangle}, g^{\beta^{(i)}}\right) \\ &= \left(g^{\underline{b}'^{(i)} - \underline{\alpha}^{(i)}}, g^{\langle \underline{b}'^{(i)}, \underline{s} \rangle} \cdot g^{\beta^{(i)}}\right) \end{aligned}$$

Where $\underline{b}'^{(i)} = \underline{b}^{(i)} + \underline{\alpha}^{(i)}$.

Up until now, we have not change a thing. The difference in H_2 is that we going to replace: $g^{\underline{b}'^{(i)}} = g^{r^{(i)} \cdot \underline{a}}$ (it doesn't matter if it is the same $r^{(i)}$ from H_0 but for simplicity consider fresh randomness).

And now:

$$\begin{aligned} c^{(i)} &= \left(g^{r(i)\underline{a} - \underline{\alpha}^{(i)}}, g^{r^{(i)} \langle \underline{a}, \underline{s} \rangle} \cdot g^{\beta^{(i)}}\right) \\ &= \left(g^{r(i)\underline{a} - \underline{\alpha}^{(i)}}, g^{r^{(i)} y} \cdot g^{\beta^{(i)}}\right) \end{aligned}$$

Now, in the view of the adversary, we have $\langle \underline{a}, \underline{s} \rangle$ in every cipher text, but it alway $\langle \underline{a}, \underline{s} \rangle = y$.

Hybrid H_3: Instead of $y = \langle \underline{a}, \underline{s} \rangle$ (which is not uniform, but statistically close to uniform) we generate $y \overset{R}{\in} \mathbb{Z}_q$ uniformly. So, now the public key is $(g^{\underline{a}}, g^y)$ and the cipher text looks like:

$$\left(g^{r^{(i)}\underline{a} - \underline{\alpha}^{(i)}}, g^{r^{(i)} y} \cdot g^{\beta^{(i)}}\right)$$

We are almost done.

We are now going to denote: $\underline{d} = \begin{bmatrix} \underline{a} \\ y \end{bmatrix}$, we define $\mathrm{pk} = g^{\underline{d}}$ and now:

$$\begin{aligned} c^{(i)} &= g^{r^{(i)}\underline{d}} \cdot g^{\begin{bmatrix} -\underline{\alpha}^{(i)} \\ \beta^{(i)} \end{bmatrix}} \\ &= g^{\underline{d}'^{(i)}} \cdot g^{\begin{bmatrix} -\underline{\alpha}^{(i)} \\ \beta^{(i)} \end{bmatrix}} \end{aligned}$$

Where: $\underline{d}'^{(i)} = r^{(i)}\underline{d}$.

Hybrid H_4: We are now changing: $g^{\underline{d}'^{(i)}} \overset{R}{\in} \mathbb{G}^{n+1}$. And now $c^{(i)}$ will be uniform, and we cannot win with probability better than $1/2$.

Chapter 11

Quadratic Residue Assumption

11.1 Number theory backgraound

First, we need a short overview of some required number theory:
Consider $\mathbb{Z}_p$ as the set of all the integers modulo p, for some prime number p. We denote by $\mathbb{Z}_p^*$ the multiplicative group in $\mathbb{Z}_p$.
Some facts regarding $\mathbb{Z}_p^*$:

- $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\}$.
- $\left|\mathbb{Z}_p^*\right| = p - 1$.

We now want to define a sub-group of $\mathbb{Z}_p^*$ called the quadratic residues group:

Definition 11.1.1 For any prime number p, we define the **quadratic residues group** modulo p as:

$$\mathrm{QR}_p = \left\{x^2 \mid x \in \mathbb{Z}_p^*\right\}$$

Some facts regarding QR_p:

- $\left|\mathrm{QR}_p\right| = \frac{p-1}{2}$.
- If $p \equiv 3 \pmod 4$ then $-1 \notin \mathrm{QR}_p$.

Definition 11.1.2 For any prime number p and a natural number x the **Legendre symbol** $\left(\frac{x}{p}\right)$ is defined as follows:

$$\left(\frac{x}{p}\right) = \begin{cases} 1 & x \in \mathrm{QR}_p \\ -1 & x \in \mathbb{Z}_p^* \setminus \mathrm{QR}_p \\ 0 & x \notin \mathbb{Z}_p \end{cases}$$

Some facts regarding the Legender symbol:

- $\left(\frac{x}{p}\right) = x^{\frac{p-1}{2}}$.
- $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$.
- $\left(\frac{a^{-1}}{p}\right) = \left(\frac{a}{p}\right)$.

In a similar way, consider $\mathbb{Z}_N$ for an integer N. Mostly we will consider the case in which N is a **Blum Integer**.

Definition 11.1.3 We call an integer N a **Blum Integer** if $N = pq$ and $p \equiv q \equiv 3 \pmod 4$.

Some facts regarding Blum Integers:

- $\mathbb{Z}_N^* = \{x \in \mathbb{Z}_N \mid \ p \nmid x \wedge q \nmid x\}$.
- $|\mathbb{Z}_n^*| = \varphi(N) = (p-1)(q-1)$, where φ is the Euler totient function.

Definition 11.1.4 For any integer N, we define the quadratic residues group modulo N as:

$$\mathrm{QR}_N = \left\{x^2 \mid \ x \in \mathbb{Z}_N^*\right\}$$

Some facts regarding QR_N for a Blum Integer N:

- $(-1) \notin \mathrm{QR}_N$.
- $|\mathrm{QR}_N| = \frac{|\mathbb{Z}_N^*|}{4} = \frac{(p-1)(q-1)}{4}$.

Definition 11.1.5 Let x, N be integers and let $N = \prod_{i=1}^{k} (p_i)^{\alpha_i}$ be the prime factorization of N. We define the **Jacobi symbol** $\left(\frac{x}{N}\right)$ as follows:

$$\left(\frac{x}{N}\right) = \prod_{i=1}^{k} \left(\frac{x}{p_i}\right)^{\alpha_i}$$

where $\left(\frac{x}{p_i}\right)$ for $i = 1, \ldots, k$ are Legendre symbols.

Some facts regarding Jacobi symbols:

- $\left(\frac{x}{N}\right)$ can be efficiently computed. That is, there exists an algorithm that on input (x, N) outputs $\left(\frac{x}{N}\right)$ and runs in polynomial time in the length of the bit representation of (x, N).
- If $x \in \mathrm{QR}_N$ then $\left(\frac{x}{N}\right) = 1$.
- If N is a Blum Integer then $\left(\frac{-1}{N}\right) = 1$.

Let N be an integer. We define the multiplicative group J_N as:

$$J_N = \left\{x \in \mathbb{Z}_N^* \mid \ \left(\frac{x}{N}\right) = 1\right\}$$

If N is a Blum Integer, then:

$$J_N = \left\{(-1)^b x^2 \mid \ b \in \{0,1\}, \ x \in \mathbb{Z}_N^*\right\}$$

11.2 The Assumption

Definition 11.2.1 Let p, q be random k-bit primes. Let N be a Blum Integer, $r \overset{R}{\in} \mathbb{Z}_N^*, \ s \overset{R}{\in} J_N$.
The **QR Problem** (for Blum Integers) is distinguishing $\left(N, r^2\right)$ from (N, s).

Definition 11.2.2 The **QR Assumption** is :

$$\left(N, r^2\right) \cong (N, s)$$

where $\cong$ means computational indistinguishability.

