Security Protocols and Application — Final Exam

Solution

Philippe Oechslin and Serge Vaudenay

25.6.2014

- duration: 3h00
- no document allowed
- a pocket calculator is allowed
- communication devices are not allowed
- the exam invigilators will not answer any technical question during the exam
- the answers to each exercise must be provided on separate sheets
- readability and style of writing will be part of the grade
- do not forget to put your name on every sheet!

The exam grade follows a linear scale. In each exercise, each question has the same weight. Both exercises have the same weight.

1 Fully Homomorphic Encryption

Questions Q.1, Q.2, and Q.3 are independent.

Q.1 We consider a fully homomorphic encryption scheme

$$\mathsf{Gen} \to (\mathsf{sk}, \mathsf{pk}) \quad , \quad \mathsf{Enc}_{\mathsf{pk}}(x) \to y \quad , \quad \mathsf{Dec}_{\mathsf{sk}}(y) = x \quad , \quad \mathsf{Eval}_{\mathsf{pk}}(f, y) \to z$$

where $x \in \mathbf{Z}_2^n$ and f is a function from $\mathbf{Z}_2^n$ to itself (defined by a circuit with universal gates). We assume that we always have

$$\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Eval}_{\mathsf{pk}}(f, \mathsf{Enc}_{\mathsf{pk}}(x))) = f(x)$$

If $\mathsf{Dec}_{\mathsf{sk}}$ is a one-to-one mapping, show that the encryption is insecure: it is easy to decrypt with pk only.
HINT: use $f_i(x) = (x_i, \ldots, x_i)$.

If $\mathsf{Dec}_{\mathsf{sk}}$ is a one-to-one mapping, the encryption is deterministic: is cannot map one plaintext to many possible ciphertexts. So, given a ciphertext $c = \mathsf{Enc}_{\mathsf{pk}}(b, \ldots, b)$, we can compare it to the encryption of $(0, \ldots, 0)$ and $(1, \ldots, 1)$ to deduce the value of b. For each i, we define $f_i(x) = (x_i, \ldots, x_i)$. We have

$$\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Eval}_{\mathsf{pk}}(f_i, \mathsf{Enc}_{\mathsf{pk}}(x))) = (x_i, \ldots, x_i)$$

So, given $y = \mathsf{Enc}_{\mathsf{pk}}(x)$, by applying the previous decryption trick to $\mathsf{Eval}_{\mathsf{pk}}(f_i, y)$, we can deduce x_i. We do this for each i and obtain $x = (x_1, \ldots, x_n)$.

Q.2 We consider a fully homomorphic encryption scheme

$$\mathsf{Gen} \to (\mathsf{sk}, \mathsf{pk}) \quad , \quad \mathsf{Enc}_{\mathsf{pk}}(x) \to y \quad , \quad \mathsf{Dec}_{\mathsf{sk}}(y) = x \quad , \quad \mathsf{Eval}_{\mathsf{pk}}(f, y) \to z$$

We assume that we always have

$$\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Eval}_{\mathsf{pk}}(f, \mathsf{Enc}_{\mathsf{pk}}(x))) = f(x)$$

We consider the following protocol between $A(\mathsf{sk})$ and $B(\mathsf{pk})$, where A holds some secret sk and both participant hold pk, a function f, and some ciphertext $y = \mathsf{Enc}_{\mathsf{pk}}(x)$ for some unknown value x:

```
1:  for $i = 1$ to $n$ do
2:      $B$ flips a coin $b_i$
3:      if $b_i = 0$ then $B$ takes $z_i = \mathsf{Enc}_{\mathsf{pk}}(0)$, otherwise $B$ takes $z_i = \mathsf{Eval}_{\mathsf{pk}}(f, y)$
4:      $B$ sends $z_i$ to $A$
5:      $A$ computes $t_i = \mathsf{Dec}_{\mathsf{sk}}(z_i)$
6:      if $t_i = 0$, $A$ sets $b'_i = 0$, otherwise, $A$ sets $b'_i = 1$
7:      $A$ sends $\mathsf{Commit}(b'_i)$ to $B$
8:      $B$ reveals $b_i$ and how he computed $z_i$
9:      $A$ checks that $z_i$ was well computed
10:     $A$ opens his commitment
11:     $B$ checks that the commitment is correct and gets $b'_i$
12:     $B$ checks that $b_i = b'_i$
13: end for
```

If any verification fails the protocol aborts.

Q.2a When correctly executed and n large enough, show that the protocol succeeds *if and only if* $f(x) \neq 0$.

If the protocol is honestly executed, then $t_i = 0$ if $b_i = 0$ and $t_i = f(x)$ otherwise, due to the homomorphic property of the encryption. So, if $f(x) \neq 0$, the verification in Step 12 always passes. If $f(x) = 0$, the verification in Step 12 does not pass for $b_i = 1$. So, the probability to pass all iterations is 2^{-n}. For n large enough, this is negligible.