We usually use the QR assumption with respect to Blum Integers.

11.3 PKE from the QR assumption

11.3.1 Goldwasser-Micali crypto-system

We define the Goldwasser-Micali (GM) crypto-system as follows:

- KeyGen $\left(1^k\right)$: Sample p, q two random k-bit Blum Primes (i.e. $p \equiv q \equiv 4 \pmod 4$). And denote $N = pq$. Output: $\text{pk} = N$, $\text{sk} = (p, q)$.
- The message space is $\mathcal{M} = \{0, 1\}$.
- $\text{Enc}_{\text{pk}}(m)$: Sample $x \overset{R}{\in} \mathbb{Z}_N^*$ (recall that $\text{pk} = N$), and output: $c = (-1)^m x^2$.
- $\text{Dec}_{\text{sk}}(c)$: Use p, q to check if $c \in \text{QR}_N$. If it is, output 0, otherwise output 1.

We can prove the security of the GM crypto-system based on the QR assumption.

11.3.2 Cocks crypto-system

We can also define another crypto-system, due to Cocks:

- KeyGen $\left(1^k\right)$: Sample N, a k-bit Blum Integer. Sample $r \overset{R}{\in} \mathbb{Z}_N^*$, and denote $R = r^2$. Output $\text{pk} = (N, R)$, $\text{sk} = r$ (We can also think of N as some publicly agreed integer prior to running of KeyGen).
- The message space is $\mathcal{M} = \{0, 1\}$.
- $\text{Enc}_{\text{pk}}(m)$: Sample $t \overset{R}{\in} \mathbb{Z}_N^*$ and output $c = \left(\frac{R}{t} + t, \left(\frac{t}{N}\right)(-1)^m\right)$, where $\left(\frac{t}{N}\right)$ is the Jacobi symbol (We denote: $f_R(t) = \frac{R}{t} + t$).
- $\text{Dec}_{\text{sk}}(c)$: Denote $c = \left(y, (-1)^b\right)$. Output m such that:

$$(-1)^m = (-1)^b \left(\frac{y + 2r}{N}\right)$$

Theorem 11.3.1

The Cocks crypto-system is correct, and is secure assuming the QR assumption.

Proof: Correctness:

$$\begin{aligned}\mathrm{Dec}_{\mathrm{sk}}\left(\left(\frac{R}{t}+t,\left(\frac{t}{N}\right)(-1)^m\right)\right) &= \left(\frac{t}{N}\right)(-1)^m\left(\frac{\frac{R}{t}+t+2r}{N}\right)\\ &= \left(\frac{t}{N}\right)(-1)^m\left(\frac{t^{-1}}{N}\right)\left(\frac{R+2rt+t^2}{N}\right)\\ &= (-1)^m\left(\frac{(r+t)^2}{N}\right)=(-1)^m\end{aligned}$$

Security:

The proof idea is to consider an Hybrid in which R is chosen such that $R \in J_N \backslash \mathrm{QR}_N$, and note that in this hybrid the cyphertext is independent of m. ■

11.4 Identity Based Encryption

11.4.1 Definition

Suppose we wish to encrypt messages to many parties. Using PKE we have to run KeyGen for every party that wishes to receive messages. We now consider an alternative called Identity Based Encryption (IBE).

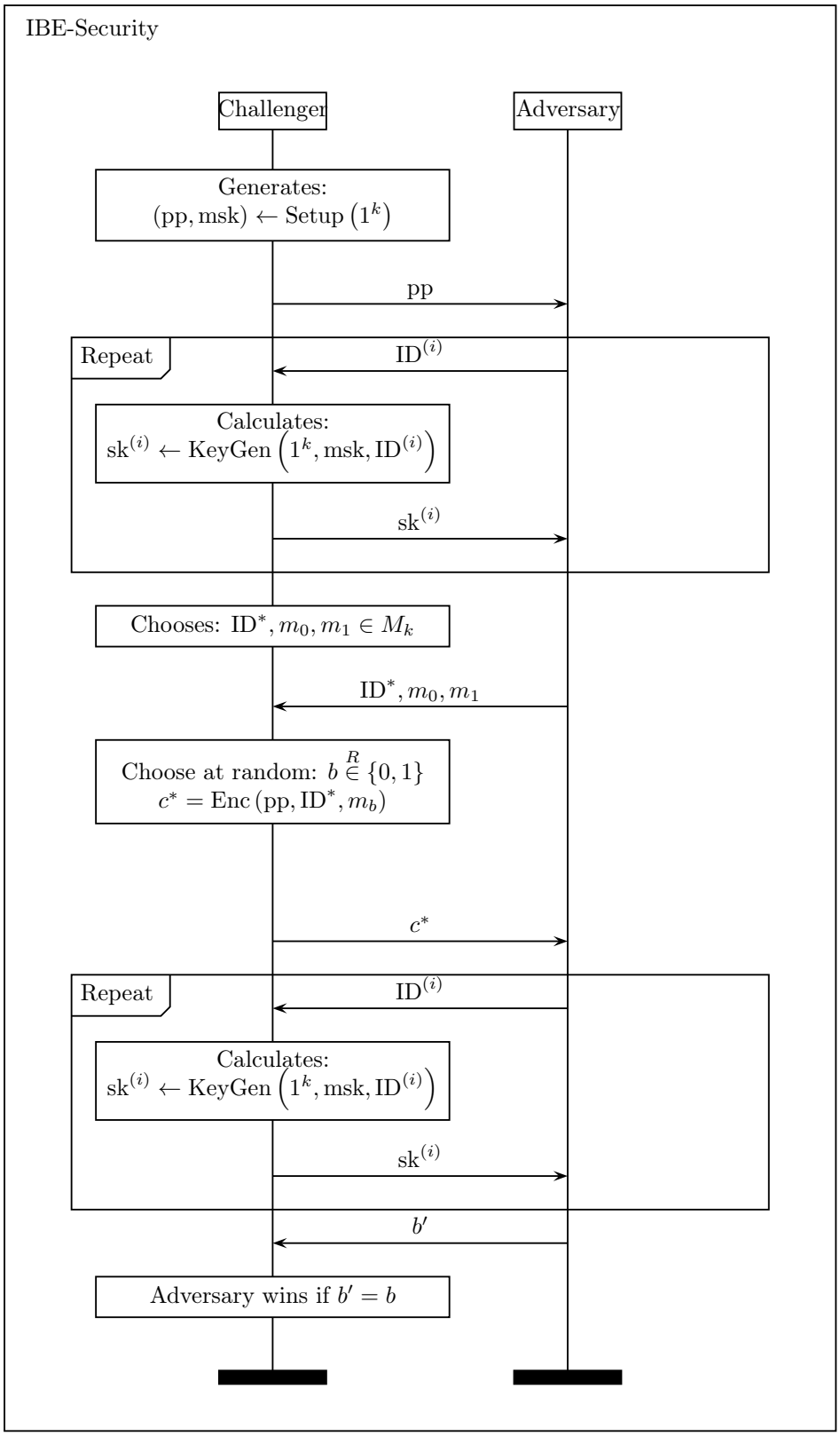
Definition 11.4.1 We assume each party has a unique identifier in $\{0,1\}^*$. An IBE is defined using four algorithms: (Setup, KeyGen, Enc, Dec):

- $\mathrm{Setup}\left(1^k\right) \to (\mathrm{pp}, \mathrm{msk})$. pp is an abbreviation of "public parameters", and msk is an abbreviation of "master secret key".
- $\mathrm{KeyGen}\left(1^k, \mathrm{msk}, \mathrm{ID}\right) \to \mathrm{sk}_{\mathrm{ID}}$.
- $\mathrm{Enc}\left(1^k, \mathrm{pp}, \mathrm{ID}, m\right) \to c$.
- $\mathrm{Dec}\left(1^k, \mathrm{sk}_{\mathrm{ID}}, c\right) \to m$.

Correctness is defined in the usual manner.

11.4.2 Security

Let (Setup, KeyGen, Enc, Dec) be an IBE. We define the security of the IBE using the following security game between the challenger and an adversary:



We say that adversary A wins the game when $b = b'$ and ID^* is not one of the IDs sent before or after the challenge. We define the advantage as:

$$\text{IBE} - \text{Adv}\left[A\right] = \left|\Pr\left[A \text{ wins}\right] - \frac{1}{2}\right|$$

We say that an IBE is secure if for every PPT adversary A, $\text{IBE} - \text{Adv}\left[A\right]$ is a negligible function in k. 28/04/2015

11.4.3 Construction from QR assumption

Consider the Cocks crypto-system, we are going to tweak it so it will construct an IBE crypto-system.

The public parameters are going to be: $\text{pp} = N$ and $\text{msk} = (p, q)$. Note that using the master-secret-key, we can evaluate the square root of an element (We check if it is a square modulo p and modulo q, if it is then it is also square modulo N).

We would like to have a random oracle: $\mathcal{O} : \{0,1\}^* \to \text{QR}_N$. But we don't really know how (even heuristic) to satisfy the operation of $\mathcal{O}$. Meaning that we don't know how to generate element QR_N without knowing the square root. But one thing we can do is to have $\mathcal{O} : \{0,1\}^* \to J_N$. But this is not good enough for the Cocks encryption scheme. So how can we solve it?

We know that in the case of Blum Integer, that every number with Jacobi Symbol 1 is of the form $(-1)^b x^2$. We are going to use this fact.

So, how is this crypto-system going to work?

- $\text{Setup}\left(1^k\right)$: $(p, q, N) \leftarrow \text{BlumGen}\left(1^k\right)$. Return: $\text{pp} = N$; $\text{msk} = (p, q)$.
- $\text{KeyGen}\left(\text{msk}, \text{ID}\right)$: $\text{msk} = (p, q)$, calculate: $R = \mathcal{O}\left(\text{ID}\right)$. If $R \in \text{QR}_N$ then output $r = \sqrt{R}$, otherwise output $r = \sqrt{-R}$.
- $\text{Enc}^{\mathcal{O}}\left(\text{pp}, \text{ID}, m\right)$: $\text{pp} = N$, $R = \mathcal{O}\left(\text{ID}\right)$. Choose $t_1, t_2 \overset{R}{\in} \mathbb{Z}_N^*$ and return:

$$c = \left(f_R\left(t_1\right), \left(\frac{t_1}{N}\right)(-1)^m, f_{-R}\left(t_2\right), \left(\frac{t_2}{N}\right)(-1)^m\right)$$

- $\text{Dec}^{\mathcal{O}}\left(\text{ID}, \text{sk}_{\text{ID}}, \left(t_1, (-1)^{\alpha_1}, t_2, (-1)^{\alpha_2}\right)\right)$: $\text{sk} = r$. If $r^2 = R$ then $(-1)^{\alpha_1}\left(\frac{y_1+2r}{N}\right)$ otherwise: $r^2 = -R$ and then: $(-1)^{\alpha_1}\left(\frac{y_2+2r}{N}\right)$.

11.4.4 Security proof

Again, we are going to build hybrids:

Hybrid H_0:

Normal IBE game, only that $\mathcal{O}$ is simulated using a list. The entries are going to look something like: $\left(\text{ID}, r, b, (-1)^b r^2\right)$. On every query, if the ID is in the list we are going to return the same value as before, if not we are going to choose $r \overset{R}{\in} \mathbb{Z}_N^*$ and $b \overset{R}{\in} \{0,1\}$ and generate a new item in the list and return it.

Hybrid H_1:

Never use (p, q), just use the list in order to know the square root.

Hybrid H_2:

There is going to be a special R^*, we are going to guess a query number and put R^* as the answer.

We want to argue here that:

$$\Pr\left[A \text{ wins } \wedge\ \mathcal{O}\left(\text{ID}^*\right) = R^*\right] \geq \square$$

(something not negligible, divided by the running time of the adversary).

Hybrid H_3:

$$c^* = \left(f_{R^*}\left(t_1\right), (-1)^{\alpha_1}, f_{-R^*}\left(t_2\right), (-1)^{\alpha_2}\right)$$

for $\alpha_1, \alpha_2 \overset{R}{\in} \{0,1\}$.

11.5 Yet another PKE from QR

(By Shafi Goldwasser and Zvika Brakerski).

Consider the following crypto-system: Let N be a Blum Integer (we don't need to know the factorization).

- KeyGen $\left(1^k\right)$: $\underline{s} \overset{R}{\in} \{0,1\}^n$. $g_1, \ldots, g_n \overset{R}{\in} \mathrm{QR}_N$. $g_0 = \prod g_i^{s_i}$. Return: sk $= \underline{s}$ and pk $= (g_0, g_1, \ldots, g_n)$. Where $n = \log N + 2k$ (going to be used in the leftover hash lemma).

- $\mathrm{Enc}_{\mathrm{pk}}(m)$: Choose $r \overset{R}{\in} \left[N \cdot 2^k\right]$ and take: $c = \left(g_1^r, \ldots, g_n^r, g_0^r (-1)^m\right)$.

Remarks 11.5.1 Recall that $|\mathrm{QR}_N| = \frac{\varphi(N)}{4} = \frac{(q-1)(p-1)}{4}$. In particular $N \cdot 2^k \gg |\mathrm{QR}_N|$.

- $\mathrm{Dec}_{\mathrm{sk}}(h_1, \ldots, h_n, h_0)$: calculate:

$$h_0 \cdot \prod h_i^{-s_i} = (-1)^m$$

We want to show security. We define a family of hash function:

$$H_{(g_1,\ldots,g_n)}(s_1, \ldots, s_n) = \prod g_i^{s_i}$$

This is a two universal has function. And we are able to use the leftover hash lemma.

The proof is done with hybrids:

Hybrid H_0:

The normal CPA game.

Hybrid H_1:

Replace $g_0 \overset{R}{\in} \mathrm{QR}_N$ using the leftover hash lemma.

Hybrid H_2:

pk $= (g_1, \ldots, g_m, -g_0)$.

If this is the public key, then the cipher text looks like:

$$c = \left(g_1^r, \ldots, g_n^r, (-1)^r g_0^r (-1)^m\right)$$

Note that: $g_i^r = g_i^{r(\mod \varphi(N)/4)}$. On the other hand: $(-1)^r = (-1)^{r(\mod 2)}$.

One thing to notice is that: $\gcd\left(\frac{\varphi(N)}{4}, 2\right) = 1$ because N is a Blum Integer then $p \equiv q \equiv 3 \pmod 4$.