Q.2b Show that if b_i is always set to 1, then A can cheat and make the protocol succeed even for $f(x) = 0$.

If b_i is always set to 1, A could just skip the computation of t_i and set $b' = 1$. The verification in Step 12 would always pass.

Q.2c Show that if Step 9 is not done, then B can cheat and learn the value of x.

If Step 9 is not done, B could cheat by setting $b_i = 1$ and using f set to the function returning the ith bit of x, so that b'_i would be the ith bit of x.

Q.3 We consider an algorithm $\mathsf{Gen} \to (\mathsf{sk}, \mathsf{pk})$ such that pk defines two commutative groups G and Q, where Q consists of half of the elements of G. (We use multiplicative notations.) We assume that it is easy to compute products, to compare two elements, to find the unit 1, and to sample elements with uniform distribution in Q or in G. The public key pk further defines a fixed element $g \in G$ which is *not* in Q. We further assume that the secret key sk makes it easy to test if an element is in Q or not, but that this operation is hard when knowing only pk.
Given a bit $b \in \mathbf{Z}_2$, we define $\mathsf{Enc}_{\mathsf{pk}}(b) = g^b r$ where r is a random element in Q (which is freshly picked to encrypt b) and g is defined by pk. We define $\mathsf{Dec}_{\mathsf{sk}}(c) = 0$ if $c \in Q$ and $\mathsf{Dec}_{\mathsf{sk}}(c) = 1$ otherwise.

Q.3a Given a linear function L from $\mathbf{Z}_2^n$ to $\mathbf{Z}_2$, define an algorithm Eval_L such that

$$\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Eval}_L(\mathsf{Enc}_{\mathsf{pk}}(b_1),\ldots,\mathsf{Enc}_{\mathsf{pk}}(b_n))) = L(b_1,\ldots,b_n) \tag{1}$$

for all $b_1,\ldots,b_n \in \mathbf{Z}_2$.
HINT: for all $x \in G$, we have $x^2 \in Q$.

For information, the hint comes from that Q being a subgroup of G containing half of the elements of G, Q appears as the kernel of the projection p modulo Q from G to the group G/Q of order 2. In G/Q, we have only two elements $p(1)$ and $p(g)$. The neutral element is $p(1)$. Since $p(g)$ must have an inverse and $p(1)$ is neutral, this inverse can only be $p(g)$ itself: $p(g)^2 = p(1)$. Since p is a group homomorphism, $p(x^2) = p(x)^2 = p(1)$. So, x^2 is in the kernel of p, so $x^2 \in Q$.
If L is linear, it can be defined by $\lambda_1,\ldots,\lambda_n \in \mathbf{Z}_2$ such that

$$L(b_1,\ldots,b_n) = \lambda_1 b_1 \oplus \cdots \oplus \lambda_n b_n$$

for all $b_1,\ldots,b_n \in \mathbf{Z}_2$. So, we define

$$\mathsf{Eval}_L(c_1,\ldots,c_n) = c_1^{\lambda_1}\cdots c_n^{\lambda_n}$$

Q.3b Formally prove that Eval_L satisfies (1).

Clearly, if we write $\mathsf{Enc}_{\mathsf{pk}}(b_i) = g^{b_i} r_i$ with $r_i \in Q$, we have

$$\mathsf{Eval}_L(\mathsf{Enc}_{\mathsf{pk}}(b_1),\ldots,\mathsf{Enc}_{\mathsf{pk}}(b_n)) = g^{\lambda_1 b_1 + \cdots + \lambda_n c_n} r_1^{\lambda_1}\cdots r_n^{\lambda_n}$$

By writing $\lambda_1 b_1 + \cdots + \lambda_n c_n = L(b_1,\ldots,b_n) + 2q$ for some integer q, we obtain

$$\mathsf{Eval}_L(\mathsf{Enc}_{\mathsf{pk}}(b_1),\ldots,\mathsf{Enc}_{\mathsf{pk}}(b_n)) = g^{L(b_1,\ldots,b_n)} g^{2q} r_1^{\lambda_1}\cdots r_n^{\lambda_n}$$

Since $g^2 \in Q$, we have that $g^{2q} r_1^{\lambda_1}\cdots r_n^{\lambda_n} \in Q$. So,

$$\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Eval}_L(\mathsf{Enc}_{\mathsf{pk}}(b_1),\ldots,\mathsf{Enc}_{\mathsf{pk}}(b_n))) = L(b_1,\ldots,b_n)$$

2 Zerocoin, the really anonymous cryptocurrency

Bitcoin

Q.1 Bitcoin uses a proof-of-work function which is used to sign blocks of transactions.
List three properties that the proof-of-work function must have and give a motivation for each property.

- *Hard to calculate, so as to control the rate at which Bitcoins are crated*
- *Easy to verify, in order to run the protocol efficiently*
- *Free of backdoors, so that users can trust the system*

Q.2 Recently a group of miners (a mining pool) reached 51% of the total mining power of the network.