Claim 11.5.2

Let $M = s \cdot t$ and $\gcd(s,t) = 1$ then if $r \overset{R}{\in} [M]$ then $(r \pmod s, r \pmod t)$ is uniform in $[s] \times [t]$.

(Basically this is the Chinese reminder theorem).

In the bottom line $(-1)^r$ completely randomize the message, so we don't have information in the ciphertext about the original message.

11.6 Decisional Composite Residuosity (DCR, Paillier)

Let N be a Blum Integer. Now, we are going to do arithmetics $\left(\mod N^2\right)$.

$$\begin{aligned} \mathbb{Z}_{N^2}^* &= \{x \in \mathbb{Z}_{N^2} \mid p \nmid x,\ q \nmid x\} \\ |\mathbb{Z}_{N^2}^*| &= N \cdot \varphi(N) \end{aligned}$$

Quadratic Residues only have 2 cosets, But we want more than that. So we can consider:

$$\begin{aligned} \text{NR} &= \left\{X^N :\ x \in \mathbb{Z}_{N^2}^*\right\} \\ |\text{NR}| &= \varphi(N) \end{aligned}$$

The DCR (Paillier) assumption:

$\text{NR} \cong \mathbb{Z}_{N^2}^*$.
We have:

$$\begin{aligned} \text{QR}_{N^2} &= \left\{x^2 \left(\mod N^2\right) \mid x \in \mathbb{Z}_{N^2}^*\right\} \qquad \text{size: } N \cdot \varphi(N)/4 \\ \text{CR} &= \left\{x^{2N} \left(\mod N^2\right) \mid x \in \mathbb{Z}_{N^2}^*\right\} \qquad \text{size: } \varphi(N)/4 \end{aligned}$$

The DCR assumption implies: $\text{QR}_{N^2} \cong \text{CR}$.
So if we have $x \in \text{QR}_{N^2} \iff x = (1+N)^i r^{2N}$. $(1+N)^i \left(\mod N^2\right) = 1 + iN$.
Now assume that $N = pq$ is a Blum Integer such that: $p = 2p' + 1$ and $q = 2q' + 1$ and p', q' are also primes.
We now want to construct an encryption-system from DCR:

- KeyGen $\left(1^k\right)$: $N \leftarrow \text{BlumSafeGen}\left(1^k\right)$. Let g be a generator for CR. $s \overset{R}{\in} \left[N \cdot 2^k\right]$. $\text{sk} = s$ and $\text{pk} = (N, g, g^s)$.
- Message space: $\mathcal{M} = [N]$
- $\text{Enc}_{\text{pk}}(m)$: We have $\text{pk} = (N, g, g^s)$. We are encrypting: $(g^r, (g^s)^r (1+N)^m)$.

  Remarks 11.6.1 Note that $(g^s)^r (1+N)^m$ is of the form $(1+N)^i r^{2N}$ and hence it is a square. Meaning that it the Jacobi Symbol of this element is always 1.

- $\text{Dec}_{\text{sk}}(h_1, h_2)$: We have $\text{sk} = s$. We compute: $h_2 h_1^{-s} = (1 + m \cdot N)$ and we output m.

Chapter 12

Learning With Errors

12.1 Introduction

We are going to work with $\mathbb{Z}_q = \left(\mathbb{Z} \cap \left(-\frac{q}{2}, \frac{q}{2}\right]\right)$ and assume that q is a prime, but most of the stuff will also work for q which are not primes.

The question is: how hard is to solve a set of linear equations modulo q?

Meaning we are going to have $\underline{s} = \mathbb{Z}_q^n$ and we are going to generate: $\underline{a_i} \overset{R}{\in} \mathbb{Z}_q^n$ and we have: $\langle \underline{a_i}, \underline{s} \rangle = b_i$. We are going to an adversary a bunch of tuples $\left(\underline{a_i}, b_i\right)$.

How easy is to solve and find $\underline{s}$? Well, very. Simply use gauss elimination.

Assume we have m samples. We can consider it in matrix form we have: $\underline{s}^T A = \underline{b}^T$ (where A is an $n \times m$ matrix and the i-th column of A is $\underline{a_i}$). We simply use gauss elimination and find $\underline{s}$ (Because $\underline{a_i}$ are generated uniformly, then with high probability A is going to be full ranked).

We can consider the matrix:

$$\left[\begin{array}{c} A \\ \hline \underline{b}^T \end{array}\right]$$

Note that this is an $(n+1) \times m$ matrix, but now the rank is at most n because the last row is linearly dependent in the others.

So that is easy, what can we do?

We would like to ask, what happen if we add noise. Meaning that:

$$b_i = \langle \underline{a_i}, \underline{s} \rangle + e_i$$

Where e_i is the noise. It is clear the if e_i is large and uniform it hides the result completely, so it is not solvable. But we would like to consider small noise.

But what is small with respect to $\mathbb{Z}_q$?

Definition 12.1.1 The absolute value of $x \in \mathbb{Z}_q$ is $\min_{\{y \in \mathbb{Z} |\ y \equiv x (\mod q)\}} |y|$.

So now we can define what is small. But we want to ask, even if the noise is bounded. Is it going to be solvable?

Note that if the noise is $\frac{q}{2}$ we hide the message completely and there is no hope to find a solution, but what happen if it is smaller?

Claim 12.1.2

If $|e_i| \leq q/4$ and $m \geq n \log q + k$ then $\underline{s}$ is specified with probability $1 - 2^{-k}$.

Proof: Consider $\underline{t} \neq \underline{s}$. Now we have $\underline{a_i}$ we want to ask what is:$\langle \underline{a_i}, \underline{s} - \underline{t} \rangle$. Note that:

$$\Pr\left[\left|\langle \underline{a_i}, \underline{s} - \underline{t}\rangle\right| < q/4\right] \leq 1/2$$

But note that:

$$\begin{aligned} \langle \underline{a_i}, \underline{t}\rangle - b_i &= \langle \underline{a_i}, \underline{t}\rangle - \langle \underline{a_i}, \underline{s}\rangle - e_i \\ &= \langle \underline{a_i}, \underline{t} - \underline{s}\rangle - e_i \end{aligned}$$

Note that:

$$\forall \underline{t} \neq \underline{s} \quad \Pr\left[\underline{t} \text{ survives}\right] \leq 2^{-m} = q^{-n} \cdot 2^{-k}$$

■

Definition 12.1.3 A distribution $\chi \subseteq \mathbb{Z}$ is B-bounded if $\Pr\left[\chi \notin [-B, B]\right] = 0$.

12.2 The LWE (Learning With Errors) Problem

Definition 12.2.1 Let $q, n, m \in \mathbb{N}$. Consider a noise distribution $\chi \subseteq \mathbb{Z}$. For all $\underline{s} \in \mathbb{Z}_q^n$ define: $A \overset{R}{\in} \mathbb{Z}_q^{n \times m}$, $\underline{e} \subseteq \chi^m$, $\underline{b}^T = \underline{s}^T A + \underline{e}^T \pmod q$.
The worst case $\text{LWE}_{q,n,\chi,m}$ is the problem of finding $\underline{s}$ given $A, \underline{b}$ distributed as above.
The average case $\text{LWE}_{q,n,k,m}$ is the same with $\underline{s} \overset{R}{\in} \mathbb{Z}_q^n$.

Of course if we can solve the worst case we can solve the average case. But note that if we take a worst case instance $A, \underline{b}^T$ and $\underline{t} \overset{R}{\in} \mathbb{Z}_q^n$ and then add $\underline{t}^T A + \underline{b}$ and we randomized it again. If we are able to solve it, then by subtracting $\underline{t}$ we will get $\underline{s}$.

Definition 12.2.2 (Decisional version) The $\text{DLWE}_{q,n,\chi,m}$ problem is to distinguish $\left(A, b^T\right)$ from $\left(A, \underline{u}^T\right)$ where $\underline{u} \overset{R}{\in} \mathbb{Z}_q^m$ uniformly.

Meaning that the matrix:

$$\left[\begin{array}{c} A \\ \hline \underline{b}^T \end{array}\right]$$

is very closed to be low rank (information-theoretic-wise), but it is indistinguishable from a completely random matrix.

In many times, m is not going to be interesting and we are going to allow the adversary as many samples as he wants. And we denote: $\text{LWE}_{q,n,\chi}$, $\text{DLWE}_{q,n,\chi}$.

12.3 Decisional to Search

Of course the search problem seems harder, because if we can find $\underline{s}$ then we can check the decisional version. But in some cases they are equivalent. If we can solve the decisional problem we can also solve the search problem. We are not going to show the proof, but only the idea. We are going to see how to do that in time poly (n, q) (ideally we wanted poly $(n, \log q)$).

Assume that we have some A that solves $\text{DLWE}_{q,n,\chi}$. And assume that we can have as many samples as we want: $(\underline{a}, \langle \underline{a}, \underline{s}\rangle + e)$.

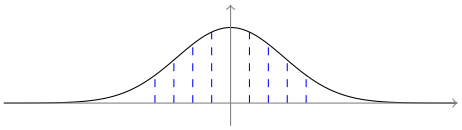
Check if $s_1 = z$:

Choose $\alpha \overset{R}{\in} \mathbb{Z}_q$ and check: $\underline{a}' = \underline{a} + (\alpha, 0, \ldots, 0)^T$. Note that $b' = b + \alpha z$. So if we are right, it is going to be a good LWE problem.
But if we are wrong if $s_1 \neq z$ this is going to rerandomize everything (because for each column that A wants we are going to pick random α which randomize everything).

12.4 Noise distribution

What χ should we use?

We want to use this problem for cryptography, so we want to ask what are good distribution to use.
Actually almost every distribution which is not made in particular to mess this up and that has enough entropy is going to be hard.
But what is the common distributions?
The distribution is the discrete gaussian:



We define the probability for a value of elements in $\mathbb{Z}_q$ is defined as:

$$D_{\alpha q,\mathbb{Z}}: \Pr\left[D_{\alpha q,\mathbb{Z}} = x\right] \propto e^{-\pi\left(\frac{x}{\alpha q}\right)^2}$$

Meaning we sample the value of the gaussian in the specific value in $\mathbb{Z}$ and not do a rounding and integrating.
We are going to define a "distribution" such that:

$$\Pr\left[D_{\alpha q,z} \notin \left[-\alpha q\sqrt{k}, \alpha q\sqrt{k}\right]\right] \leq 2^{-\Omega(k)}$$

It is not really a distribution because it is not summed up to 1.
$\mathrm{LWE}_{q,n,\alpha}$ is at least $2^{\tilde{\Omega}\left(n/\log(1/\alpha)\right)}$-hard so long as $\alpha q \geq O\left(\sqrt{n}\right)$.
If $q = 2^{n^\varepsilon}$ then it is hard as: $2^{n^{1-\varepsilon}/\mathrm{poly}\log(n)}$. If $q = 2^{n^{1-\varepsilon}}$ then 2^{n^ε}. 12/05/2015

12.5 Encryption Scheme From LWE

Recall that at the second class of the semester we've seen the El Gamal encryption scheme. We had a generator g and a secret key s and public key g^s and the cipher text looked like: $(g^r, g^{rs+\mu})$.
We want to ask: "How can we do the same things from LWE?".

$$\left[\begin{array}{c|c} A & \\ \hline \underline{b}^T & = s^T A + e^T \end{array}\right]$$

We now have A and $b^T = s^T A + e^T$. In the El Gamal case we had a low rank matrix in the exponent: $\begin{bmatrix} g & g^r \\ g^s & g^{rs+\mu} \end{bmatrix}$ and it looked like full rank because the hardness of DDH (note that it is low rank for $\mu = 0$ and full rank for $\mu = 1$, from DDH we couldn't distinguish).
In LWE we are going to have A as the public key, and we are going to sample $\underline{r}$ and calculate: $\underline{v} = A\underline{r}$ and the the last row, and we have:

$$\left[\begin{array}{c|c} A & \\ \hline \underline{b}^T & = \underline{s}^T A + \underline{e}^T \end{array}\right] \qquad \left[\begin{array}{c} \underline{v} \\ \hline \end{array}\right] = A \cdot \underline{r}$$

$$[w] = \underline{b}^T \cdot r / \underline{s}^T \cdot \underline{v}$$

Note that:

$$\begin{aligned} \underline{b}^T \cdot \underline{r} &= \left(\underline{s}^T A + \underline{e}^T\right) \underline{r} = \underline{s}^T A \underline{r} + \underline{e}^T \underline{r} \\ \underline{s}^T \underline{v} &= \underline{s}^T A \underline{r} \end{aligned}$$

We want: $\underline{e}^T \underline{r}$ to be small. In order to do so, we choose $r \overset{R}{\in} \{0,1\}^m$. So, if $\underline{e}$ comes from a B bounded distribution then we have:

$$\left|\underline{e}^T \underline{r}\right| \leq B \cdot m$$

So we want to choose the parameters such that:

$$\left|\underline{e}^T \underline{r}\right| \leq B \cdot m \ll q$$

12.5.1 Regev's Encryption Scheme

The values of n, B are going to be chosen according to the security parameter later on.

- KeyGen $\left(1^k\right)$: $A \overset{R}{\in} \mathbb{Z}_q^{n \times m}$ and $\underline{s} \overset{R}{\in} \mathbb{Z}_q^m$ and then we going to choose $\underline{e} \overset{R}{\in} \chi^m$ (B-bounded distribution) and $\underline{b}^T = \underline{s}^T A + \underline{e}^T$. We return: $\text{sk} = \underline{s}$, $\text{pk} = \left(A, \underline{b}^T\right)$.
- $\text{Enc}_{\text{pk}}(\mu)$: We generate $r \overset{R}{\in} \{0,1\}^m$, and calculate $\underline{v} = A \cdot \underline{r}$ and $w = \underline{b}^T \cdot \underline{r} + \mu \lceil q/2 \rfloor$. And we return: $\underline{c} = (\underline{v}, w)$.
- $\text{Dec}_{\text{sk}}(\underline{v}, w)$: Recall that $\text{sk} = \underline{s}$, we calculate: $|w - \underline{s}^t \cdot \underline{v}| < q/4$, If yes we return 0, otherwise we return 1.