- Describe an attack that they could have carried out in order enrich themselves.
- Explain why it is probably not in their interest to carry out such an attack.

- 1: *They spend some Bitcoins to buy electronic goods (e.g. an e-book on cryptography).*
  2: *They let the network create a block that confirms the transaction and even a few blocks more.*
  3: *The vendor sees the transaction and the following blocks and releases the goods.*
  4: *The miners now create an alternative block that conflicts with the original block and says the Bitcoins were spent for something else, e.g. transferring them to themselves.*
  5: *They propagate the conflicting block and as they have the majority, they can resolve the conflict in favor of the alternative block.*
  6: *They have succeeded in spending the coins twice.*
- *All transactions and conflict resolution being public, large irregularities would be noticed. People would lose trust in Bitcoin. The value of Bitcoins as well as of the mining equipment would be lost.*

Q.3 If a seller does not wait for a confirmation of a transaction to appear in a new block, it may be possible to invalidate a transaction even with very limited or no mining power at all.
Can you describe an attack were a transaction signed by the buyer and propagated by the seller does not end up in the next chain block ?

- *Race attack: At the same time the buyer propagates an alternative transaction to many users, hoping that they will include that alternative transaction in the block chain and not the original transaction.*
- *Witholding attack: A miner can try to mine a block that contains an alternative transaction and wait before publishing the block. As soon as the seller has accepted the original transaction (without waiting for confirmation), the miner releases the block that invalidates the original transaction. This only works if the attacker succeeds in mining a block and no other miner succeeds before the attack is carried out.*

Q.4 A simple Bitcoin laundering service could be the following: Customers who wish to wash their coins transfer them to the single account of a laundry service. After a random delay the laundry service transfers the coins to a destination account chosen by the customer.
Describe two attacks that a dishonest laundry service could run against its customers.

- *It could just keep the money (and use an honest laundry service to spend it anonymously).*
- *It could disclose the relations between input accounts and output accounts to third parties.*

Zerocoin

Q.5 Zerocoin uses cryptographic accumulators which can store a set of elements. Given only the accumulator, it is not possible to find the elements that it holds. However, given the accumulator and one element, it is possible to prove that the element is in the accumulator.
A simple accumulator can be created with multiplication. You add a value x to the accumulator A by multiplying the value and the accumulator:

$$A_{n+1} = A_n \cdot x$$

- How can you prove that a value v is stored in the accumulator A ?
- What conditions must hold to make it hard to find the elements that are in the accumulator ?
- What makes this accumulator unpractical to implement ?

- *v must divide A*
- *A must be large enough to be hard to factorize*
- *A becomes bigger every time you add an element to it.*

Q.6 The accumulator suggested for Zerocoin is of the form

$$A_{n+1} = A_n^x \mod P$$

- How must the values for x be chosen for the accumulator to work properly ?
- How can you create a proof that a certain value v is in the accumulator ?

- *x must be prime (or at least co-prime to all other possible values of x).*
- *You create an accumulator w that contains all values stored in A except for v. You present w and v. Anybody can verify that $w^x = A \mod P$.*

Q.7 When spending a Zerocoin, you have to create a zero knowledge proof (ZKP) that shows that you have the right to spend a coin.

- What is the information that you need to keep when you mint a Zerocoin such that you can generate the ZKP when you want to spend a coin ?
- What exactly does the ZKP prove ?

> - *The serial number S and the random r, used to create the coin (commitment) c.*
> - *You know a c that is in the accumulator and that c has the serial number S. More precisely you reveal S and prove that you know r in* $c = g^S h^r$ *for one of the* c_i *in the accumulator.*

Q.8 More precisely, Zerocoin uses *non-interactive zero-knowledge proofs of knowledge* (NIZKPoK) and *signatures of knowledge* of messages (ZKSoK).

- Explain the goal of a ZKSoK.
- What is it used for in Zerocoin ?

> - *A ZKSoK proves knowledge of some information (e.g. a logarithm) and binds this proof to a public message.*
> - *In Zerocoin, a ZKSoK is used to prove the legitimacy of a* spend *operation while binding it to a* mint *operation. This creates a link between the* mint *operation that froze a Bitcoin and the* spend *operation that releases the same Bitcoin into another wallet.*
>   *In the same way that you sign a Bitcoin transaction with your private key to prove that you are legitimated to spend the money from your wallet, you sing a* spend *operation with your knowledge to prove that you are legitimated to spend the money frozen in a Zerocoin.*

Q.9 Although Zerocoins are mathematically untraceable there can still arise scenarios where it will be possible to link a spend operation to the mint operation of the same user.
Describe two such scenarios

> - *There are 0 Zerocoins in the accumulator (all previously minted coins have been spent). A coin is minted and then a coin is spent. Obviously the last* spend *and the last* mint *have been done by the same person.*
> - *An particular amount of* mints *are made from single account. At a later point in time, the exact same amount of Zerocoins are spent into another single account.*