Correctness:

Note that:

$$w - \underline{s}^t \cdot \underline{v} = \underline{b}^T \underline{r} + \mu \lceil q/2 \rfloor - \underline{s}^T \cdot \underline{v} = \underline{e}^T \cdot \underline{r} + \mu \lceil q/2 \rfloor$$

Now, If $\mu = 0$ then: $w - \underline{s}^t \cdot \underline{v} = \underline{e}^T \cdot \underline{r}$ and it is much smaller than: $q/4$ If the message $\mu = 1$ then we are going to be either very close to $q/2$ or $-q/2$, either way we are going to have large absolute value and get that the message is 1.
Note that this work with probability 1 in the case that $\left|\underline{e}^T \underline{r}\right| \leq B \cdot m \ll q$.
What is m? Again, we will show that later on.

Security:

Hybrid H_0: The CPA game.
We are sending $A, \underline{b}^T$ and $(\underline{v}, w)$.

Hybrid H_1: Choose $\underline{b}^T \overset{R}{\in} \mathbb{Z}_q^m$. Note that if adversary succeeds differently between H_0 and H_1 then he breaks LWE. Hence $H_0 \cong H_1$ due to the LWE assumption.
Now, for notation define:

$$\hat{A} = \left[\begin{array}{c} A \\ \hline \underline{b}^T \end{array}\right]$$

Now we can consider the ciphertext as:

$$c = \hat{A} \cdot \underline{r} + \underline{\mu}$$

where:

$$\underline{\mu} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \mu q/2 \end{pmatrix}$$

Note that $\left(\hat{A}, \hat{A}\underline{r}\right)$ is indistinguishable from uniform from the leftover hash lemma, that is:

$$\left(\hat{A}, \hat{A}\underline{r}\right) \equiv \left(\hat{A}, \underline{U}\right)$$

So, in order to actually use the LHL we need: $m = (n+1)\log q + 2k$. So we are ready for our next hybrid:

Hybrid H_2: Replace $\hat{A} \cdot \underline{r}$ with $\underline{U}$. From the LHL H_2 is indistinguishable from H_1. But now in H_2 the message is completely masked. Note that this is not CCA2-secure scheme, as we can simply add small noise to the w element, meaning that we get a different ciphertext for the same message.
Note that it is not CCA1-secure as well! We can choose $w = 0$ and then choose:

$$\underline{v} = \begin{pmatrix} q/2 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

And we will learn the cipher the first bit of the secret key, and we can repeat for every bit and learn the secret key completely.

12.5.2 Dual Regev

Note that in El Gamal we had some kind of symmetry we could tilt our head an consider g, g^r as the pk and $g^s, g^{rs+\mu}$ as the ciphertext. But in Regev's scheme it is a bit different

$$\left[\begin{array}{c|c} A & \\ \hline \underline{b}^T & = \underline{s}^T A + \underline{e}^T \end{array}\right] \qquad \left[\underline{v}\right] = A \cdot \underline{r}$$
$$\left[w\right] = \underline{b}^T \cdot r / \underline{s}^T \cdot \underline{v}$$

Now, we want to tilt our head in the case of Regev:

$$\left[\begin{array}{c} A \\ \hline \left[\underline{b}^T\right] \end{array}\right] \qquad \left[\underline{v}\right] = A \cdot \underline{r} \qquad = \text{pk}$$
$$\left[w\right] = \underline{b}^T \cdot r / \underline{s}^T \cdot \underline{v} \qquad = \text{sk}$$

- $\text{KeyGen}\left(1^k\right)$: $A \overset{R}{\in} \mathbb{Z}_q^{n \times m}$ and $\underline{r} \overset{R}{\in} \{0,1\}^m$, $\underline{v} = A\underline{r}$ $\text{pk} = (A, \underline{v})$ and $\text{sk} = \underline{r}$.
- $\text{Enc}_{\text{pk}}(\mu)$: Choose $\underline{s} \overset{R}{\in} \mathbb{Z}_q^m$. $\underline{e} \overset{R}{\in} \chi^m$, $\underline{b}^T = \underline{s}^T A + \underline{e}^T$. $w = \underline{s}^T \cdot \underline{v} + \mu\,(q/2)\,(+\eta)$. Return $c = \left(\underline{b}^T, w\right)$.
- $\text{Dec}_{\text{sk}}\left(\underline{b}^T, w\right)$: Check $\left|w - \underline{b}^T \cdot \underline{r}\right|$...

Chapter 13

Homomorphic Encryption

13.1 Introduction

Assume that Alice has some input x and there is some function f that Alice wants to compute on x but it is hard do compute on Alice hardware, but bob can do that, but Alice doesn't trust Bob with x (but f is known to both). How can we possibly hope to do that? We could ask Bob nicely not to do that, and that's what we actually do with all the cloud computing. We have a promise from the other guy to not reveal our data.

So we want some kind of scheme that Alice will send bob $\mathrm{Enc}(x)$, and we want some kind of scheme that will allow Bob to return $\mathrm{Enc}(f(x))$ without knowing anything about x Meaning that Alice has the sk and Bob has only the pk but still he succeed to form $\mathrm{Enc}(f(x))$ from $\mathrm{Enc}(x)$.

13.2 Definition

Definition 13.2.1 ($\mathcal{F}$-Homomorphic Encryption Scheme) Let $\mathcal{F} = \{\mathcal{F}_k\}$ class of functions $F_k \subseteq \{0,1\}^* \to \{0,1\}$. An $\mathcal{F}$-Homomorphic Encryption Scheme is a tuple of algorithms: $(\mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Dec}, \mathrm{Eval})$. Where $\mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Dec}$ are just like in regular PKE. And $\mathrm{Eval}_{\mathrm{pk}}\left(1^k, \underbrace{f}_{\in \mathcal{F}_k}, c_1, \ldots, c_l\right) \to c_f$ where $f : \{0,1\}^l \to \{0,1\}$ (Note that l is not a parameter of the scheme, we can use eval on any value of l).

Correctness: $\forall k, f : \{0,1\}^l \to \{0,1\} \in \mathcal{F}_k, , x_1, \ldots, x_l$, Now if:$(\mathrm{sk}, \mathrm{pk}) = \mathrm{KeyGen}\left(1^k\right)$ and $c_i = \mathrm{Enc}_{\mathrm{pk}}(x_i)$. And finally $c_f = \mathrm{Eval}_{\mathrm{pk}}(f, c_1, \ldots, c_l)$ and $x_f = \mathrm{Dec}_{\mathrm{sk}}(c_f)$ then:

$$x_f = f(x_1, \ldots, x_l)$$

We defined here 1-hop homomorphic encryption, we assume that the c_i are fresh and not evaluated ciphertexts. One could hope to not get just 1-hop and perhaps get many hops.

Moreover, note that right now, this definition can be solved trivially. We do not require c_f to have the same structure as $c_1, \ldots, c_l$. And in fact we can define:

$$c_f = (c_1, \ldots, c_l, f)$$

And the decryption will in fact compute f .

But the definition here doesn't cover the intuition. We wanted to help Alice compute f. So we want to require some none degeneracy condition that will solve this problem:

Definition 13.2.2 (Compactness) One option for the compactness definition is:$|C_f|$ is independent of f.
Another is: The computational complexity of$\mathrm{Dec}_{\mathrm{sk}}\left(1^k, \cdot\right)$ is going to be a fixed polynomial in k regardless the ciphertext.

The security definition is the regular PKE security definition.

Definition 13.2.3 (Fully Homomorphic Encryption (FHE)) We want $\mathcal{F}$ be the class of all functions. One needs to be careful because the complexity of Eval depends on the complexity of the function.

Moreover, when we say that Eval gets the function, we mean it gets it as a representation of a circuit. We could also ask what about C code for example? But we will not discuss this in class.

13.3 Additive homomorphism

We have $c_1 = \mathrm{End}(m_1)$ and $c_2 = \mathrm{Enc}(m_2)$ and we want to get: $c' = \mathrm{Enc}(m_1 \oplus m_2)$.

We consider a scheme similar to Regev, only we define: $\hat{A} = \begin{bmatrix} A \\ b^T \end{bmatrix}$ and $\underline{\hat{s}} = \begin{bmatrix} -\underline{s} \\ 1 \end{bmatrix}$ and we define:

$$\underline{\hat{s}} \cdot \hat{A} = \underline{e}^T$$

Using this notation the decryption looks like:

$$\underline{\hat{s}}^T \underline{\hat{C}} \approx m \cdot q/2$$

Candidate:

Now, if we have two of these ciphertexts and add them together:

$$\underline{\hat{C}}' = \underline{\hat{C}}_1 + \underline{\hat{C}}_2$$

Note that:

$$\underline{\hat{s}}^T \underline{\hat{C}}' = \underline{\hat{s}}^T \left(\underline{\hat{C}}_1 + \underline{\hat{C}}_2 \right) = \underline{\hat{s}}^T \underline{\hat{C}}_1 + \underline{\hat{s}}^T \underline{\hat{C}}_2 \approx \mu_1 q/2 + \mu q/2 = (\mu_1 \oplus \mu_2)\, q/2$$

Remarks 13.3.1 We cannot add as many as we want, because of the noise. Recall that the middle step is $\approx$, and as we add more and more items the approximation gets more and more worse. So we cannot do unlimited number of additions.

13.4 Multiplicative

Wait... How can we multiply vectors? Tensor product? Outer product? Well, there is an hard way that these can probably be used. But wouldn't it be easier to use matrices instead of vectors? And we will hide the message in the approximated eigenvalues of the matrix.

Now, let's assume that we have the following scheme. Instead of vectors, the ciphertextx will be matrices. So, we are going to multiply the ciphertext with the secret key(still a vector) and we want:

$$\underline{s}^T C \approx \mu \underline{s}^T$$

Meaning that we want μ to be "almost" an eigenvalue of the matrix C. Note that we require it to be "almost" eigenvalue, and not require equality because finding eigenvalues is an easy problem.

Note that now:

$$\underline{s}^T (C_1 + C_2) \approx (\mu_1 + \mu_2)\, \underline{s}^T$$

Meaning that addition still holds (again the approximation varies according to addition, we are much less exact).

But what about multiplication?

If we take two matrices and multiply them, and multiply by $\underline{s}^T$ what will we get?

$$\underline{s}^T (C_1 \cdot C_2) = \underbrace{\left(\underline{s}^T C_1\right)}_{\approx \mu_1 \underline{s}^T} \cdot C_2 \overset{?}{\approx} \mu_1 \underline{s}^T C_1 \approx \mu_1 \mu_2 \underline{s}^T$$

So, keeping this intuition in mind, let's try to see how we can hope to achieve this. We are going to do that using LWE.

We have:

$$\begin{bmatrix} \\ \tilde{A} \\ \\ \hline [\underline{b}^T] \end{bmatrix} \qquad \begin{bmatrix} \underline{v} \\ \hline [w] \end{bmatrix} = A \cdot \underline{r}$$
$$[w] = \underline{b}^T \cdot r / \underline{s}^T \cdot \underline{v}$$

From LWE, we can consider the entire matrix and it is indistinguishable from a uniform matrix.

Now consider:

$$A = \begin{bmatrix} \\ A \\ \\ \hline \underline{b}^T \end{bmatrix}$$

And:

$$\underline{s} = \begin{bmatrix} \underline{s} \\ \hline 1 \end{bmatrix}$$

So we have that:

$$\underline{s}^T A = \underline{e}^T$$

Previously we had:

$$\underline{c} = A\underline{r}$$

And this is how we generated ciphertexts that looked like vectors. And we had:

$$\begin{aligned} \underline{s}^T \underline{c} &= \underline{s}^T A \underline{r} \\ &= \underline{e}^T \underline{r} \end{aligned}$$

But how can we generate matrices? This matrices are just going to be bunch of this vectors. We generate a matrix C as: $C = A \cdot R$ (This is equivalent to generate the matrix by repeating the vectors multiplication). Due to LWE we going to have:

$$(A, C) \cong \text{Uniform}$$

So we have $A \in \mathbb{Z}_q^{(n+1)\times m}$ and $\underline{s} \in \mathbb{Z}_q^{n+1}$, so now: $C = A \cdot R \in \mathbb{Z}_q^{(n+1)\times N}$ where we can choose N to be whatever we want (we will choose it later on). Now:

$$\underline{s}^T \cdot C = \underline{s}^T A \cdot R = \underline{e}^T \cdot R$$

And we require that R will be small norm, and therefore:

$$\underline{s}^T \cdot C \approx 0$$

But how can we get:

$$\underline{s}^T \cdot C \approx \mu \underline{s}^T$$

Note that $(C + H)$ for H that is not dependent on the randomness of C, is indistinguishable from uniform. So we want to find H such that:

$$\underline{s}^T (C + H) = \underline{s}^T \cdot S + \underline{s}^T H \approx \underline{s}^T \cdot H$$

So, what will happen if we choose $N = n + 1$ and then:

$$H = \mu \cdot I$$

And we indeed get:

$$\underline{s}^T (C + H) \approx \mu \underline{s}^T$$

It is trivial that addition will work. Let's look at multiplication.

Now we have:

$$\underline{s}^T \cdot C = \mu \underline{s}^T + \underline{\varepsilon}^T$$

Where: $\underline{\varepsilon}^T = \underline{e}^T \cdot R$.

Now we have:

$$\begin{aligned} \underline{s}^T (C_1 C_2) &= \left(\underline{s}^T C_1\right) C_2 \\ &= \left(\mu_1 \underline{s}^T + \underline{\varepsilon}_1^T\right) C_2 \\ &= \mu_1 \left(\mu_2 \underline{s}^T + \underline{\varepsilon}_2^T\right) + \underline{\varepsilon}_1^T \cdot C_2 \\ &= \mu_1 \mu_2 \underline{s}^T + \mu_1 \underline{\varepsilon}_2^T + \underline{\varepsilon}_1^T C_2 \end{aligned}$$

But note that C_2 is not bounded, hence $\underline{\varepsilon}_1^T C_2$ can be large. So if only C_2 had low norm, we would have finish. But it can't be because it needs to be indistinguishable from uniform. So what can we do?

13.4.1 Reducing the norm of a matrix

Let's start by reducing the norm of the following matrix:

$$\begin{bmatrix} 1 & 3 \\ 2 & 5 \end{bmatrix} \ (\mod 8)$$

The stupidest thing we can do is to multiply it by a small element, but since we are working with finite field it is not clear what will happen, and we will loose some of the precision.

So what can we do? we can transform to binary by enlarging the dimensions:

$$\left[\begin{array}{cc} 0 & 0 \\ 0 & 1 \\ 1 & 1 \\ \hline 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{array}\right]$$

We want to find a matrix such that:

$$\begin{bmatrix} & ? & \end{bmatrix} \left[\begin{array}{cc} 0 & 0 \\ 0 & 1 \\ 1 & 1 \\ \hline 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{array}\right] = \begin{bmatrix} 1 & 3 \\ 2 & 5 \end{bmatrix} \ (\mod 8)$$

Note that the matrix needs to be 2×6. But what should it be? Consider the following:

$$\begin{bmatrix} 4 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 2 & 1 \end{bmatrix} \left[\begin{array}{cc} 0 & 0 \\ 0 & 1 \\ 1 & 1 \\ \hline 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{array}\right] = \begin{bmatrix} 1 & 3 \\ 2 & 5 \end{bmatrix} \ (\mod 8)$$

So although the original operation is not linear operation, the inverse operation is indeed linear (we even calculated the matrix related to the operation).

Let's denote the inverse operation's matrix with G:

$$G = \begin{bmatrix} 2^{\log q - 1} & 2^{\log q - 2} & \dots & 2 & 1 & 0 & & & \\ & & & & & 2^{\log q - 1} & \dots & 1 & \\ & & & & & & & & \ddots \end{bmatrix}$$

If q is not a power of two, we can simply pad it.
What are the dimensions of G? $G \in \mathbb{Z}_q^{\ell \times \ell \log q}$. So this is a short and wide matrix. Because G is the inverse of the binary decomposition we have seen before, we going to denote the decomposition by $G^{-1}(A) : \mathbb{Z}_q^{\ell \times \ell'} \to \{0,1\}^{\ell \log q \to \ell'}$ that does the binary decomposition as we've seen (again, this is not a matrix, because this is not a linear transformation!). But it still holds that:

$$G \cdot G^{-1}(A) = A$$

13.5 Homomorphic Encryption Scheme

Now we can construct an homomorphic encryption scheme:

- KeyGen $\left(1^k\right)$: Construct $A, \underline{s}$ (LWE instance). Output: pk $= A$, sk $= \underline{s}$.
- $\text{Enc}_{\text{pk}}(\mu)$: $C = AR + \mu G$, for $R \overset{R}{\in} \{0,1\}^{m \times N}$ where $N = (n+1)\log q$ (note that G is of dimension $(n+1) \times (n+1)\log q$).
- $\text{Dec}_{\text{sk}}(C)$: $\left\|\underline{s}^T C\right\|_\infty \leq q/8$ output 0, otherwise output 1.

Why is this an encryption scheme? Or why is it hold correctness?
Note that:

$$\begin{aligned} \underline{s}^T(AR + \mu G) &= \underline{s}^T AR + \mu \underline{s}^T G \\ &= \underline{e}^T R + \mu \underline{s}^T G \end{aligned}$$

If $\mu = 0$ then we have only $\underline{e}^T R$ but note that:

$$\left\|\underline{e}^T R\right\|_\infty \leq m \cdot B \left\|R\right\|_\infty \leq mB$$

where m is the bound on the error. Last step holds is because R is $\{0,1\}$ matrix.
Now, If $\mu = 1$ note that we also have $\underline{s}^T \cdot G$ in the term. We have:

$$\underline{s}^T G = \left[2^{\log q - 1} \cdot s_1, 2^{\log q - 2} s_1, \ldots, 2s_1, s_1, 2^{\log q - 1} s_2, \ldots, 2s_2, s_2, s^{\log q - 1} s_3, \ldots\right]$$

Note that because the way we construct it, one of the values must be in the range of $\left(\frac{q}{4}, \frac{q}{2}\right)$ (in our case it must be because $\underline{s}$ as a coordinate which is 1), so we have:

$$\left\|\underline{s}^T \cdot G\right\|_\infty \geq \frac{q}{4}$$

So if: $\mu = 0$ then:

$$\left\|\underline{s}^T C\right\| = \left\|\underline{e}^T R\right\| \leq mB$$

But if $\mu = 1$ then:

$$\left\|\underline{s}^T C\right\| \geq q/4 - mB$$

So we have correctness for fresh ciphertext.

What about homomorphic evaluation? Note that now: $\underline{s}^T C = \mu \cdot \underline{s}^T G + \underline{\varepsilon}^T$. In order to do multiplication we calculate: $C_1 G^{-1}(C_2)$. What will happen when we try to decrypt it?

$$\begin{aligned} \underline{s}^T\left(C_1 G^{-1}(C_2)\right) &= \left(\underline{s}^T C_1\right) G^{-1}(C_2) \\ &= \left(\mu_1 \underline{s}^T G + \underline{\varepsilon}_1^T\right) G^{-1}(C_2) \\ &= \mu_1 \underline{s}^T C_2 + \underline{\varepsilon}_1^T G^{-1}(C_2) \\ &= \mu_1\left(\mu_2 \underline{s}^T G + \underline{\varepsilon}_2^T\right) + \underline{\varepsilon}_1^T G^{-1}(C_2) \\ &= \mu_1 \mu_2 \underline{s}^T G + \underbrace{\mu_1 \underline{\varepsilon}_2^T + \underline{\varepsilon}_1^T G^{-1}(C_2)}_{\underline{\varepsilon}_{\text{mult}}^T} \end{aligned}$$

So, how big is going to be $\underline{\varepsilon}_{\text{mult}}^{T}$?

$$\left\|\underline{\varepsilon}_{\text{mult}}^{T}\right\| \leq \|\underline{\varepsilon}_2\| + N\|\underline{\varepsilon}_1\|$$

So $\underline{\varepsilon}_{\text{mult}}^{T}$ is in order of $(n+1)\log q$!

But AND is not universal, we want to get a NAND gate. Note that:

$$\text{NAND}(\mu_1, \mu_2) = 1 - \mu_1\mu_2$$

So note that we can calculate:

$$\text{HomNAND}(C_1, C_2) = G - C_1 \cdot G^{-1}(C_2)$$

Note that the noise here grows exactly like in multiplication.

But NAND is universal, so we can now calculate every function f. And we define to evaluate homomorphically:

- $\text{Eval}(f, C_1, \ldots, C_\ell)$: (The pk is not required here). Present f as NAND circuit, in topological order (Gate with inputs, then the next level that can be evaluated, etc. Until we get values to the output wire). In a more formal way, we are numbering all the wires in the circuit in a way such that for every gate the output wires are strictly larger then the input wires of the gate. For wire: $w = \ell+1, \ldots, \text{out}$, Let i, j be the inputs to the gate that computes w. Inductively C_i, C_j are well defined. Define: $C_w = \text{HomNAND}(C_i, C_j)$.$\Rightarrow C_{\text{out}}$ is an encryption of $f(x_1, \ldots, x_\ell)$.

13.6 The Noise

What happen to the noise when we calculate arbitrary circuits? At the beginning the noise is mB. But after the evaluation, the noise of the output is going to be $\|\underline{\varepsilon}_{\text{out}}\| \leq m \cdot B(N+1)^d$ where d is the depth of the NAND circuit representing the function f we are evaluating.

Note that we require that $\|\underline{\varepsilon}_{\text{out}}\| < q/8$ that is $m \cdot B(N+1)^d \leq q/8$ so in order to support circuits of depth d we want to choose: $q \approx N^{O(d)}$. But note that B (the bound of the noise) cannot get much smaller that q because then LWE become easier. We don't really know the hardness of LWE, but for the current best algorithms, we want: $d = \tilde{o}(N)$.

So, sadly the scheme as it is cannot support arbitrary circuits.